FYS-3001: Physics of remote sensing

Name: Nea Schrøder

UiT - Norges Arktiske Universitet

Spring 2025

0.1 Task 1: Fundementals and thermal sensing

1. What is spectral and spatial resolution? How are these two resolutions connected?

Spectral resolution is a measure of how adept an instrument is at distinguishing between wavelengths (/frequencies) in the electromagnetic spectrum. An instrument with a small number of wide spectral bands will have a low spectral resolution, while an instrument with a high amount of narrow spectral bands will have a high spectral resolution Copernicus SentiWiki ((n.d.)).
Spatial resolution is a measure of the instruments ability to separate between points in space. The pixel size, which is most often given in meters, is a measure of how closely you will be able to detect details in the image, so for a 20 meters spatial resolution you will not be able to see detail less that 20 meters Borotkanych ((March 28, 2025)).
Because of technological limits in remote sensing instruments and data handling capabilities there is often a trade-off between spatial and spectral resolution. We can then use information about what we want to study to decide if we want a higher spectral or spatial resolution. For example, in atmospheric studies a higher spectral resolution should be prioritized. In some cases, like surface studies we are interested in both resolutions and will then have use for instruments such as imaging spectrometers Elachi and Van Zyl ((2021)).

2. What is a BRDF? Why are they important for Earth observation from satellites?

BRDFs are Bi-directional reflectance distribution functions that, as the name suggests, are dependent on both illumination and viewing geometry. This means that for different angles of incidence and observation angles we will get different optical scatter. The BRDF describes something we already see each day in real life: objects look different when light illuminates them from different angles. In figure 1 we see an example of this. The left image shows the trees from a viewpoint in the same direction as the illumination is coming from (back-scatter), while on the right the viewer is opposite to the illumination source.

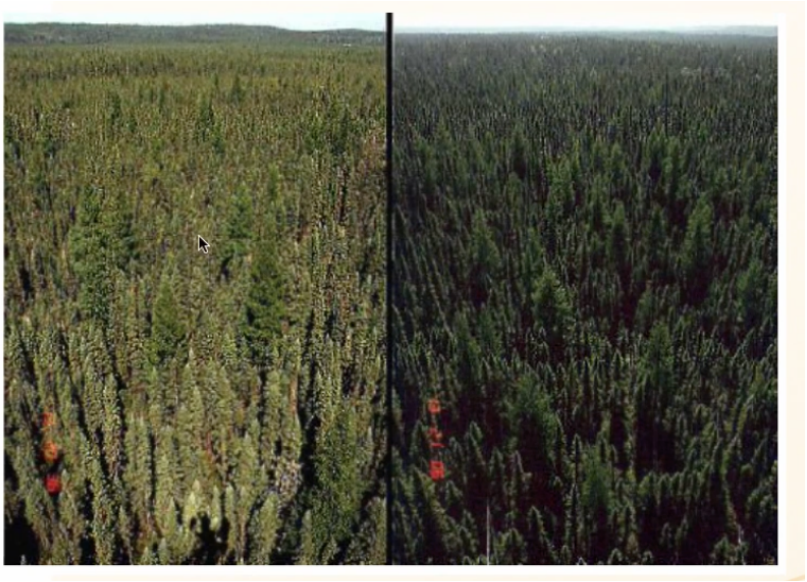

Figure 1: Pine trees at different illumination angle, from Johansson ((2025a))

The BRDF depends on different properties of the surface, both structural and optical, such as height structures and roughness. In Earth observation the BRDF is useful to account for corrections of atmospheric interference of the radiation, characterizations of different surfaces and finding albedo. Johansson ((2025a))

3. What is the difference between a scanning system and a pushbroom system? And why would one be preferable over the other?

A scanning system is a system that only uses a single detector, with a mirror used to project a surface resolution element onto the detector. As the system moves along-track the mirror sweeps in the across-track direction to image this area. The pushbroom system does not have these moving parts but instead has a linear array of detectors across the flight path. This means that they capture all the swath width at once. For both systems it is the movement of the spacecraft that makes it possible to image in the along-track direction. The pushbroom system is often preferable to the scanning system for several reasons. One big reason is that the pushbroom system avoids moving parts that are prone to wear out over time, and also has a longer dwell time because it does not have to sweep over the whole swath as the spacecraft moves. This system also gives better geometric fidelity than the scanning systems as the scanning systems will have an enlargement towards the sides of the swath Elachi and Van Zyl ((2021)). Figure 2 (from Elachi and Van Zyl ((2021))) shows illustrations of the different systems.

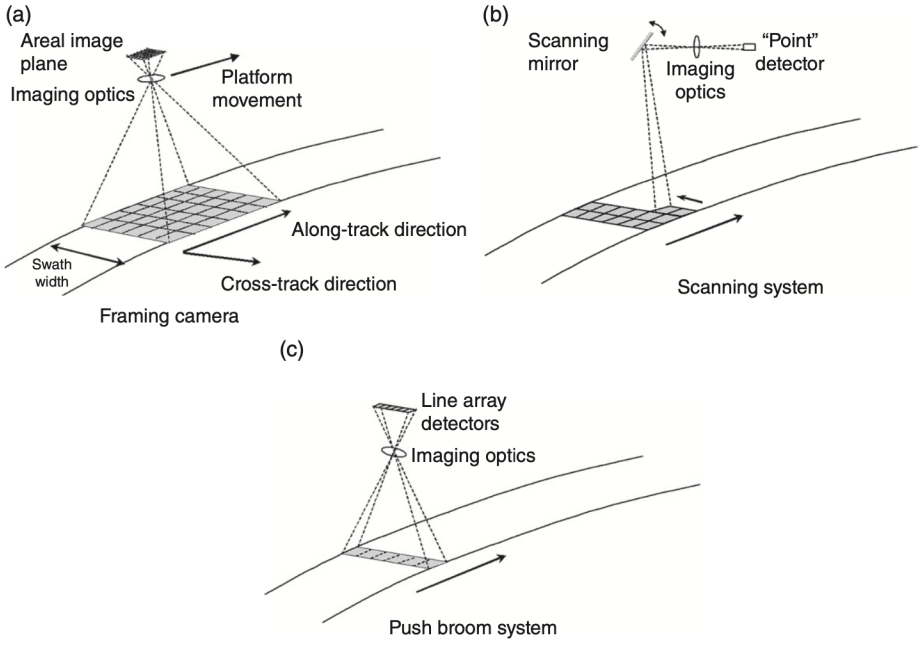

Figure 3.36 Different types of imaging sensor implementations.

Figure 2: Imaging systems: b) shows a scanning system and c) shows a pushbroom system

4. Not all energy is reflected directly from the Earth. Why is that? And what are the two most important material properties that regulate this? And how do they vary?

Of the energy that reaches the Earth, not all of it is reflected directly back. When radiation hits a smooth surface some of the radiation is transmitted into the material and some of the radiation is reflected back. How much of the incoming energy that will be reflected off of a surface is called the reflectivity. The reflectivity differs for different materials and is also dependent on the properties of the electromagnetic radiation itself. The reflectivity (and thus also the emissivity) of the material is dependent on the material's complex dielectric constant. Materials with high electric permittivity tend to have higher absorption and lower reflectivity,

such as water.

Another important factor when looking at the incoming electromagnetic radiation is the structure of the surface it hits. A rough surface will scatter energy an multiple directions, meaning that more of the energy will not reflect back out. In fact, this property is often used in active remote sensing, as we are looking for the back-scatter of electromagnetic waves. The roughness of a surface will be characterized relative to the wavelength of the electromagnetic radiation. A surface is considered smooth if the wavelength of the electromagnetic wave is large compared to the variations in the surface Johansson ((2025a)). In figure 3 a sketch shows the behavior of electromagnetic radiation as it hits a smooth and a rough surface.

Figure 3: Reflection and transmission of EM waves

0.2 Task 2: Practical Surface Sensing – optical remote sensing

In this task the goal is to identify which of a farmers fields have the most need of fertilizer.

a) We made a python program to plot the spectral profile of bare soil, green grass, and the atmosphere from the data files that have been provided us. The data files we have used to plot are jhu.becknic.vegetation.grass.green.solid.gras.spectrum.txt for grass, jhu.becknic.soil.alfisol.paleustalf.coarse.87P2410.spectrum.txt for soil and ASTMG173.csv for the atmosphere. The plotted spectral profiles of green grass, bare soil and atmosphere can be seen in figures 4, 5 and 6. To get the same units for all three plots y-axis we wanted to use the information provided in ASTMG173.csv to give us a normalized transmission profile for the atmosphere. We then had to divide the extraterrestrial irradiance by the global tilt irradiance and multiply by 100.

