



UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Physics and Technology

Two-axis tracking solar irradiance measurements at Tromsø

Jakob Holden Hansen

EOM-3901 Master's Thesis in Energy, Climate and Environment

June 2020



Abstract

The world is changing towards using more renewable, rather than fossil energy sources in order to reach the 1.5°C goal of the Paris agreement (Merchant, 2018). This thesis' main focus is to investigate the solar irradiance in Tromsø using 2-axis tracking measurements from May 2020. The reduction of global irradiance and direct irradiance is also compared against the values of the other European cities, and the diffuse and clearness index is calculated. A comparison between Skartveit & Olseth (S&O) model and measurements in Tromsø is made.

The aim of the thesis is to investigate atmospheric scattering and absorption of sunlight in Tromsø and to compare these results with measurements from other cities. The SO comparison show a low accuracy between modelled values and actual measured values for diffuse and clearness index. The global horizontal irradiance (GHI) and direct normal irradiance (DNI) is compared with the extraterrestrial irradiance. Compared to other European cities, the reduction of GHI show that Tromsø have 5% less mean reduction, meanwhile the DNI reduction have an increased mean value of 10%. Conclusively, it can be shown that Tromsø has higher global irradiance values for given solar elevation angles, when compared to the other European cities.

Contents

Abstract	I
List of Figures	II
List of Tables	IV
Abbreviations	VII
Nomenclature	VIII
1 Introduction	1
1.1 Background	1
1.2 Aim of Thesis	2
1.3 Structure of Thesis	2
2 Theoretical Background	4
2.1 Solar Energy	4
2.2 Solar Irradiance	6
2.3 Air Mass	7
2.4 Cloud Enhancement	9
2.5 Clearness and Diffuse Index	10
2.6 Modelling Solar Irradiance	10
2.6.1 Skartveit and Olseth	11
2.7 Solar Elevation Angle	15
2.8 Sun Tracker	16
2.9 Pyranometer	16
2.10 Pyrheliometer	18
2.11 Calibration Standards	18

3	Method and Instruments	20
3.1	EKO STR-22G Solar Tracker	20
3.1.1	EKO STR-22G	21
3.1.2	EKO MS-57 Pyrheliometer	21
3.1.3	EKO MS-80 Pyranometer	21
3.2	Dataset	22
3.2.1	Data Quality Control	23
4	Results and Discussion	25
4.1	Irradiance Reduction Analysis	25
4.1.1	Reduction of Global Horizontal Irradiance	27
4.1.2	Reduction of Direct Normal Irradiance	33
4.1.3	Reduction of Global Normal Irradiance	38
4.1.4	Dealing with altitude difference of Izana and Cener	39
4.2	Clearness and Diffuse Index Analysis	45
4.3	User Manual	51
4.3.1	Communication	51
4.3.2	Wi-Fi Connection	53
4.3.3	Time and Date	55
4.3.4	CR6 Programming	56
4.3.5	Public Variables	58
4.3.6	Uploading CR6 Program	60
4.3.7	Azure Storage	61
4.4	Work Distribution	63
5	Final discussion and conclusions	65
5.1	Reduction of Global Horizontal Irradiance and Direct Normal Irradiance	66
5.2	Diffuse and Clearness Index	68
5.3	Final Summary	68
5.4	Further work	69
6	Bibliography	70
7	Appendices	74
7.1	Appendix A	74
7.2	Appendix B	74

