1.Niño 3.4 index

The *Niño 3.4 anomalies* may be thought of as representing the average equatorial sea surface temperatures (SSTs) across the Pacific from about the dateline to the South American coast (5N-5S, 170W-120W). The Niño 3.4 index typically uses a 3-month running mean, and El Niño or La Niña events are defined when the Niño 3.4 SSTs exceed +/- 0.5°C for a period of 5 months or more. Check Equatorial Pacific Sea Surface Temperatures for more about the Niño 3.4 index.

In this problem set, you will use the sea surface temperature (SST) data from NOAA. Download the netCDF4 file (NOAA_NCDC_ERSST_v3b_SST.nc) here.

1.1 [10 points] Compute monthly climatology for SST from Niño 3.4 region, and subtract climatology from SST time series to obtain anomalies.

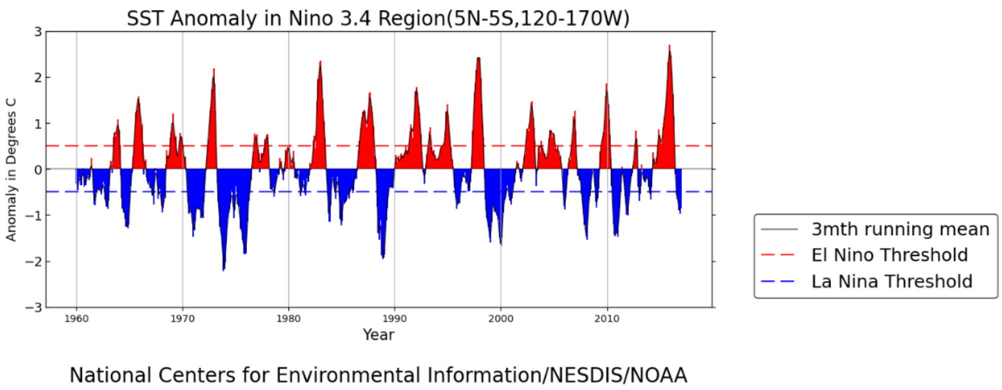
1.2 [10 points] Visualize the computed Niño 3.4. Your plot should look similar to this one.

Program:

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xarray as xr

#Q1.1
# Load dataset
ds = xr.open_dataset('NOAA_NCDC_ERSST_v3b_SST.nc', engine="netcdf4")

# Calculate anomalies and rolling mean
group_data = ds.sst.groupby('time.month')
sst_anom = group_data - group_data.mean(dim='time')
ano = sst_anom.sel(lat=slice(-5, 5), lon=slice(190, 240)).mean(dim=['lat', 'lon'])

ds_anom_rolling = sst_anom.rolling(time=3, center=True).mean()
g = ds_anom_rolling.sel(lat=slice(-5, 5), lon=slice(190, 240)).mean(dim=['lat', 'lon'])
#Q1.2
# Convert to DataFrames
y = ano.to_dataframe()
y1 = g.to_dataframe()

# Create subplots
fig, ax = plt.subplots(figsize=(12, 5))

# Bar chart
colors = ['r' if value >= 0 else 'blue' for value in y['sst']]
```

```
ax.bar(y.index, y['sst'], width=50, color=colors)
ax.grid(axis='x')
ax.tick_params(axis='y', labelsize=13, direction='in', length=4)
ax.tick_params(axis="y", direction="in", which="minor", length=2)
ax.set_ylim(-3, 3)
ax.set_yticks(np.arange(-3, 3.2, 1))
ax.set_xlabel('Year', fontsize=15)
ax.set_ylabel('Anomaly in Degrees C', fontsize=13)
ax.set_title('SST Anomaly in Nino 3.4 Region(5N-5S,120-170W)', fontsize=20)
ax.axhline(y=0, color='k', linestyle='-', alpha=0.3)
ax.axhline(y=0.5, color='r', linestyle=(0, (9, 4)), alpha=0.9, label='El Nino Threshold')
ax.axhline(y=-0.5, color='b', linestyle=(0, (9, 4)), alpha=0.9, label='La Nina Threshold')