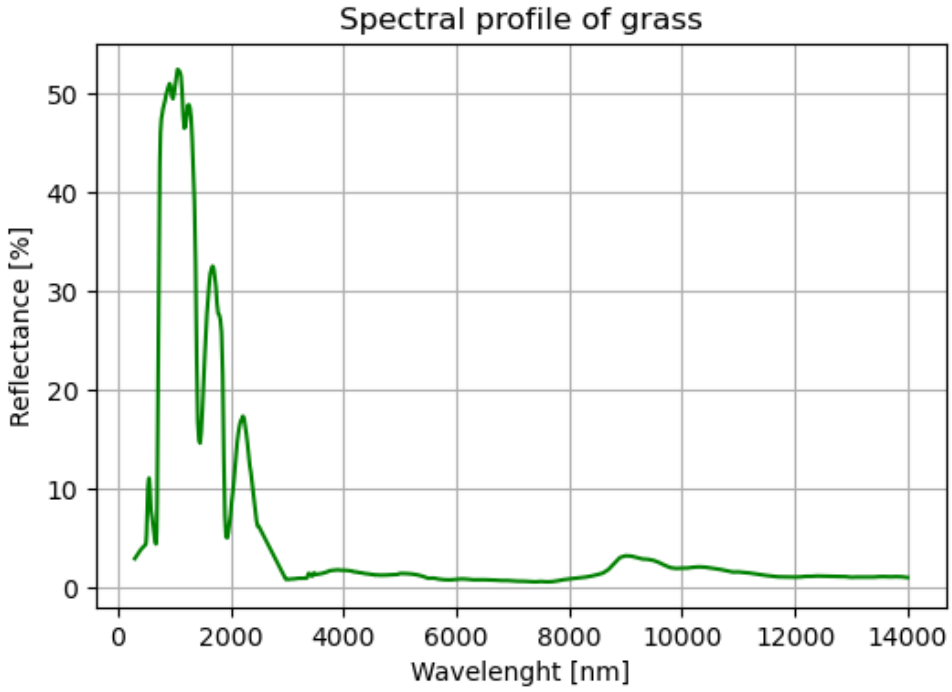

Figure 4: Plotted spectral profile of green grass

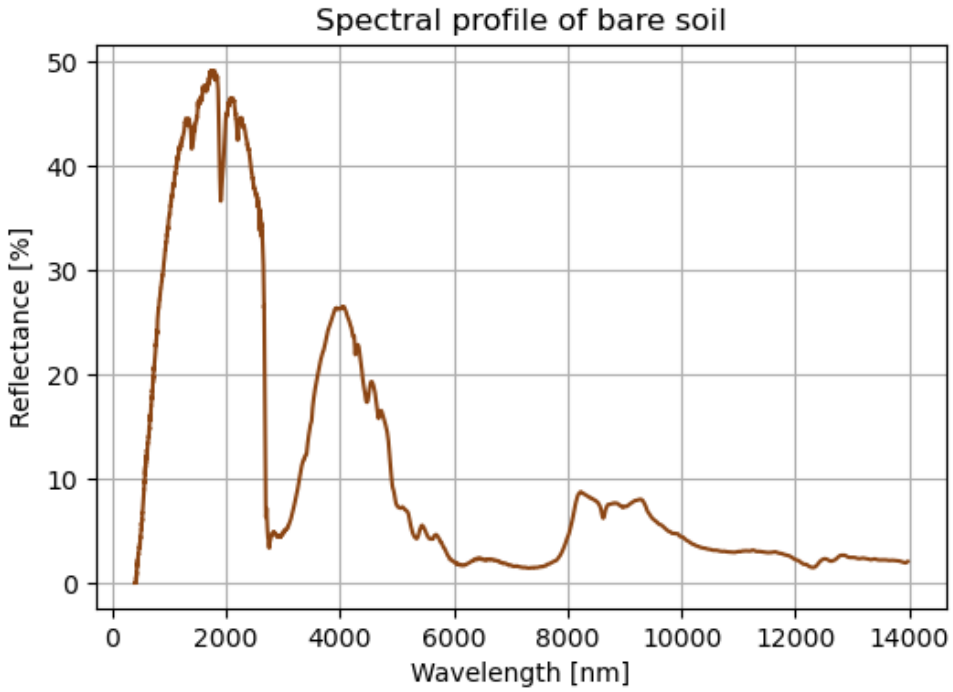

Figure 5: Plotted spectral profile of green grass

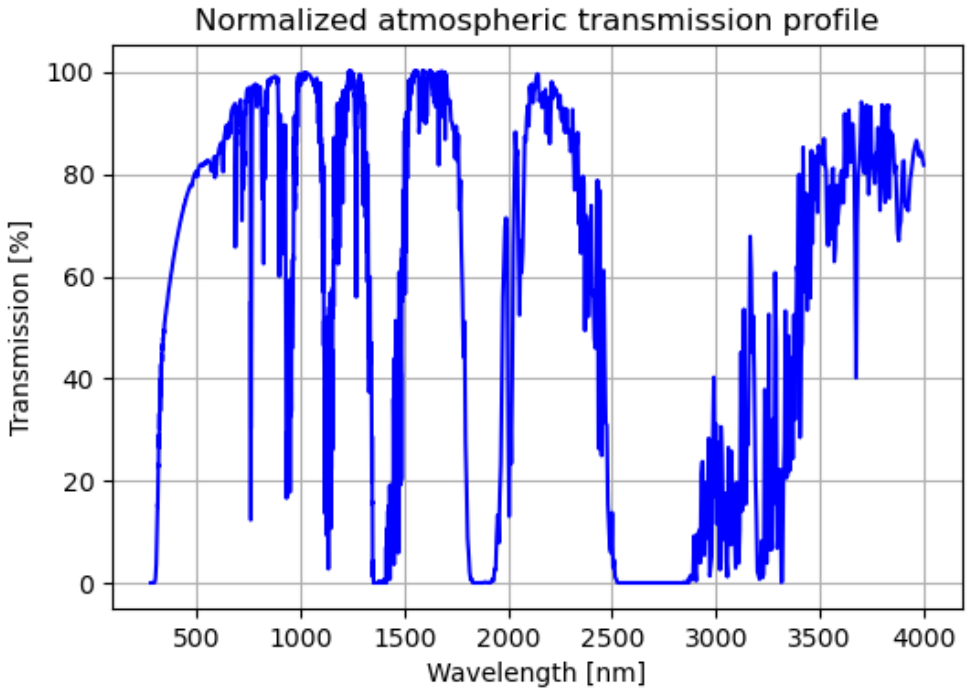

Figure 6: Plotted spectral profile of the atmosphere

b) In this task we wanted to see the different spectral profiles as they would be seen by the

13 spectral bands of Sentinel-2A. To do this we found the bands of the satellite by using the bandwidths and central wavelengths of the 13 bands and then we averaged the reflectance values over each spectral band.