List of Figures

2.1	Atmospheric effects on a typical clear day (Bowden,2019)	5
2.2	The electromagnetic spectrum of photons (Bowden, 2019)	7
2.3	Schematic of Air Mass (Bowden, 2019)	8
2.4	Schematic of Pyranometer (Kipp & Zonen, 2015)	18
3.1	Setup of measurement system at Nordlysobservatoriet	20
4.1	Difference between cloudless and clear sky day	26
4.2	Reduction in Global Horizontal Irradiance for given angles in Tromsø, Norway	27
4.3	Reduction in Global Horizontal Irradiance for given angles in for Izana, Spain	28
4.4	Reduction in Global Horizontal Irradiance for given angles in Cener, Spain	30
4.5	Reduction in Global Horizontal Irradiance for given angles in Lindenberg, Germany	31
4.6	Reduction in Global Horizontal Irradiance for given angles in Toravere, Estonia	32
4.7	Reduction in Direct Normal Irradiance for given angles in Tromsø,Norway	33
4.8	Direct Normal Irradiance for Izana, Spain	34
4.9	Direct Normal Irradiance for Cener, Spain	35
4.10	Direct Normal Irradiance for Lindenberg, Germany	36
4.11	Direct Normal Irradiance for Toravere, Estonia	37
4.12	Reduction in Global Normal Irradiance for given angles in Tromsø,Norway	38
4.13	Compensated GHI reduction plots for Izana, Spain	41
4.14	Compensated DNI reduction plots for Izana, Spain	41

4.15	Comparison between Izana and Tromsø for compensated DNI and GHI values	42
4.16	Compensated GHI reduction plots for Cener, Spain	43
4.17	Compensated DNI reduction plots for Cener, Spain	43
4.18	Comparison between Cener and Tromsø for compensated DNI and GHI values	44
4.19	Skartveit Model of Tromsø	46
4.20	Clearness and Diffuse Index for Izana, Spain	47
4.21	Clearness and Diffuse Index for Cener, Spain	48
4.22	Clearness and Diffuse Index for Lindenberg, Germany	49
4.23	Clearness and Diffuse Index for Toravere, Estonia	50
4.24	View of start window in Device Configuration Utility where connection is established	52
4.25	change in status in the start window from "Connect" to "Disconnect"	53
4.26	SW command to turn on the terminals used for powering the router	54
4.27	Time settings in Device Configuration Utility	55
4.28	Prefixed voltage difference command in CRBasic	56
4.29	DataTable function used to create 5minute average datatable	57
4.30	USR directory in the CR6 Datalogger	58
4.31	Public Variables directory in Device Configuration Utility	59
4.32	Illustation of the data table in device configuration ulitivity	60
4.33	Running and uploading a new program to the CR6 Datalogger	61
4.34	Layout of Azure Storage Software for accessing the files from the sun tracker	62
4.35	Specific commands for downloading a file from Azure Storage	63

List of Tables

3.1	Measurement Equipment installed at Nordlysobservatoriet . . .	22
3.2	Overview of the stations for comparing the radiation measurements and indexes	23
4.1	Air mass and distance at given solar elevation angles	26
4.2	Overview of clear days found after manually checking the dataset	27
4.3	Altered atmospheric distance and scaling at all solar elevation angles for Izana	39
4.4	Altered atmospheric distance and scaling at all solar elevation angles for Cener	40

Acknowledgements

First, i would like to thank my supervisor Tobias Boström for the opportunity and guidance throughout this project. I would also like to give a huge thank to Rolf Andersen and Per Ivar Emmanuelsen at UiT for the expertise and help in the resulting website and storage of data.

Thanks to my parents, who always show their support and advised me for the better.

And finally, thanks Barista Boyz for the amazing memories created throughout these five years.

Abbreviations

AM	Air Mass
GHI	Global Horizontal Irradiance
GNI	Global Normal Irradiance
DHI	Diffuse Horizontal Irradiance
DNI	Direct Normal Irradiance
CE	Cloud Enhancement
HRA	Hour Angle
LST	Local Solar Time
UTC	Coordinated Universal Time
TC	Time Correction
EoT	Equation of Time
BSRN	Baseline Surface Radiation Network

Nomenclature

Symbol	Description	SI Unit
AM	Air Mass	-
c	Speed of light	m/s^2
E	Energy	J
h	Planck's constant	m^2kg/s
H	height above sea level	m
I_{ex}	Extraterrestrial radiation	W/m^2
I_{sc}	Solar constant	W/m^2
I_{dn}	Direct normal irradiance	W/m^2
I_{dh}	Diffuse horizontal irradiance	W/m^2
I_g	Global radiation	W/m^2
θ	zenith angle	radians
n	day of year	-
α	height of sun above the horizon	m
δ	declination angle	radians
R_E	radius of Earth	m
k	Boltzmann's constant	JK^{-1}
T_0	mean atmospheric temperature	K
m	mean mass of a molecule	kg
g	acceleration due to gravity	ms^{-2}
k_t	clearness index	-
k_d	diffuse index	-