# Second y-axis for the line plot
ax2 = ax.twinx()
ax2.plot(y1.index, y1['sst'], color='k', alpha=0.8, linewidth=1, label='3mth running mean')
ax2.set_ylim(ax.get_ylim())

# Combine legends for both axes
lines, labels = ax.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax.legend(lines2 + lines, labels2 + labels, loc='lower left', bbox_to_anchor=(1.05, 0),fontsize=18,edgecolor='black')

# Hide right y-axis tick labels
ax2.set_yticklabels([])

# Add a text annotation below the x-axis
ax.text(0.5, -0.25, 'National Centers for Environmental Information/NESDIS/NOAA', fontsize=20, ha='center', va='center', transform=ax.transAxes)

# Show the plot
plt.show()
```

The Result



Problem-solving ideas

1.Data Loading:

Use the xarray library to open the NetCDF file and store it as the ds dataset.

2.Anomaly Calculation and Rolling Mean:

Use the groupby method to group the SST data by month.Calculate anomalies by subtracting the monthly mean from the monthly SST values.Apply a rolling mean with a time window of 3.

3.Data Extraction:

Select the region of interest (Nino 3.4 Region).Compute the average anomalies over the latitude and longitude in this region, resulting in ano and g.

4.Conversion to DataFrame:

Convert ano and g to DataFrame objects, named y and y1.

5.Plotting:

Create a figure and axis object.Use the bar method to plot a bar chart, with different colors based on the positive or negative values of anomalies.Add grid lines, labels, a title, and threshold lines for El Nino and La Nina.Add a second y-axis and use the plot method to draw a 3-month running mean line.Merge legends from both y-axes and adjust the display position.

6.Graphical Enhancements:

Hide tick labels on the second y-axis as they coincide with the first y-axis.

Add a text annotation below the x-axis to indicate the data source.

2. Earth's energy budget

In this problem set, you will analyze top-of-atmosphere (TOA) radiation data from NASA's CERES project. Read this post for more about Earth's energy budget.

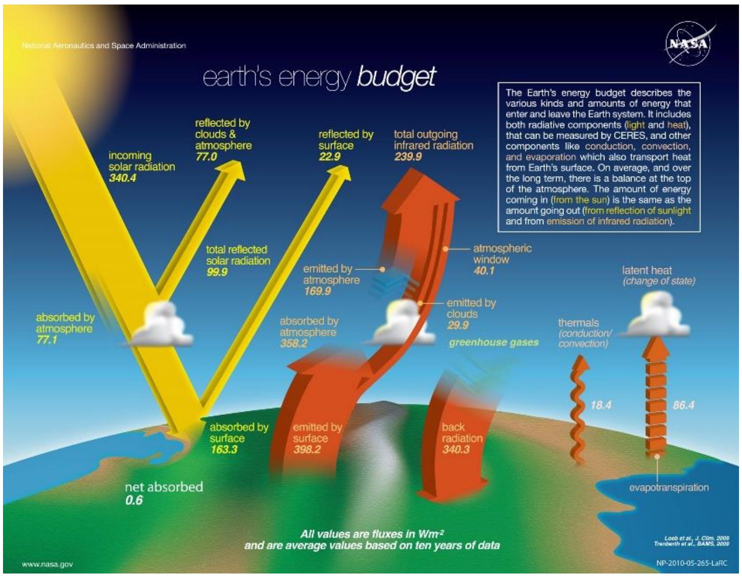


Figure source

Download the data (`CERES_EBAF-TOA_200003-201701.nc`) here. The size of the data file is `702.5 MB`. It will take a minute or two to download. Start by importing `xarray`, `numpy`, and `matplotlib`.

2.1 [5 points] Make a 2D plot of the time-mean TOA longwave, shortwave, and solar radiation for all-sky conditions. Add up the three variables above and verify (visually) that they are equivalent to the TOA net flux.

2.2 [10 points] Calculate and verify that the TOA incoming solar, outgoing longwave, and outgoing shortwave approximately match up with the cartoon above.

[Hint: Consider calculating the area of each grid]

2.3 [5 points] Calculate and plot the total amount of net radiation in each 1-degree latitude band. Label with correct units.

2.4 [5 points] Calculate and plot composites of time-mean outgoing shortwave and longwave radiation for low and high cloud area regions. Here we define low cloud area as `≤25%` and high cloud area as `≥75%`. Your results should be 2D maps.