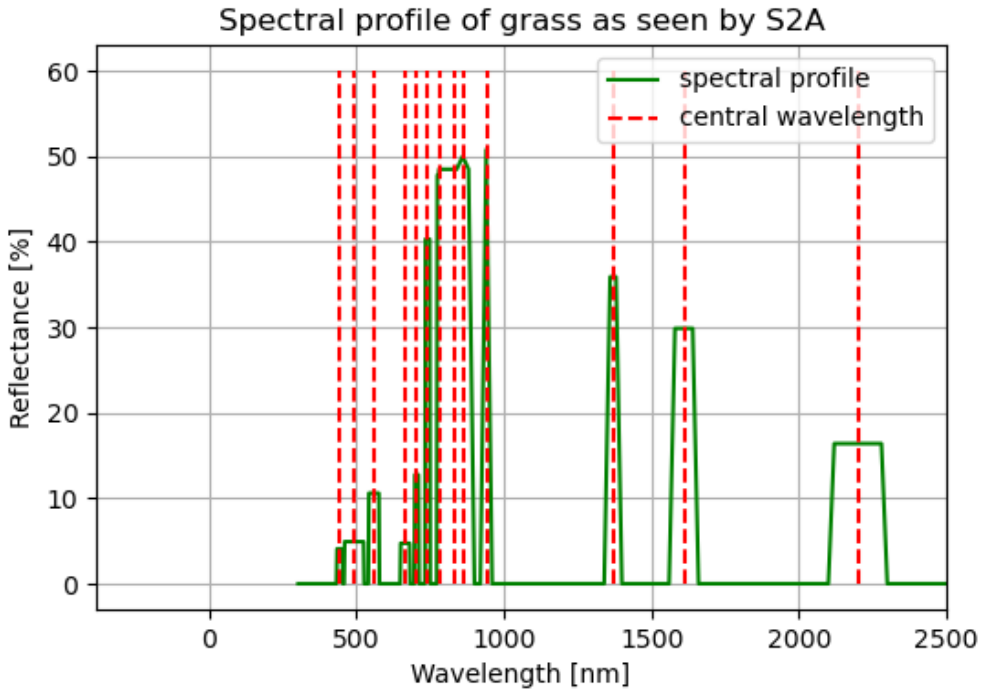

Figure 7: Reflectance profile of grass averaged over S2A's 13 spectral bands

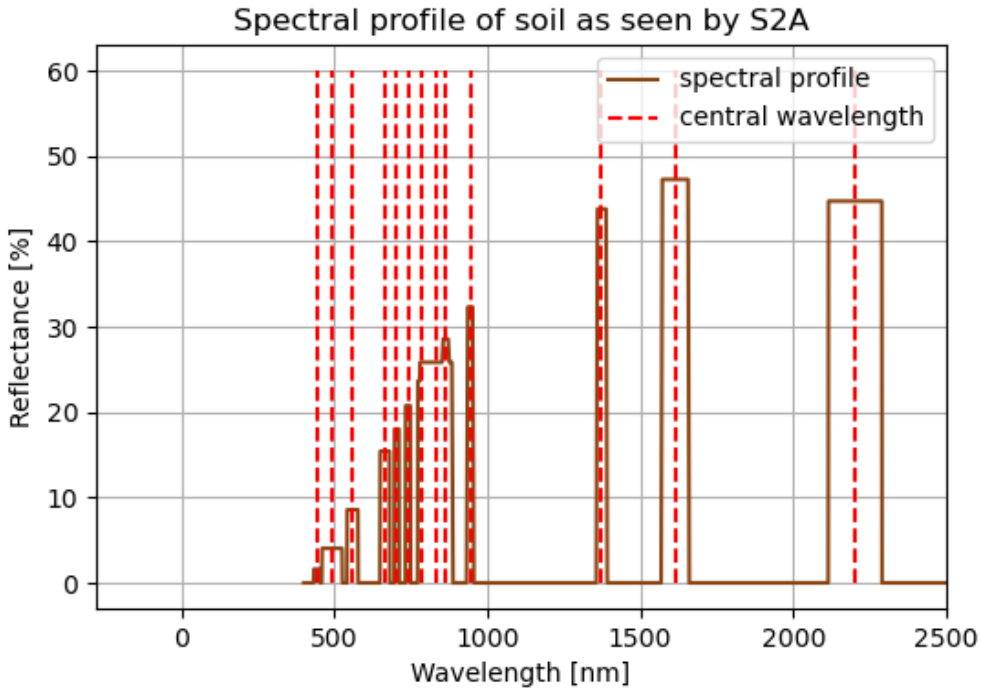

Figure 8: Reflectance profile of soil averaged over S2A's 13 spectral bands

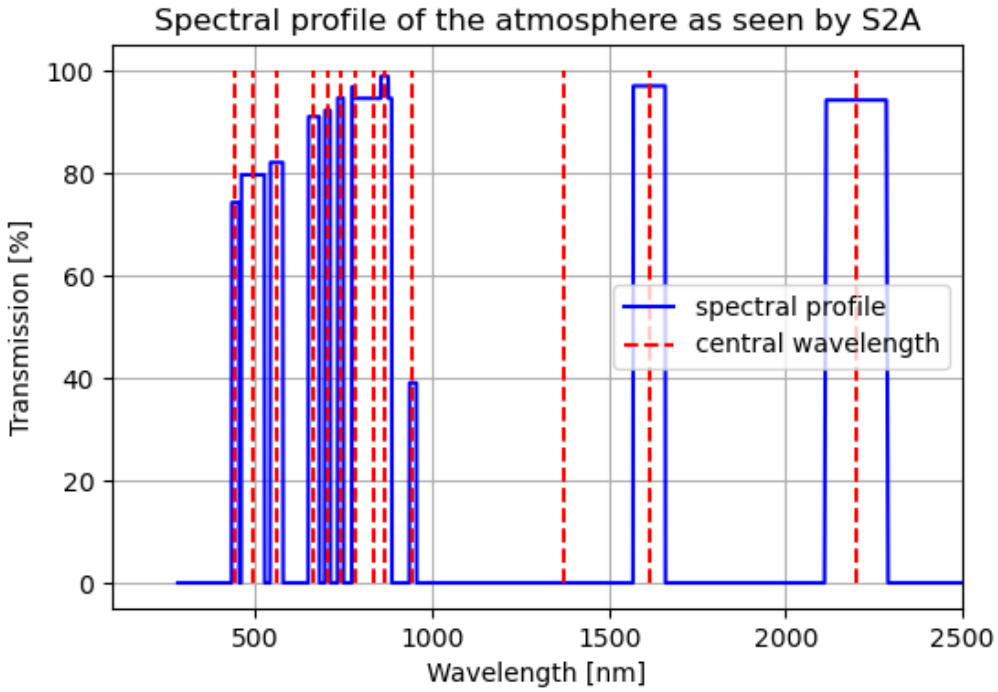

Figure 9: Transmission profile of the atmosphere averaged over S2A's 13 spectral bands

c) In figure 6, we see that we have several dips. These are caused by the different chemicals that

are in the composition of the atmosphere, Different chemical compositions will absorb electromagnetic radiation at different wavelengths. In our atmosphere water vapor (H2O) and carbon dioxide (CO2) are the main molecules responsible for the strong absorption bands. Elachi and Van Zyl ((2021)) On a quantum level this absorption is caused because different particles can only be at discrete energy states, and thus we will get absorption only on wavelengths that are equivalent to these energies. In figure 9 we see that two "dips" corresponding to bands of Sentinel-2A. These are for band 9 (water vapor) and band 10 (SWIR - cirrus). From the naming of these to bands we can extrapolate what they are used for. As mentioned, water vapor is responsible for much of the absorption of solar radiation in the atmosphere. SWIR (short wave infrared) - cirrus indicated that band 10 has something to do with clouds as cirrus is a type of cloud. Dowling and Radke ((1990)). From this we can assume that these bands are not for ground detection, but rather for atmospheric detection or correction.

d) In this task we wanted to only use the bands of Sentinel-2A that have a special resolution of 10 meters. Because of this we have to filter out the 20 and 60 meter bands. We then averaged the reflectances of grass and of bare soil over these bands the same way we did in task b). We plotted these in the same plot as seen in figure 10.

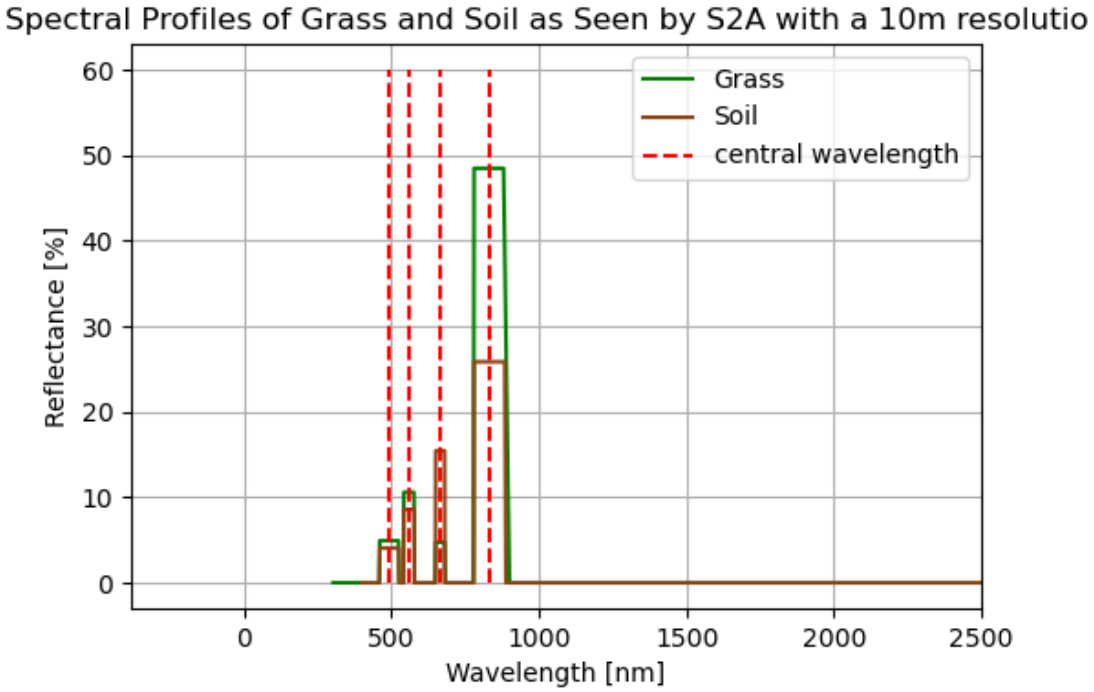

Figure 10: Spectral profiles of bare soil and grass as seen by the 10 meters bands of the Sentinel-2A satellite

We see that the bands give different responses for soil and dirt. The bands that show the most pronounced difference between soil and dirt are B4 (Vegetation red edge) and B8 (Near infrared). These bands could then be used as the most useful in this case.

e) In this task we wanted to use linear mixing to find the fraction of grass for different fields. We used equation 2 over the different 10 meter resolution bands for the Sentinel-2A satellite.

$$R_{\text{observed}}(\lambda) = f_{\text{grass}} \cdot R_{\text{grass}}(\lambda) + (1 - f_{\text{grass}}) \cdot R_{\text{soil}}(\lambda) \tag{1}$$

$$f_{\text{grass}} = \frac{R_{\text{observed}}(\lambda) - R_{\text{soil}}(\lambda)}{R_{\text{grass}}(\lambda) - R_{\text{soil}}(\lambda)} \tag{2}$$

From what we discussed in task d) the most recommended bands to use for the linear mixing will be B4 and B8.

f) Here we used the file TestFields.txt and the function we wrote for task e) to calculate the grass fraction for each of the five fields. We calculated both the average values for each band and the mean over all bands. We did one calculation over all four of the 10 meter resolution bands and one using B4 and B8 since these two are the most useful. The mean values over all four bands are plotted in figure 11 and the mean value using only the two bands are plotted in figure 12.