Chapter 1

Introduction

1.1 Background

The global population rises simultaneously with the global energy demand. The need for renewable energy sources becomes even more important in order to pursue the goal of the 2015 Paris agreement to limit the increase in global temperature by 1.5°C (Merchant, 2018). Changes in all aspects of society are needed to move towards a more sustainable future. In Norway, the interest in photovoltaic solar cell installations is increasingly growing since 2015. The economic benefit from low cost per megawatt-hour, paired with the increased focus on sustainability and renewable energy is possibly some of the reasons behind this. From 2016 to 2017, the capacity of installed solar energy installations in Norway increased with 59% (Multiconsult, 2018).

The need for measurements of solar irradiance at higher latitudes is needed, in order to strengthen the knowledge on the energy yield of photovoltaic solar cell installations. UiT – The Arctic University of Norway has installed a PV system to increase the knowledge of Solar energy potential in the high north. The Institute for Physics and Technology (IFT) has installations of measurement devices on several locations in Troms county to investigate the solar potential (UiT, 2019). In order to further strengthen the research on the solar potential in the high north, a 2-axis tracking system has been installed at Tromsøya. The potential for solar energy in Tromsø is limited to factors such as snow cover and polar night during winter. However, the snow cover could also be used as an advantage to harness even more solar energy

reflected from the surface. The efficiency of photovoltaic solar cells is also inversely proportional to temperature, causing a higher efficiency in colder climates.

1.2 Aim of Thesis

The idea of the thesis was conceived after talking with Professor Tobias Boström about installing a new solar irradiance measurement system for UiT - The Arctic University of Tromsø. The idea of how to further use the measurement data received from the station was discussed. The idea of investigating the atmospheric effects such as scattering and absorption in Tromsø was suggested by Tobias Boström.

The thesis aims to analyse the irradiance received in Tromsø, and comparing it with measurements taken in middle and central Europe of different climates. This evaluation should be used as a basis for discussing whether the hypothesis that Tromsø has a comparably cleaner atmosphere than most of mainland Europe, causing less reduction in the solar irradiance, is likely or not. Another aim for the thesis was the actual installation of the measurement system at Nordlysobservatoriet, Tromsø.

1.3 Structure of Thesis

Chapter 2 provides the basic principles behind the methodology of the thesis. A description of solar energy principles, measurement devices and modelling of solar irradiance is provided.

Chapter 3 informs of the methodology used to reach the result of the thesis. It includes the individual devices and sensors used in the sun tracker system installed, as well as how the dataset was acquired and filtered.

Chapter 4 presents the results and discussions of the analysis from diffuse and clearness index comparison, and reduction comparison. It presents the comparison for each location, as well as a comparison to the Skartveit&Olseth model. Possible errors are discussed and the significance of the results. Lastly it presents a user manual for maintenance of the system.

Chapter 5 hold a final summary of the results

Chapter 2

Theoretical Background

2.1 Solar Energy

Plants, animals and humans can use solar energy directly and indirectly. This energy can be viewed as a parcel of photons with energy given by (Bowden, 2019)

$$E = \frac{hc}{\lambda} \quad (2.1)$$

Where h is Planck's constant, λ , is the photon wavelength and c is the speed of light, which commonly is set as a constant as well. The equation shows the correlation between wavelength and energy of the photons, where photons with higher wavelength correspond to a low energy photon, and vice versa (Bowden, 2019). The wavelength also determines how far into the atmosphere the photons reach, due to absorption or scattering caused by molecules. Distinct gasses such as carbon dioxide, water vapour and ozone have a high absorption of photons. Infrared light with wavelengths above 2000nm is absorbed by carbon dioxide and water vapour, while ozone absorbs ultraviolet light below 0.2nm (Bowden, 2019).