2.5 [5 points] Calculate the global mean values of shortwave and longwave radiation, composited in high and low cloud regions. What is the overall effect of clouds on shortwave and longwave radiation?

Program

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xarray as xr

# Load dataset
ds = xr.open_dataset('CERES_EBAF-TOA_200003-201701.nc', engine="netcdf4")
#Q2.1
import xarray as xr
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
ds = xr.open_dataset('CERES_EBAF-TOA_200003-201701.nc')

# Extract relevant variables
toa_sw_all = ds['toa_sw_all_mon']
toa_lw_all = ds['toa_lw_all_mon']
solar_mon = ds['solar_mon']
toa_net_all = ds['toa_net_all_mon']

# Calculate time mean for each variable
toa_sw_mean = toa_sw_all.mean(dim='time')
toa_lw_mean = toa_lw_all.mean(dim='time')
solar_mean = solar_mon.mean(dim='time')
toa_net_mean = toa_net_all.mean(dim='time')

# Calculate the sum of shortwave, longwave, and solar to verify TOA net flux
sum_radiation =    solar_mean - toa_sw_mean - toa_lw_mean

# Create separate 2D plots for each variable
fig, axes = plt.subplots(2, 3, figsize=(18, 10))

# TOA Shortwave
toa_sw_mean.plot(ax=axes[0, 0], cbar_kwargs={'label': 'TOA Shortwave (W/m^2)'})
axes[0, 0].set_title('TOA Shortwave')

# TOA Longwave
toa_lw_mean.plot(ax=axes[0, 1], cbar_kwargs={'label': 'TOA Longwave (W/m^2)'})
axes[0, 1].set_title('TOA Longwave')

# Solar Radiation
```

```
solar_mean.plot(ax=axes[0, 2],cbar_kwargs={'label': 'Solar Radiation (W/m^2)'})
axes[0, 2].set_title('Solar Radiation')

# Sum of Shortwave, Longwave, and Solar Radiation
sum_radiation.plot(ax=axes[1, 0],cbar_kwargs={'label': 'Sum of Shortwave, Longwave, and Solar (W/m^2)'})
axes[1, 0].set_title('Sum of Radiation')

# TOA Net Flux
toa_net_mean.plot(ax=axes[1, 1],    cbar_kwargs={'label': 'TOA Net Flux (W/m^2)'})
axes[1, 1].set_title('TOA Net Flux')

# Blank subplot for better layout
axes[1, 2].axis('off')

plt.tight_layout()
plt.show()

#Q2.2
lat = np.radians(toa_sw_all['lat'])
lon = np.radians(toa_sw_all['lon'])

# Calculate area weights
cos_lat = np.cos(lat)
area_weights = cos_lat / cos_lat.mean()

# Calculate area-weighted average for each variable
toa_sw_avg = (toa_sw_all * area_weights).mean(dim=['lon', 'lat'])
toa_lw_avg = (toa_lw_all * area_weights).mean(dim=['lon', 'lat'])
solar_avg = (solar_mon * area_weights).mean(dim=['lon', 'lat'])

# Print the calculated values
print(f"TOA Incoming Solar Flux: {solar_avg.mean().values} W/m^2")
print(f"TOA Outgoing Longwave Flux: {toa_lw_avg.mean().values} W/m^2")
print(f"TOA Outgoing Shortwave Flux: {toa_sw_avg.mean().values} W/m^2")

#Q2.3
# Calculate area-weighted net radiation for each latitude band
net_radiation_by_lat = ds.toa_net_all_mon.mean(dim='lon').plot.contourf(x='time', levels=100, cmap='RdYlBu')
net_radiation_by_lat.colorbar.set_label('W·m^-2')

# Plot the total amount of net radiation in each 1-degree latitude band
plt.xlabel('Year')
```