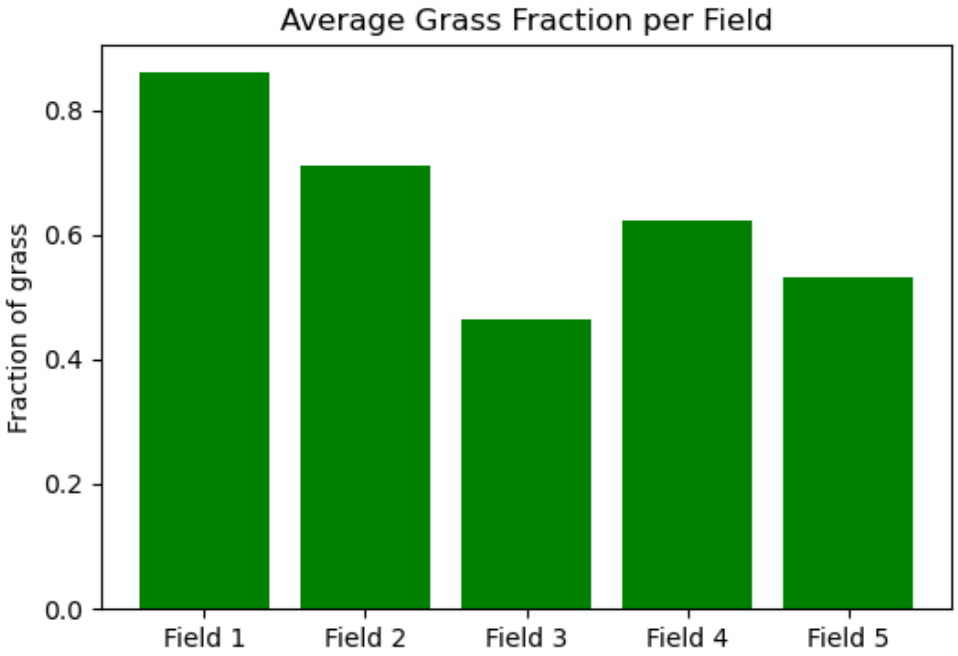

Figure 11: Average estimated fraction of grass for each field using B2, B3, B4 and B8

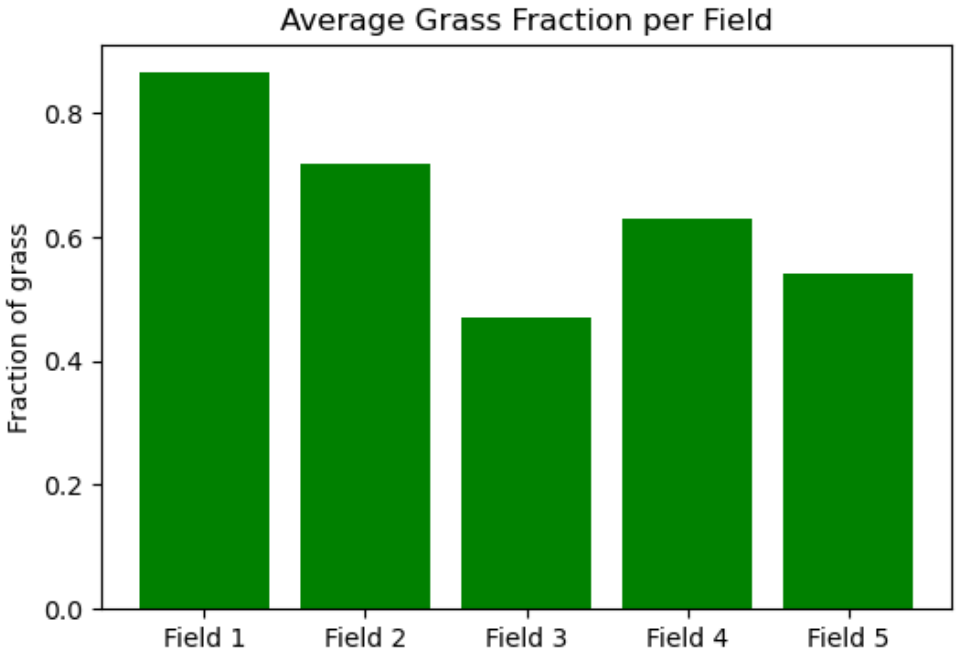

Figure 12: Average estimated fraction of grass for each field using B4 and B8

We see that there is not that large of a difference between the two plots. For both calculations we see that the fields that has the smallest fraction of grass (the fields that has the most need of fertilizer) is field 3. All code used in this task can be seen in the appendix A.1.

0.3 Task 3: Practical Surface Sensing – passive microwave

a) AMSR-2 (Advanced Microwave Scanning Radiometer 2)is a passive remote sensing instrument used to measure microwave emissions from the Earth's surface and atmosphere. We know that the microwave region is useful for surface remote sensing as the atmosphere is transparent or partially transparent to wavelengths in the microwave spectrum. Since microwaves are emitted by the Earth and not reflected we can use microwave sensors both during day and night which is especially important for sea ice sensing in the polar regions as they experience the polar night, and visible or NIR sensors will be "blind" during these periods. Continuous mapping of the sea ice is important to be able to see the movement and structural changes of the ice over the seasons Spreen and Kern ((2017)). The AMSR-2 sensors large swath width makes it possible to map the global sea ice over no more than a 2-day span, giving us a good data set for watching the ices temporal variation NOAA Office of Satellite and Product Operations ((n.d.)).

The difference in brightness temperature of sea ice compared to open water give us a way to differentiate the to and find the extent of sea ice. The open water, which has a much higher reflectivity than the sea ice, will look cooler that the sea ice to microwave sensors. This makes it possible to map the sea ice concentration. We also know that the ice itself changes over time, and first-year ice has a different composition and structure than multi-year ice, for example the multi-year ice usually has lower salinity than the first-year ice Johansson ((2025b)). These changes also cause changes in the observed brightness temperature which makes mapping of the different ices possible with microwave remote sensing.

There are some considerations to make when doing sea ice sensing in the microwave spectrum, regarding polarization and frequency. Using higher frequency bands of the sensor give a higher spatial resolution making it possible to detect smaller scale changes. However, there is a trade-off when choosing these higher frequencies as they are more prone to atmospheric influence, and areas affected by clouds can be confused for sea ice. At lower frequencies the atmospheric influence is much lower but we also get a much lower spatial resolution Spreen and Kern ((2017)). The frequency also affects the penetration depth, so sensing a higher frequency will limit us to the surface, while a lower frequency can give more information about the ice layers as it has a larger penetration depth Johansson ((2025b)). The polarization used will also have an effect on emissivity, and the contrast between vertical and horizontal linear polarization can be applied for seeing the difference between different types of ices and water. Water usually has a much higher emissivity change between polarizations that the ice Spreen and Kern ((2017)).

b) In this task we are asked to plot the brightness temperature over the Svalbard area using three different frequencies: 18.7 GHz, 36.5 GHz and 89.0 GHz. The data we use contains data for 19 GHz, 37 GHz and 85. GHz so we will assume the two first are correct and that the last one should be 85.9 GHz. To only plot over the Svalbard region we filtered out the latitudes between 75 and 85 degrees north and the longitudes between 10 degrees west and 50 degrees east. The plotted brightness temperatures are shown in figure 13.

c) In this task we want to find the sea ice concentration (SIC) of the area around Svaldbard that we looked at in task b). We will start by identifying the areas containing sea ice. To help us with this we will use figure 14.

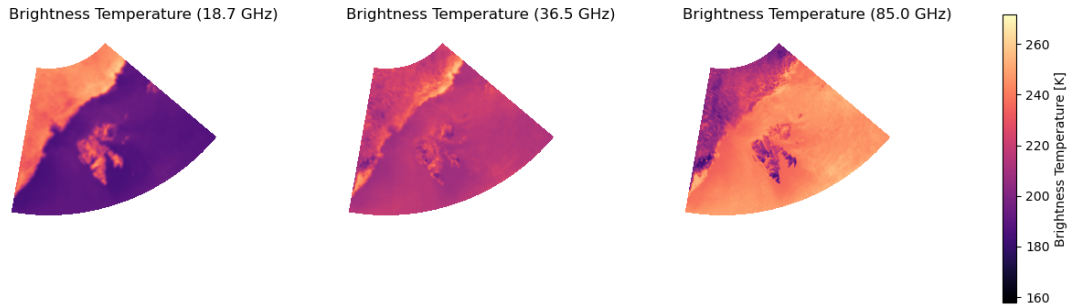

Figure 13: Brightness temperature over Svalbard using three different frequencies: 18.7 GHz, 36.5 GHz, 85 GHz

Observations of vertically (V) and horizontally (H) polarized emissivity's ε of sea ice and sea water at an incident angle of 50° from two field campaigns. For frequencies typically used for sea ice concentration retrieval (19, 37, 89 GHz) polarization difference for water, D, is larger than for all sea ice types (A, B, C). You can ignore the late summer ice type.

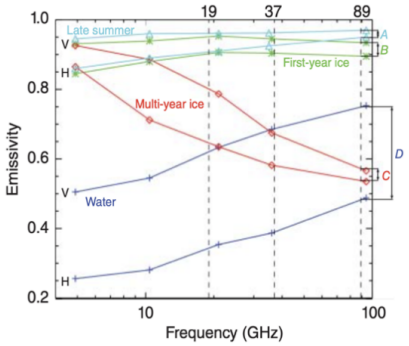

Figure 14: Emissivity for different types of ice and water

We will use the information from the plot in figure 14 regarding first-year ice, multi-year ice and water. We assume that the surface temperature is constant over the whole area which means that the only quantity affecting the brightness temperature is the emissivity. We are starting with finding the open water regions. From the figure we see that water has low emissivity at the lower frequencies and the emissivity grows larges as the frequency increases. Because of this we expect the open-water areas to be darker than ice at 18.7 GHz. We are using the vertical polarization and we see that at 89 GHz the emissivity of water for vertical polarization should be higher than for multi-year ice. We then expect the water to be a brighter area in figure 13.