Particles in the atmosphere contribute to the reduction of solar irradiance due to scattering. Scattering occurs when a photon hits a specific particle in the atmosphere and therefore deflects from the original path. The effects of both the scattering and absorption are influenced by the distance travelled through the atmosphere, called Air Mass. The reduction in solar power is uniform across the visible spectrum when the sun is directly above the surface of the

Earth. For other angles, the photons have to travel through more molecules to reach the surface, causing an uneven reduction of solar power across the visible spectrum of light. Higher energy photons with shorter wavelengths are influenced more by the atmospheric effects than the lower energy photons and cause a higher power reduction for short-wavelength photons than high wavelength photons (Bowden, 2019).

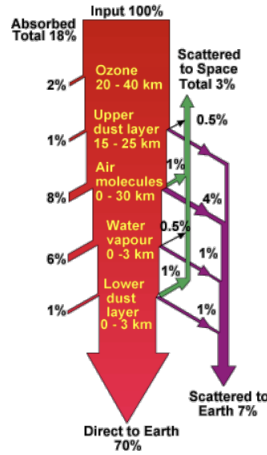


Figure 2.1: Atmospheric effects on a typical clear day (Bowden,2019)

The solar irradiance at the top of the atmosphere is called extraterrestrial irradiance I_{ex} . The extraterrestrial irradiance is dependent on the solar constant, $I_{sc} = 1367W/m^2$, the day of the year n and the height of the sun above the horizon h (Böhme, 2019).

$$I_{ex} = I_{sc} \cdot f(n) \cdot \sin(h) \quad (2.2)$$

The extraterrestrial radiation is affected by atmospheric effects such as scattering and absorption, lowering the incoming radiation at the surface of Earth. The radiation that hits the surface of Earth after being subjected to the scattering effect, causing a change in direction, is referred to as diffuse irradiance, I_{dh} . The radiation not affected by scattering, going directly toward the surface, is referred to as beam irradiance or direct irradiance, I_{dn} . The direct irradiance is measured with a 5° angle of the solar disc and therefore, in practice, the direct irradiance contains portions of diffuse irradiance

(Blanc, 2014). The annual mean diffuse irradiance reaching the surface after being affected by scattering and absorption is approximately 10%, while the direct irradiance is significantly higher at 70% (Bowden, 2019). The total irradiance also referred to as global irradiance, I_g , is the sum of the beam and the diffuse irradiance (Duffie, 2013).

$$I_g = I_{dn} + I_{dh} \quad (2.3)$$

2.2 Solar Irradiance

The amount of solar energy hitting a square meter of surface per second is called solar irradiance, measured in W/m^2 (Böhme, 2019). By measuring the solar irradiance at different wavelengths, ranging from the solar irradiance spectrum can be composed. The solar irradiance spectrum can be used to gain knowledge on which molecules absorb wavelengths in the atmosphere, and which are unaffected by the absorption phenomena. While radiation in the X-ray or ultraviolet spectrum is absorbed early by the atmosphere, the infrared and visible light spectrum has the opportunity to reach the surface. Local variations in the atmosphere due to pollution, clouds and water vapour concentration affect the solar irradiance spectrum, especially the lower energy radiation (Garner, 2008). The solar spectrum is an important tool to map the conditions of the atmosphere at a given location, and the effects on solar energy reaching the surface.

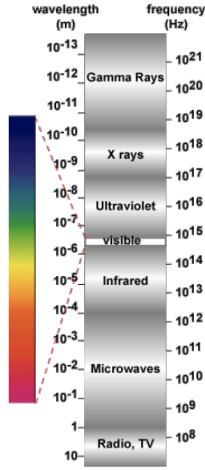


Figure 2.2: The electromagnetic spectrum of photons (Bowden, 2019)

2.3 Air Mass

The atmospheric effects such as absorption, reflection and scattering are affected by the distance the photons travel through the atmosphere. A longer path increases the number of molecules in the atmosphere the photons have to get through. Air Mass (AM) is a way to quantify this reduction of solar intensity at the surface. Air Mass is as the distance the photons have to travel through the atmosphere normalized to the shortest distance where the sun is directly over the surface (Bowden, 2019). The airmass is defined as

$$AM = \frac{1}{\cos(\theta)} \quad (2.4)$$

Where θ is the zenith angle, which is the angle between the shortest sun distance and the actual angle of incident sunlight, increasing the zenith angle θ corresponds to an increased AM. If the $AM = 1$, the sun is directly over the surface, causing minimal interaction with the atmosphere and therefore minimal reduction of solar intensity.