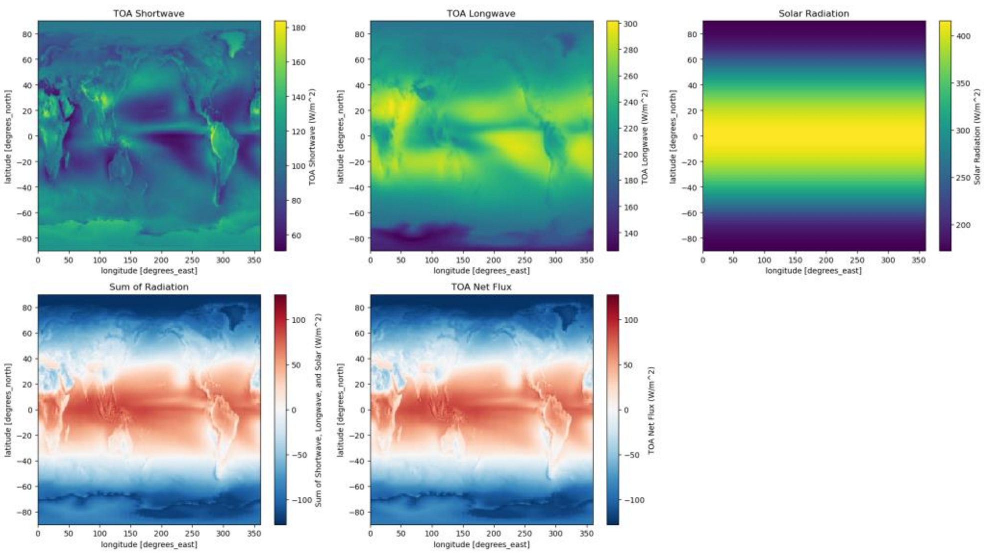
```
plt.ylabel('Latitude (degrees)')
plt.title('Annual Mean Net Radiation in Each 1-degree Latitude Band')

plt.show()

#Q2.4
# Extract relevant variables
cloud_area = ds['cldarea_total_daynight_mon']

# Define low and high cloud area regions
low_cloud_area = cloud_area.mean(dim='time') <= 25
high_cloud_area = cloud_area.mean(dim='time') >= 75

# Calculate time-mean outgoing shortwave and longwave radiation for low and high cloud areas
toa_sw_low_cloud = toa_sw_all.where(low_cloud_area).mean(dim='time')
toa_lw_low_cloud = toa_lw_all.where(low_cloud_area).mean(dim='time')

toa_sw_high_cloud = toa_sw_all.where(high_cloud_area).mean(dim='time')
toa_lw_high_cloud = toa_lw_all.where(high_cloud_area).mean(dim='time')

# Plot the composites of time-mean outgoing shortwave radiation
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
toa_sw_low_cloud.plot(cmap='YlGnBu', vmin=toa_sw_all.min(), vmax=toa_sw_all.max(),cbar_kwargs={'label': 'Outgoing SW Radiation (W/m^2)'})
plt.title('Low Cloud Area')

plt.subplot(1, 2, 2)
toa_sw_high_cloud.plot(cmap='YlGnBu', vmin=toa_sw_all.min(), vmax=toa_sw_all.max(),cbar_kwargs={'label': 'Outgoing SW Radiation (W/m^2)'})
plt.title('High Cloud Area')

plt.suptitle('Time-Mean Outgoing Shortwave Radiation for Low and High Cloud Area Regions')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

# Plot the composites of time-mean outgoing longwave radiation
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
toa_lw_low_cloud.plot(cmap='YlOrRd', vmin=toa_lw_all.min(), vmax=toa_lw_all.max(),cbar_kwargs={'label': 'Outgoing LW Radiation (W/m^2)'})
plt.title('Low Cloud Area')
```

```
plt.subplot(1, 2, 2)
toa_lw_high_cloud.plot(cmap='YlOrRd', vmin=toa_lw_all.min(), vmax=toa_lw_all.max(),cbar_kwargs={'label': 'Outgoing LW Radiation (W/m^2)'})