We will calculate a gradient ratio for this assumed open-water area using equation 3.

$$GR_{v1,v2} = \frac{T_B(v1) - T_B(v2)}{T_B(v1) + T_B(v2)} \tag{3}$$

Where T_B is the brightness temperature measured at two different vertically polarized frequencies. We will find the values in our plot, figure 13, and we will compare the 18.7 GHz and 36.5 GHz frequencies. For the 18.7 GHz measurement we will use a brightness temperature of $T_B = 200$ K and for the 36.5 GHz we will use a brightness temperature of $T_B = 230$ K. Inserting this in equation 3 we get a gradient ratio of $GR = 0.07$, which is a percentage of 7 %. For sea ice we would expect lower values, around zero, or negative for multi-year sea ice. We see that first-year ice has high emissivity for all of the frequencies we are using. Thus we expect areas containing first-year ice to be bright in all three plots in figure 13. We see that we have some small areas this seem to apply to around the edge of the large area on the top-left

of the plots.

For the multi-year ice we expect it to be bright in the 18.7 Ghz plot and much darker in the 89.0 GHz plot. We see that the island (Svalbard) and the large mass in the top left of the plots meet this requirement, but Svalbard is a landmass and is not sea ice. We can thus conclude that the large area on the top-left contains mostly multi-year sea-ice with small areas of first-year sea-ice as mentioned. There also seems to be some first-year ice around Svalbard.

The GR assumes that a sea ice concentration between 0-15 % is open water. There is good reason to use a range for this. When we are doing remote sensing we are often using systems that have a large spatial resolution. This means that we can get some overlap of detection in regions where we have sea ice and water. In addition to this we do not always have an abrupt change between sea ice areas and open-water areas. Saying that SIC between 0 and 15 % is open water can help us with better mapping of the water and ice areas.

To make a SIC (sea ice concentration) map we are using a two-type mixing model. We can use equation 4 to calculate the sea ice concentration C. This equation, from Markus and Cavalieri ((2012)), is derived by assuming the brightness temperature is a linear mixture of the brightness temperature of water and of ice.

$$C_i = \frac{T_b - T_o}{T_i - T_o} \tag{4}$$

In this equation T_B is the brightness temperature, T_o is the brightness temperature of 100 percent water and T_i is the brightness temperature of 100 percent ice. To find values for T_i and T_o we use the plot in figure 14 and find some values for the emissivity at the correct frequencies and polarizations. We assume that most of the ice will be multi-year ice so we will use that for the calculations now. We use a surface temperature of 273 K for all calculations. We will use the 18.7 GHz frequency. This is because it has a high contrast between water emissivity and ice emissivity. Here we also will have a higher emissivity for ice than for water.

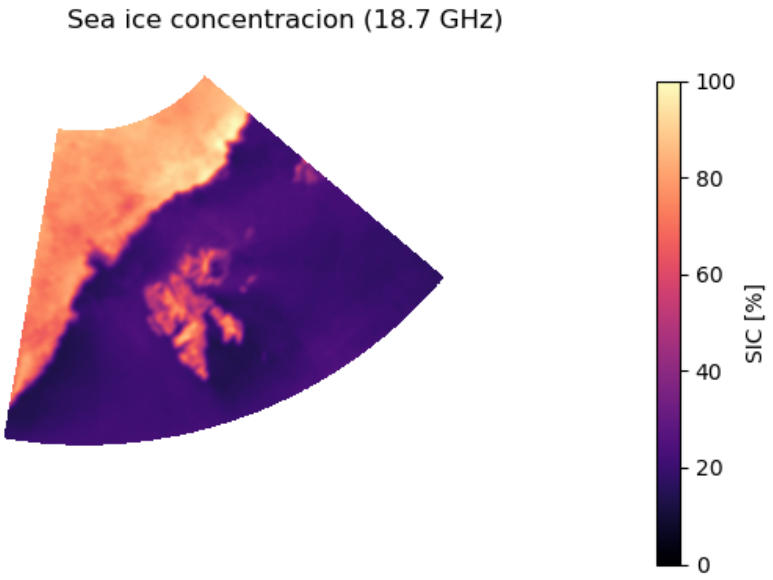

Figure 15: Sea ice concentration in the Svalbard region

We see that the sea ice concentration is high in the top-left area. This is also the area we predicted to be sea ice earlier when looking at the brightness temperature. The whole area is light colored, indicating a SIC of 70 percent and above. The dark purple area is open water, while the medium colored blob in the middle, that could look like some higher concentration of sea ice, is Svalbard, and is therfore not considered sea ice.

d) In this task we want to use the gradient ratio to detect the different types of ice (multi- or first-year ice). Using values for emissivity from figure 14 we calculate some expected values for the gradient temperature of these two different ices. For first year ice we get values of around -0.005 which is close to zero. This can be expected from looking at 14 where we see that the line for FYI is not varying greatly which means that the brightness temperature also will not vary much, giving a small ratio. For the gradient value of MYI we get more negative values which is expected by the steeper decline of the emissivity as the frequency increases. For water we would then expect values larger than zero for the gradient ratio by the same type of arguments. We calculated the gradient ratio for the observed brightness temperature over the Svalbard region and plotted, shown in figure 16.

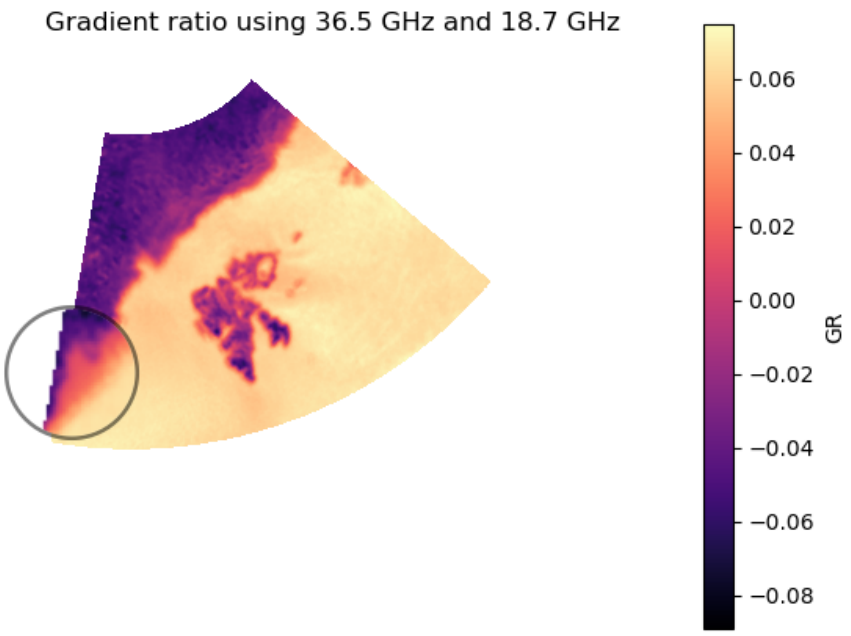

Figure 16: Gradient ratio over Svalbard area

We see that we get a red/pink line around the large area at the top left if the image, which is also what we have zoomed into. These are areas that have gradient ratio close to zero. This will be the newer ice because of the reasons stated above. In figure 13 wo noted that the same area was bright for all frequencies when plotting brightness temperature, which is also a strong indicator of new ice.

e) To map the areas where we see open water, first-year ice and multi-year ice we can use the gradient ratios. As discussed in task d) we have some expectations for the different values we can use to identify the different surfaces. We will use the following limits:

- MYI: GR < -0.02
- FYI: $-0.2 <$GR < 0.02
- Water: GR ¿ 0.02

We also use the provided code example for help so Svalbard does not look like sea ice anymore. In figure 17 we see that the ice looks as we earlier predicted, with a line of new ice at the border between open water and multi-year ice.

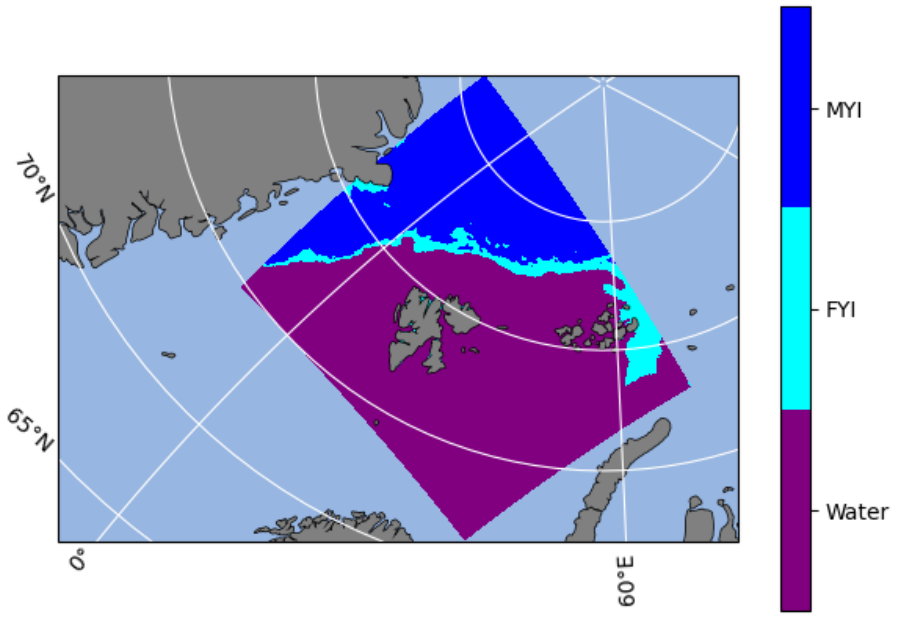

Figure 17: Map of sea ice and water using gradient ratio

All code used for this task can be seen in the appendix A.2.

This is the report for the home exam in passive remote sensing for the FYS:3001 course at UIT. Artificial intelligence has not been used to write this report. In the appendix the code used to solve the different tasks can be found. The code is written by me, but AI has been used to de-bug and to help with things like plotting. For task 3 I have used the example code provided as inspiration for the landmask.

References

N. Borotkanych. Spatial resolution in remote sensing: Which is enough?, March 28, 2025. URL https://eos.com/blog/spatial-resolution/.

Copernicus SentiWiki. S2 mission, n.d. URL https://sentiwiki.copernicus.eu/web/s2-mission#S2Mission-OverviewofSentinel-2Mission.

D. R. Dowling and L. F. Radke. A summary of the physical properties of cirrus clouds. *Journal of Applied Meteorology and Climatology*, 29(9):970–978, 1990.

C. Elachi and J. J. Van Zyl. *Introduction to the physics and techniques of remote sensing*. John Wiley & Sons, 2021.

M. Johansson. Spectra from gases and solids. transmission, reflection, absorption and scattering mechanisms, 2025a.

M. Johansson. Passive microwave sensing, 2025b.

T. Markus and D. J. Cavalieri. Algorithm theoretical basis document sea ice products, 2012. URL https://nsidc.org/sites/default/files/amsr-atbd-supp12-seaice.pdf.

NOAA Office of Satellite and Product Operations. About amsr-2, n.d. URL https://www.ospo.noaa.gov/products/atmosphere/gpds/about_amsr2.html.

G. Spreen and S. Kern. *Methods of Satellite Remote Sensing of Sea Ice*, pages 231–258. John Wiley & Sons, Ltd, 3rd edition, 2017.

A Appendix

A.1 Code for task 2

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt

def data_reader(file_name: str, column_names: list, delimiter_in:
    str = None) ->  tuple[pd.DataFrame, np.ndarray]:
    '''Function that takes in the name a data file and the names
       of the columns and returns
    the data as a pandas dataframe

    Parameters:
        file_name (str): Name of the file that will be read
        column_names (list): List containg the names wanted for
            the columns of data
        delimiter_in (str): Delimiter for the file, such as comma
             or whitespace, if no argument is passed, whitespace
            is assumed

    Returns:
        data (pd.DataFrame): Pandas dataframe with the data from
            the file
        data2 (np.ndarray): Numpy array containing the data from
            the file
```

```
    '''

    #Finding the first line starting with a number to find the
        data we are interested in
    with open(file_name, 'r') as f:
        lines = f.readlines()
        for i, line in enumerate(lines):
            if line.strip() and line.strip()[0].isdigit():
                skip_rows = i
                break

    #Reading the data in the file given as an argument
    if delimiter_in:
        data = pd.read_csv(file_name, comment = '%', header =
            None, delimiter = delimiter_in,
                           names = column_names, skiprows =
                               skip_rows)
    else:
        data = pd.read_csv(file_name, comment = '%', header =
            None, delim_whitespace = True,
                           names = column_names, skiprows =
                               skip_rows)

    data2 = data.to_numpy()

    return data, data2

def spectral_profile_sat(spectral_profile: np.ndarray,
    satellite_band_central_wavelength: np.ndarray,
                         sat_bandwidth: np.ndarray)-> np.ndarray:

    '''
    A function that takes in a spectral profile and band
        specifications of a satellite and averages the
    spectral values over the satellite bands such as seen by the
        satellite

    All wavelengths should be in meters

    Parameters:
        spectral_profile (np.ndarray): 2d array containing
            wavelenghts (m) and irradiance
        satellite_band_central_wavelength (np.ndarray): Array
            containg the central wavelengths (m) of the satellites
             bands
        sat_bandwidth (np.ndarray): Array containg the satellites
             bandwidths (m), has the same length as the array
             containg central wavelengths

    Returns:
        band_averages (np.ndarray): An array of the same length
            as the spectral values containing the
             values avaraged over the satellites bands
        avg_band_values (np.ndarray): AN array of the same length
```

```
            as the number of spectral bands
            containing the average value over each band
    '''

    #Defining the bands
    i = 0
    bands = np.zeros((len(satellite_band_central_wavelength), 2))

    #Finding the bamds of the satellite by +- hakf the bandwidth
       from each central wavelength
    for central_freq in satellite_band_central_wavelength:
        band_min = central_freq - (sat_bandwidth[i] / 2)
        band_max = central_freq + (sat_bandwidth[i] / 2)
        bands[i] = [band_min, band_max]
        i += 1

    wavelengths = spectral_profile[0]
    values = spectral_profile[1]
    band_averages = np.zeros(len(values))
    avg_band_values = np.zeros(len(
       satellite_band_central_wavelength))
    i = 0

    #Finding the numbers inside the band using masking
    for band in bands:
        mask = (wavelengths >= band[0]) & (wavelengths <= band
           [1])

        if np.any(mask):
            band_averages[mask] = np.mean(values[mask])
            avg_band_values[i] = np.mean(values[mask])
        i += 1

    return (band_averages, avg_band_values)

def linear_mixing(R_observed: np.ndarray, R_soil: np.ndarray,
   R_grass: np.ndarray
                )-> tuple[np.ndarray, float]:

    '''
    Using a spectral mixing model to estimate the fraction of
       grass

    Parameters:
        R_obs (np.ndarray): Observed reflectance (13 bands)
        R_grass (np.ndarray): Grass reflectance (13 bands)
        R_soil (np.ndarray): Soil reflectance (13 bands)

    Returns:
        f_grass (np.ndarray): fraction of grass
        f_grass_mean (float): Mean grass fraction across bands
    '''

    fraction_grass = (R_observed - R_soil) / (R_grass - R_soil)
```

```
    fraction_grass_avg = np.mean(fraction_grass)


    return fraction_grass, fraction_grass_avg

if __name__ == '__main__':

    '''---- Task 2 a) ----'''

    #Defining column names and reading the data from the three
        files containing the data
    grass_col_names = ['wavelength [micrometer]', 'reflectance [
        percent]']
    spectral_profile_grass = data_reader(file_name = 'jhu.becknic
        .vegetation.grass.green.solid.gras.spectrum.txt',
                                         column_names =
                                            grass_col_names)[1]

    soil_col_names = ['wavelength [micrometer]', 'reflectance [
        percent]']
    spectral_profile_soil = data_reader(file_name = 'jhu.becknic.
        soil.alfisol.paleustalf.coarse.87P2410.spectrum.txt',
                                         column_names =
                                            soil_col_names)[1]

    atmos_col_names = ['wavelength [nanometer]', 'etr [watts per
        square meter per nm]',
                       'global tilt [watts per square meter per
                          nm]', 'direct circumsolar [watts per
                          square meter per nm]']
    spectral_profile_atmosphere = data_reader(file_name = '
        ASTMG173.csv',column_names = atmos_col_names,
                                              delimiter_in = ',')
                                                 [1]

    #Plotting all wavelengths in nanometers
    #Grass plotting
    fig1, ax1 = plt.subplots(figsize = (6, 4))
    ax1.plot(spectral_profile_grass[:, 0] * 1e3,
        spectral_profile_grass[:, 1], color = 'green')
    ax1.set_xlabel('Wavelenght [nm]')
    ax1.set_ylabel('Reflectance [%]')
    ax1.set_title('Spectral profile of grass')
    ax1.grid(True)
    plt.savefig('spectral_profile_grass.png')
    plt.show()

    #Soil plotting
    fig2, ax2 = plt.subplots(figsize = (6, 4))
    ax2.plot(spectral_profile_soil[:, 0] *1e3,
        spectral_profile_soil[:, 1], color = 'saddlebrown')
    ax2.set_xlabel('Wavelength [nm]')
    ax2.set_ylabel('Reflectance [%]')
    ax2.set_title('Spectral profile of bare soil')
```

```
ax2.grid(True)
plt.savefig('spectral_profile_soil.png')
plt.show()

#Atmpspheric plotting
#We have divided irradiance at ground level by irradiance at
    above atmosphere to
#get a percentage that is transmitted through the atmosphere
fig3, ax3 = plt.subplots(figsize = (6,4))
ax3.plot(spectral_profile_atmosphere[:, 0],
         100*spectral_profile_atmosphere[:, 2]/
            spectral_profile_atmosphere[:, 1], color='blue')
ax3.set_xlabel('Wavelength [nm]')
ax3.set_ylabel('Transmission [%]')
ax3.set_title('Normalized atmospheric transmission profile')
ax3.grid(True)
plt.savefig('spectral_profile_atmosphere2.png')
plt.show()

# #Atmospheric plotting, plotting the three different
    irradiances
# fig3, ax3 = plt.subplots(3, 1, figsize=(7, 10), sharex=True
    )

# #Extraterrestrial irradiance
# ax3[0].plot(spectral_profile_atmosphere[:, 0],
    spectral_profile_atmosphere[:, 1], color='cyan')
# ax3[0].set_ylabel(r'Etr [W m$^{-2}$ nm$^{-1}$]')
# ax3[0].set_title('Extraterrestrial Irradiance')
# ax3[0].grid(True)