plt.title('High Cloud Area')

plt.suptitle('Time-Mean Outgoing Longwave Radiation for Low and High Cloud Area Regions')
plt.tight_layout(rect=[0, 0.03, 1, 0.95])
plt.show()

#Q2.5
# Extract relevant variables and coordinates
toa_sw_all = ds['toa_sw_all_mon']
toa_lw_all = ds['toa_lw_all_mon']
lat = np.radians(toa_sw_all['lat'])
lon = np.radians(toa_sw_all['lon'])

# Calculate area-weighted global mean values for shortwave and longwave radiation in low and high cloud areas
global_sw_low_cloud = (toa_sw_all * area_weights).where(low_cloud_area).mean(dim=['lat', 'lon'])
global_lw_low_cloud = (toa_lw_all * area_weights).where(low_cloud_area).mean(dim=['lat', 'lon'])

global_sw_high_cloud = (toa_sw_all * area_weights).where(high_cloud_area).mean(dim=['lat', 'lon'])
global_lw_high_cloud = (toa_lw_all * area_weights).where(high_cloud_area).mean(dim=['lat', 'lon'])
# Plot the four time series on a single graph
plt.figure(figsize=(10, 6))

# Plot global mean TOA shortwave radiation in low cloud areas
plt.plot(global_sw_low_cloud['time'], global_sw_low_cloud, label='TOA SW Low Cloud', color='blue')

# Plot global mean TOA longwave radiation in low cloud areas
plt.plot(global_lw_low_cloud['time'], global_lw_low_cloud, label='TOA LW Low Cloud', color='orange')

# Plot global mean TOA shortwave radiation in high cloud areas
plt.plot(global_sw_high_cloud['time'], global_sw_high_cloud, label='TOA SW High Cloud', color='green')

# Plot global mean TOA longwave radiation in high cloud areas
plt.plot(global_lw_high_cloud['time'], global_lw_high_cloud, label='TOA LW High Cloud',
```

```
color='red')

# Set plot title and labels
plt.title('Global Mean TOA Radiation in Low and High Cloud Areas')
plt.xlabel('Time')
plt.ylabel('Radiation (W/m^2)')

# Add legend
plt.legend(loc='lower left',bbox_to_anchor=(1.05, 0))

# Show the plot
plt.show()
```