# #Global tilt
# ax3[1].plot(spectral_profile_atmosphere[:, 0],
    spectral_profile_atmosphere[:, 2], color='blue')
# ax3[1].set_ylabel(r'Global Tilt [W m$^{-2}$ nm$^{-1}$]')
# ax3[1].set_title('Global Tilt Irradiance')
# ax3[1].grid(True)

# #Direct + circumsolar
# ax3[2].plot(spectral_profile_atmosphere[:, 0],
    spectral_profile_atmosphere[:, 3], color='purple')
# ax3[2].set_ylabel(r'Direct + Circumsolar [W m$^{-2}$ nm$
    ^{-1}$]')
# ax3[2].set_title('Direct + Circumsolar Irradiance')
# ax3[2].set_xlabel('Wavelength [nm]')
# ax3[2].grid(True)
# plt.tight_layout()
# plt.savefig('spectral_profile_atmosphere.png')
# plt.show()

'''---- Task 2 b) ----'''

#Reading the data file containing information about the S2A
    spectral bands
```

```
s2_column_names = ['Band', 'Name', 'Central Wavelength (nm)',
    'Bandwidth (nm)', 'Spatial Resolution (m)']
s2a_spectrals = data_reader('s2spec.csv', s2_column_names, ',
    ')[1]
s2a_central_wl = s2a_spectrals[:, 2] / 1e9 #Converted to SI
    unit meters
s2a_bandwidths = s2a_spectrals[:, 3] / 1e9 #m

#Converting wavelengths to meters for the 3 spectral profiles
spectral_profile_grass_m = np.vstack([spectral_profile_grass
    [:, 0] / 1e6, spectral_profile_grass[:, 1]])
spectral_profile_soil_m = np.vstack([spectral_profile_soil[:,
    0] / 1e6, spectral_profile_soil[:, 1]])

atmos_profile_percent = 100*spectral_profile_atmosphere[:,
    2]/spectral_profile_atmosphere[:, 1]
spectral_profile_atmos_m = np.vstack([
    spectral_profile_atmosphere[:, 0] / 1e9,
    atmos_profile_percent])

#Plotting the values of grass as seen by the satellite,
    converting to nm for plotting
fig4, ax4 = plt.subplots(figsize = (6, 4))
ax4.plot(spectral_profile_grass[:, 0] * 1e3,
    spectral_profile_sat(spectral_profile_grass_m,
    s2a_central_wl, s2a_bandwidths)[0]
          , color = 'green', label = 'spectral profile')
ax4.vlines(s2a_central_wl * 1e9, 0, 60, colors= 'red',
    linestyles= 'dashed', label = 'central wavelength')
ax4.set_xlim(None, 2500)
ax4.set_xlabel('Wavelength [nm]')
ax4.set_ylabel('Reflectance [%]')
ax4.set_title('Spectral profile of grass as seen by S2A')
ax4.grid(True)
plt.legend()
plt.savefig('grass_s2a.png')
plt.show()

#Plotting the values of bare soil as seen by the satellite,
    converting to nm for plotting
fig5, ax5 = plt.subplots(figsize = (6, 4))
ax5.plot(spectral_profile_soil[:, 0] * 1e3,
    spectral_profile_sat(spectral_profile_soil_m,
    s2a_central_wl, s2a_bandwidths)[0]
          , color = 'saddlebrown', label = 'spectral profile')
ax5.vlines(s2a_central_wl * 1e9, 0, 60, colors= 'red',
    linestyles= 'dashed', label = 'central wavelength')
ax5.set_xlim(None, 2500)
ax5.set_xlabel('Wavelength [nm]')
ax5.set_ylabel('Reflectance [%]')
ax5.set_title('Spectral profile of soil as seen by S2A')
ax5.grid(True)
plt.legend()
plt.savefig('soil_s2a.png')
```

```
plt.show()

#Plotting the values of the atmosphere as seen by the
    satellite, converting to nm for plotting
fig6, ax6 = plt.subplots(figsize = (6, 4))
ax6.plot(spectral_profile_atmosphere[:, 0],
    spectral_profile_sat(spectral_profile_atmos_m,
    s2a_central_wl, s2a_bandwidths)[0]
          , color = 'blue', label = 'spectral profile')
ax6.vlines(s2a_central_wl * 1e9, 0, 100, colors= 'red',
    linestyles= 'dashed', label = 'central wavelength')
ax6.set_xlim(None, 2500)
ax6.set_xlabel('Wavelength [nm]')
ax6.set_ylabel('Transmission [%]')
ax6.set_title('Spectral profile of the atmosphere as seen by
    S2A')
ax6.grid(True)
plt.legend()
plt.savefig('atmosphere_s2a.png')
plt.show()

'''---- Task 2 d) ----'''
#To plot only the 10 m resolution bands we make an array
    containing only these bands, using masking
s2a_resolutions = s2a_spectrals[:, 4]
mask = (s2a_resolutions == 10)
s2a_10m_resolution = s2a_spectrals[mask]

s2a_central_wl_10m = s2a_10m_resolution[:, 2] / 1e9 #
    Converted to SI unit meters
s2a_bandwidths_10m = s2a_10m_resolution[:, 3] / 1e9 #m

fig7, ax7 = plt.subplots(figsize = (6, 4))

# Plot grass profile
ax7.plot(spectral_profile_grass[:, 0] * 1e3,
    spectral_profile_sat(spectral_profile_grass_m,
    s2a_central_wl_10m,
                                                    s2a_bandwidths_1
    )
    [0],

color

    =

    'green'

    ,

    label

    =
```

```
                                                    ,
                                                    Grass
                                                    '
                                                    )


# Plot soil profile
ax7.plot(spectral_profile_soil[:, 0] * 1e3,
    spectral_profile_sat(spectral_profile_soil_m,
                                                s2a_central_wl_
                                                    ,
                                                    s2a_bandwidt
                                                    )
                                                    [0],
                                                color
                                                    =
                                                    '
                                                    saddlebrown
                                                    '
                                                    ,
                                                    label
                                                    =
                                                    '
                                                    Soil
                                                    '
                                                    )

ax7.vlines(s2a_central_wl * 1e9, 0, 60, colors = 'red',
    linestyles = 'dashed', label = 'central wavelength')
ax7.set_xlim(None, 2500)
ax7.set_xlabel('Wavelength [nm]')
ax7.set_ylabel('Reflectance [%]')
ax7.set_title('Spectral Profiles of Grass and Soil as Seen by
     S2A with a 10m resolution')
ax7.grid(True)
ax7.legend()
plt.savefig('grass_soil_s2a_10m.png')
plt.show()


'''---- Task 2 f) ----'''

test_field_col_names = ['Name',      'B2', 'B3', 'B4', 'B8']
test_field_data = pd.read_csv('TestFields.txt', comment = '%'
    , header = None, delim_whitespace= True,
                        names = test_field_col_names, skiprows
```

```
                                = None).to_numpy()

float_data_fields_percent = 100*test_field_data[1:, 1:].
    astype(float)

avg_grass_percentage = []
for field in float_data_fields_percent:
    grass_field = linear_mixing(field,
                                spectral_profile_sat(
                                    spectral_profile_soil_m,
                                    s2a_central_wl_10m,
                                    s2a_bandwidths_10m)[1],
                                spectral_profile_sat(
                                    spectral_profile_grass_m,
                                    s2a_central_wl_10m,
                                    s2a_bandwidths_10m)[1])

    avg_grass_percentage.append(grass_field[1])
    print(grass_field[1])

field_names = [f'Field {i}' for i in range(1,6)]

fig8, ax8 = plt.subplots(figsize = (6, 4))
ax8.bar(field_names, avg_grass_percentage, color = 'green')
ax8.set_ylabel('Fraction of grass')
ax8.set_title('Average Grass Fraction per Field')
plt.savefig('grass_fraction_barplot.png')
plt.show()

avg_grass_percentage = []
for field in float_data_fields_percent[:, [2, 3]]:
    grass_field = linear_mixing(field,
                                spectral_profile_sat(
                                    spectral_profile_soil_m,
                                                    s2a_central_wl_10m
                                                        [2:],

                                                        s2a_bandwidths_10m
                                                        [2:])
                                                        [1],
                                spectral_profile_sat(
                                    spectral_profile_grass_m,
                                                    s2a_central_wl_10m
                                                        [2:],

                                                        s2a_bandwidths_10m
                                                        [2:])
                                                        [1])

    avg_grass_percentage.append(grass_field[1])
    print(grass_field[1])

fig9, ax9 = plt.subplots(figsize = (6, 4))
ax9.bar(field_names, avg_grass_percentage, color = 'green')
```

```
    ax9.set_ylabel('Fraction of grass')
    ax9.set_title('Average Grass Fraction per Field')
    plt.savefig('grass_fraction_barplot2.png')
    plt.show()
```

A.2 Code for task 3

```
import xarray as xr
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import colors
import cartopy
import cartopy.crs as ccrs


def sea_ice_extent(T_b_water, T_b_ice, T_b_observed):

    '''Function that uses a two mixing model to calculate the sea
        ice concentration.

    Parameters:
        T_b_water (float or array): Brightness temperature for
            open water.
        T_b_ice (float or array): Brightness temperature for sea
            ice.
        T_b_observed (float or array): Observed brightness
            temperature from satellite data.

    Returns:
        SIC (float or array): Sea ice concentration
    '''

    SIC = np.clip((T_b_observed - T_b_water) / (T_b_ice -
       T_b_water), 0, 1)

    return SIC

def gradient_ratio(T_b1, T_b2):

    '''Function that calculates the gradient ratio.

    Parameters:
        T_b1(float or array): Brightness temperature for the
            first frequency.
        T_b2 (float or array): Brightness temperature for the
            second frequency.


    Returns:
        grad_ratio (float or array): Sea ice concentration
    '''

    grad_ratio = (T_b1 - T_b2) / (T_b1 + T_b2)
```

```
    return grad_ratio

if __name__ == '__main__':

    '''---- Task 3 b) ----'''

    data = xr.open_dataset('FYS-3001 amsr2File_20181112.nc')

    #Finding the data i wnat to use and converting to np arrays
    # latitudes = np.array(data['lat'].values)
    # longitudes = np.array(data['lon'].values)
    # bright_t_19GHz = np.array(data['tb19v'].values)[0]
    # bright_t_37GHz = np.array(data['tb37v'].values)[0]
    latitudes = data['lat'].values
    longitudes = data['lon'].values

    bright_t_19GHz = data['tb19v'][0].values
    bright_t_37GHz = data['tb37v'][0].values
    bright_t_85GHz = data['tb85v'][0].values

    #Using masking to find the right region
    mask = (latitudes >= 75) & (latitudes <= 85) & (longitudes>=
        -10) & (longitudes <= 50)
    indices = np.argwhere(mask)

    ymin, xmin = indices.min(axis = 0)
    ymax, xmax = indices.max(axis = 0) + 1

    #Finding the coordinates and brightness temperatures at the
        range given
    svalbard_latitudes = latitudes[ymin:ymax, xmin:xmax]
    svalbard_longitudes =  longitudes[ymin:ymax, xmin:xmax]
    bright_t_svalbard_19GHz = bright_t_19GHz[ymin:ymax, xmin:xmax
        ]
    bright_t_svalbard_37GHz = bright_t_37GHz[ymin:ymax, xmin:xmax
        ]
    bright_t_svalbard_85GHz = bright_t_85GHz[ymin:ymax, xmin:xmax
        ]

    svalbard_latitudes_nan = np.where(mask, latitudes, np.nan)
    svalbard_longitudes_nan = np.where(mask, longitudes, np.nan)
    bright_t_svalbard_19GHz_nan = np.where(mask, bright_t_19GHz,
        np.nan)
    bright_t_svalbard_37GHz_nan = np.where(mask, bright_t_37GHz,
        np.nan)
    bright_t_svalbard_85GHz_nan = np.where(mask, bright_t_85GHz,
        np.nan)

    #Finding max and min latitudes and longitudes
    lon_min_geo = np.nanmin(svalbard_longitudes)
    lon_max_geo = np.nanmax(svalbard_longitudes)

    lat_min_geo = np.nanmin(svalbard_latitudes)
    lat_max_geo = np.nanmax(svalbard_latitudes)
```

```
#Finding the central latitude and longitude
central_lat = (lat_max_geo - lat_min_geo) / 2
central_long = (lon_max_geo - lon_min_geo) / 2

# define projections for the figure and the original set of
    coordinates
orig_projection = ccrs.PlateCarree()
target_projection = ccrs.Stereographic(central_longitude =
    central_long,
                                       central_latitude =
                                          central_lat)

vmin = min((np.nanmin(bright_t_19GHz), np.nanmin(
    bright_t_37GHz), np.nanmin(bright_t_85GHz)))
vmax = max((np.nanmax(bright_t_19GHz), np.nanmax(
    bright_t_37GHz), np.nanmax(bright_t_85GHz)))

#Plotting
fig, ax = plt.subplots(1, 3, figsize = (12, 4),
    constrained_layout = True)

#Plotting 18.7 GHz
im1 = ax[0].imshow(bright_t_svalbard_19GHz_nan, cmap = 'magma
    ', vmin = vmin, vmax = vmax)
ax[0].set_title('Brightness Temperature (18.7 GHz)')
ax[0].set_axis_off()

#Plotting 36.5 GHz
im2 = ax[1].imshow(bright_t_svalbard_37GHz_nan, cmap = 'magma
    ', vmin = vmin, vmax = vmax)
ax[1].set_title('Brightness Temperature (36.5 GHz)')
ax[1].set_axis_off()

#Plotting 85 GHz
im3 = ax[2].imshow(bright_t_svalbard_85GHz_nan, cmap = 'magma
    ', vmin = vmin, vmax = vmax)
ax[2].set_title('Brightness Temperature (85.0 GHz)')
ax[2].set_axis_off()

cbar = fig.colorbar(im1, shrink = 0.8, pad = 0.02)
cbar.set_label('Brightness Temperature [K]')
plt.savefig('Svalbard_imshow_19GHz_37GHz.png')
plt.show()


'''---- Task 3 c) ----'''

#Defining the emissivity for each frequency/polarization
    using the plot from the task
water_emissivity_19V = 0.63
water_emissivity_37V = 0.69
water_emissivity_85V = 0.76
```

```
FYI_emissivity_19V = 0.95
FYI_emissivity_37V = 0.93
FYI_emissivity_85V = 0.92

MYI_emissivity_19V = 0.83
MYI_emissivity_37V = 0.67
MYI_emissivity_85V = 0.60

#Defining the surface temperature / actual temperature
surface_temp = 273 #K

#Using this to calculate the brightness temperatre for ice
    and water
T_i_19V = surface_temp * FYI_emissivity_19V
T_o_19V = surface_temp * water_emissivity_19V

#Finding the SIC
sea_ice_19V = sea_ice_extent(T_o_19V, T_i_19V,
    bright_t_svalbard_19GHz_nan)

#Plottign
fig, ax = plt.subplots(figsize = (7, 5))
im = ax.imshow(100* sea_ice_19V, cmap = 'magma', vmin = 0,
    vmax = 100)
ax.set_title('Sea ice concentracion (18.7 GHz)')
ax.set_axis_off()
cbar = fig.colorbar(im, shrink = 0.8, pad = 0.02)
cbar.set_label('SIC [%]')
plt.savefig('SIC_svaldbard.png')
plt.show()

'''---- Task 3 d) ----'''

T_FYI_19V = surface_temp * FYI_emissivity_19V
T_FYI_37V = surface_temp * FYI_emissivity_37V
T_FYI_85V = surface_temp * FYI_emissivity_85V

T_i_37V = surface_temp * MYI_emissivity_37V
T_i_85V = surface_temp * MYI_emissivity_85V

calc_GR_FYI = gradient_ratio(T_FYI_37V, T_FYI_19V)
calc_GR_MYI = gradient_ratio(T_i_37V, T_i_19V)

calc_GR_FYI = gradient_ratio(T_FYI_85V, T_FYI_37V)
calc_GR_MYI = gradient_ratio(T_i_85V, T_i_37V)

# print(calc_GR_FYI)
# print(calc_GR_MYI)

observed_GR_37_19V = gradient_ratio(
    bright_t_svalbard_37GHz_nan, bright_t_svalbard_19GHz_nan)

fig, ax = plt.subplots(figsize = (7, 5))
im = ax.imshow(observed_GR_37_19V, cmap = 'magma')
```

```
ax.set_title('Gradient ratio using 36.5 GHz and 18.7 GHz')
ax.set_axis_off()
cbar = fig.colorbar(im, pad = 0.02)
cbar.set_label('GR')

plt.savefig('grad_ratio_37_19.png')
plt.show()


# '''---- Task 3 e) ----'''

#Calculating the observed GR using data with no nan values
observed_GR_37_19V = gradient_ratio(bright_t_svalbard_37GHz,
    bright_t_svalbard_19GHz)

#Maksing for the different materials
water_mask = (observed_GR_37_19V >= 0.02)
FYI_mask = (observed_GR_37_19V > -0.02) & (observed_GR_37_19V
     < 0.02)
MYI_mask = (observed_GR_37_19V <= -0.02)

GR_37_19_changed = np.full_like(observed_GR_37_19V,
    fill_value = np.nan)

#Setting the values as constant within the materials
GR_37_19_changed[water_mask] = 0
GR_37_19_changed[FYI_mask] = 1
GR_37_19_changed[MYI_mask] = 2

#Plotting, inspiration from
#https://github.com/CryosphereVirtualLab/public-notebooks/
    blob/main/S1_ice_water_classification/
    load_and_calibrate_S1_scene.ipynb
cmap = colors.ListedColormap(['purple','cyan', 'blue'])
fig2, ax2 = plt.subplots(figsize = (7, 5),  subplot_kw={'
    projection': target_projection})
ax2.set_extent([lon_min_geo-1,lon_max_geo+1, lat_min_geo,
    lat_max_geo])
ax2.add_feature(cartopy.feature.OCEAN)
ax2.add_feature(cartopy.feature.LAND, facecolor = 'gray',
    zorder = 2)
gl0 = ax2.gridlines(color='white',draw_labels=True, y_inline=
    False)
ax2.coastlines(color='black')
pc2 = ax2.pcolormesh(svalbard_longitudes, svalbard_latitudes,
     GR_37_19_changed,
                     cmap = cmap, transform=orig_projection,
                        zorder = 1)
gl0.top_labels = False
gl0.right_labels = False
cbar = plt.colorbar(pc2, cmap = cmap, ticks = [0.33, 1,
    1.66])
cbar.ax.set_yticklabels(['Water', 'FYI', 'MYI'])
```

```
plt.savefig('GR_3_color_fyi_myi_W')
plt.show()
```