The Result

Q2.1

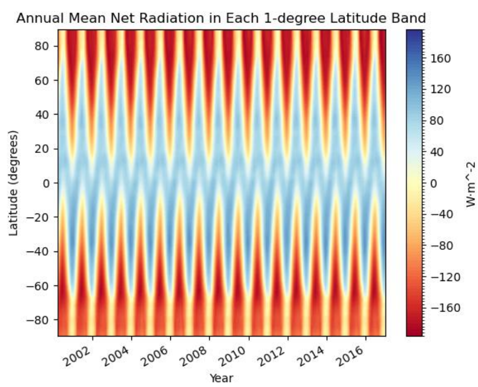


Two images visually identical

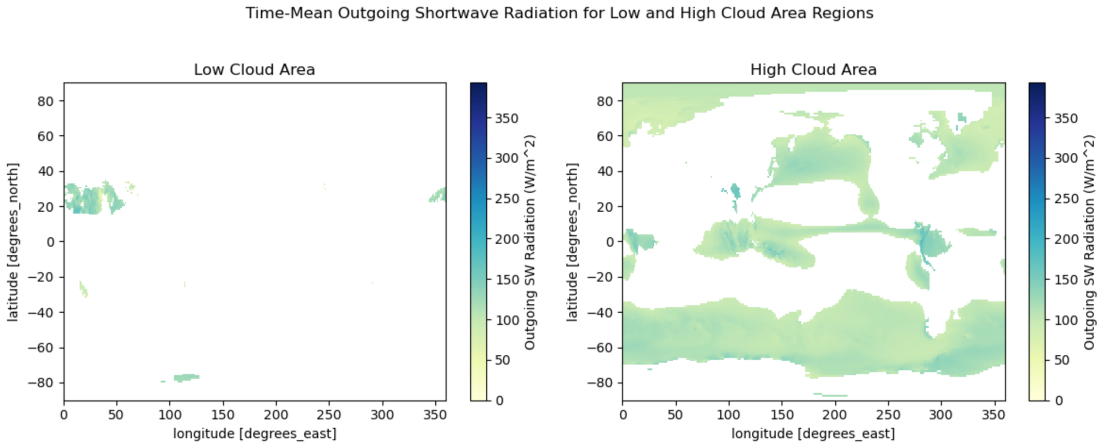
Q2.2

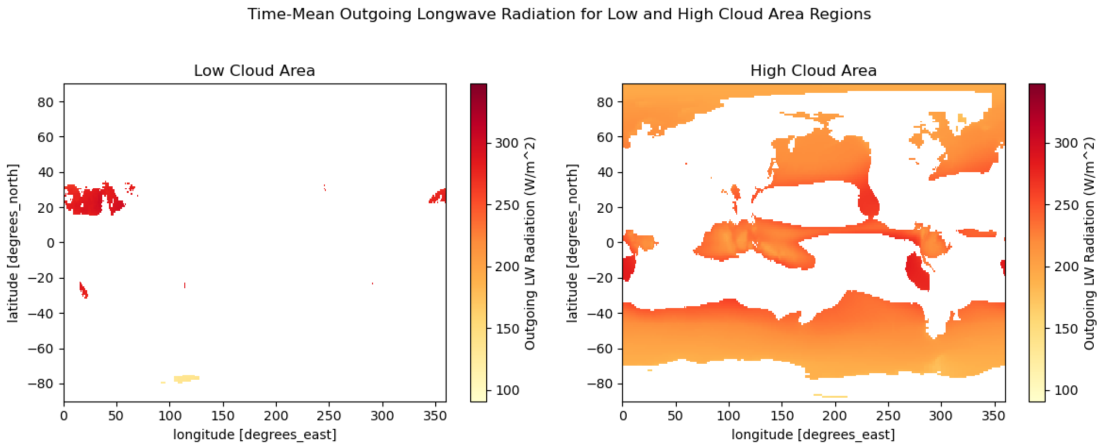
```
TOA Incoming Solar Flux: 340.2851257324219 W/m^2
TOA Outgoing Longwave Flux: 240.2679901123047 W/m^2
TOA Outgoing Shortwave Flux: 99.13904571533203 W/m^2
```

Q2.3

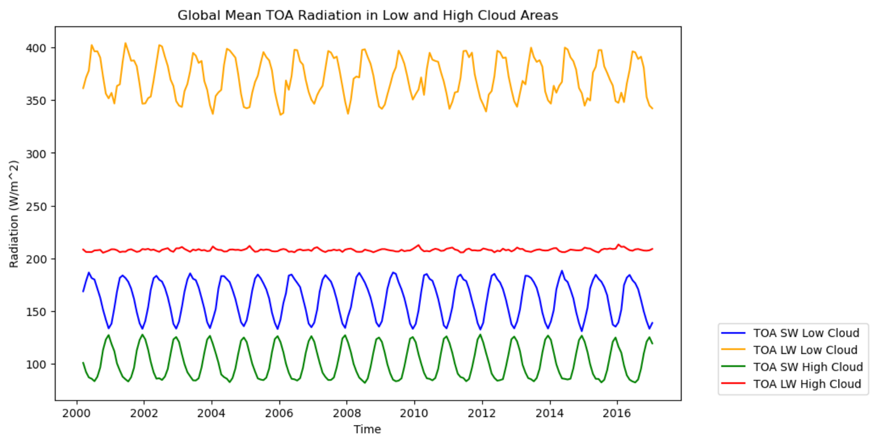


Q2.4





Q2.5



The sum of short waves remains basically unchanged in both high and low clouds, ensuring the balance of short wave radiation. Long waves do not change with periodic fluctuations in high clouds, while in low clouds, they fluctuate with periodic fluctuations.

Problem-solving ideas

1.Loading and Time-Mean Analysis:

Load the CERES dataset using xarray. Extract relevant variables: toa_sw_all_mon, toa_lw_all_mon, solar_mon, and toa_net_all_mon. Calculate time means for each variable.

2.2D Plotting:

Create subplots to display individual 2D plots for TOA shortwave, longwave, solar radiation, and the sum of shortwave, longwave, and solar radiation. Utilize contourf and plot functions for visualization. Customize each subplot with titles, labels, and colorbars.

3.Global Mean and Area-Weighted Analysis:

Calculate area weights based on latitude. Calculate area-weighted global mean values for TOA shortwave, longwave, and solar radiation. Print the calculated values.

4.Latitude Band Analysis:

Contour plot the area-weighted net radiation for each latitude band. Display the total amount of net radiation in each 1-degree latitude band over the years.

5.Cloud Area and Radiation Analysis:

Extract the cldarea_total_daynight_mon variable. Define low and high cloud area regions based on a threshold (e.g., 25% and 75%). Calculate time-mean outgoing shortwave and longwave radiation for low and high cloud areas. Plot composites of time-mean outgoing shortwave and longwave radiation for low and high cloud area regions.

6.Global Mean TOA Radiation Analysis:

Calculate area-weighted global mean values for shortwave and longwave radiation in low and high cloud areas. Plot the four time series (TOA SW/LW in Low/High Cloud Areas) on a single graph.

3. Explore a netCDF dataset

Browse the NASA's Goddard Earth Sciences Data and Information Services Center (GES DISC) website. Search and download a dataset you are interested in. You are also welcome to use data from your group in this problem set. But the dataset should be in `netCDF` format, and have temporal information.

3.1 [5 points] Plot a time series of a certain variable with monthly seasonal cycle removed.

3.2 [5 points] Make at least `5` different plots using the dataset.

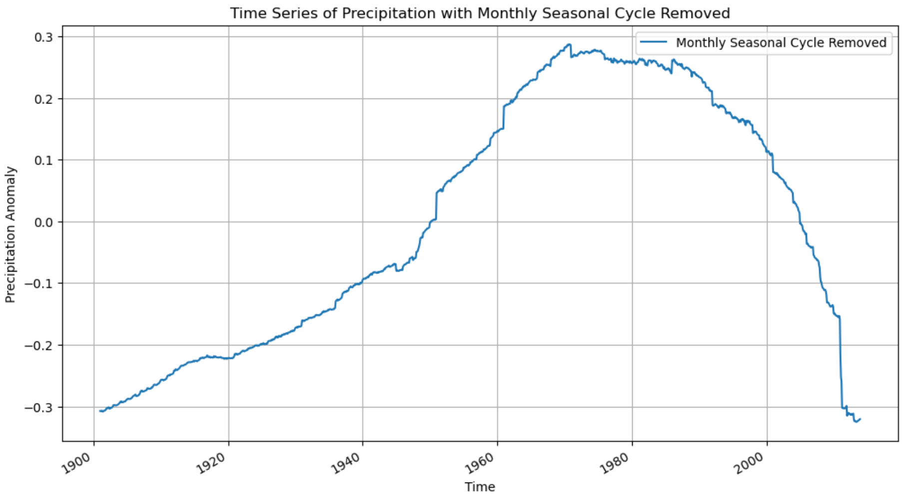
Program

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import xarray as xr

# Load dataset
ds = xr.open_dataset('precip.mon.nobs.1x1.v7.nc', engine="netcdf4")

#Q3.1
# Extract the variable of interest (precipitation)
precip = ds['precip']

# Calculate the monthly climatology (mean) to remove the seasonal cycle
monthly_climatology = precip.groupby('time.month').mean(dim='time')

# Remove the monthly climatology from the original data
anomaly = precip.groupby('time.month') - monthly_climatology

# Plot the time series of the variable with the monthly seasonal cycle removed
plt.figure(figsize=(12, 6))
anomaly.mean(dim=['lat', 'lon']).plot(label='Monthly Seasonal Cycle Removed')

plt.xlabel('Time')
plt.ylabel('Precipitation Anomaly')
plt.title('Time Series of Precipitation with Monthly Seasonal Cycle Removed')
plt.legend()
plt.grid(True)
plt.show()

#Q3.2
# Extract years 1910, 1930, 1950, 1970, 1990, 2010
years = [1910, 1930, 1950, 1970, 1990, 2010]
```

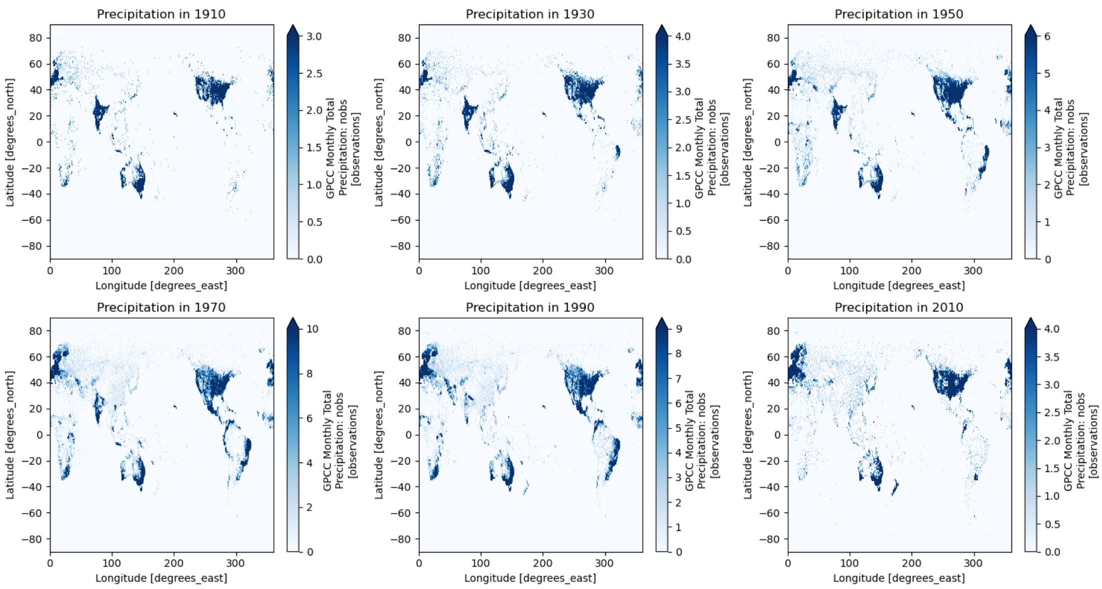
```
selected_years = precip.sel(time=precip['time.year'].isin(years))

# Plot 2D maps for each selected year in a 2x3 grid
fig, axes = plt.subplots(2, 3, figsize=(15, 8))

for i, year in enumerate(years):
    row, col = divmod(i, 3)
    ax = axes[row, col]
    selected_years.sel(time=f'{year}-01-01').plot(ax=ax, cmap='Blues', robust=True)
    ax.set_title(f'Precipitation in {year}')

plt.tight_layout()
plt.show()
```

The Result

Q3.1



Q3.2



Problem-solving ideas

1.Programming Approach:

Loading and Data Preparation:Load the precipitation dataset using xarray. Extract the precipitation variable (precip).

2.Monthly Seasonal Cycle Removal:

Calculate the monthly climatology (mean) to represent the seasonal cycle. Subtract the monthly climatology from the original data to obtain the anomaly (seasonal cycle removed).

3.Time Series Plotting:

Create a time series plot of the precipitation anomaly, averaged over all latitudes and longitudes. Customize the plot with labels, title, legend, and grid for clarity.

4.Selected Year Analysis:

Define a list of selected years (1910, 1930, 1950, 1970, 1990, 2010). Extract data for these years from the original precipitation dataset. Plot 2D maps for each selected year in a 2x3 grid. Utilize the plot function with a blue colormap for visualization. Customize each subplot with titles.

5.Visualization and Presentation:

Adjust the figure size for visual appeal. Arrange subplots in a 2x3 grid for clear presentation. Utilize appropriate colormaps and styling for better visualization.