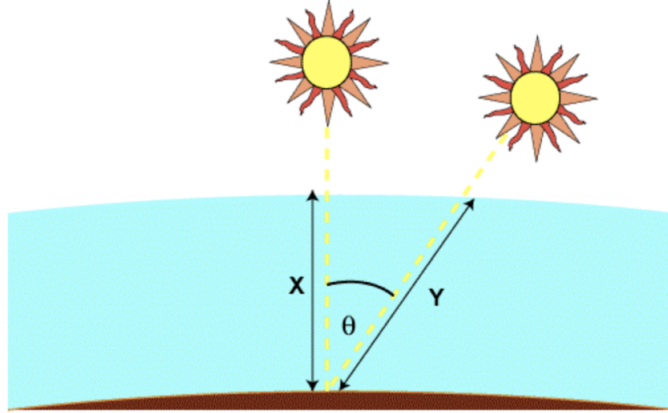


Figure 2.3: Schematic of Air Mass (Bowden, 2019)

The air mass formula assumes that the atmosphere is a flat layer, not including the curvature of Earth. At degrees close to the horizon, the air mass is not equal to the atmospheric distance. The sun at 90 degrees zenith angle, results in an infinite air mass, where the distance through the atmosphere is not. Equation (2.5) includes the curvature of the Earth (Bowden, 2019).

$$AM = \frac{1}{\cos(\theta) + 0.50572(96.07995 - \theta)^{-1.6364}} \quad (2.5)$$

Equation (2.5) yields an AM close to 35 at degrees close to the horizon compared to the infinite result from Equation (2.4). Taking the atmospheric effects, local variations, varying spectral content and AM into account, the direct irradiance at the surface can be calculated. Excluding the effects of height above sea level, the direct irradiance can be calculated with good accuracy as a function of AM

$$I_{dn} = 1.367 \cdot 0.7^{AM^{0.678}} \quad (2.6)$$

Where I_{dn} is the solar intensity on a disc perpendicular to the sun's rays, given in kW/m². The constant 0.7 is derived from the fact that approximately 70% of the incident radiation is transmitted through the atmosphere. The

constant 1.367 is the solar constant in kW/m², and 0.678 is an empirical constant fitted with observed data (Bowden, 2019). The intensity of sunlight increases with height above sea level. In order to achieve better accuracy in calculating the intensity of sunlight, the height is taken into account. The direct component is expressed as

$$I_{dn} = 1.353 \cdot \left[(1 - a \cdot H) \cdot 0.7^{AM^{0.678}} + a \cdot h \right] \quad (2.7)$$

Where H is the height above sea level in km and a is an empirical constant with the value 0.14 (Bowden, 2019).

The distance of atmosphere the solar radiation have to travel through is dependent on the zenith angle of the sun. Due to both dealing with inhomogeneous atmosphere and continuity issues close to the horizon, the calculation of distance with several models with different restrictions. Due to not looking at close to horizon angles, the non-refracting spherical atmosphere model does meet the given limitations (Sterken, 1992). The path of light, s, is given by

$$s = \sqrt{R_E^2 \cdot \cos^2(z) + 2R_E \cdot y_{atm} + y_{atm}^2} - R_E \cos(z) \quad (2.8)$$

where R_E is the radius of Earth, y_{atm} is the height of the atmosphere, and z is the zenith angle of the incident sunlight. If the atmosphere is assumed homogeneous, the hydrostatic considerations yield that

$$y_{atm} = \frac{k \cdot T_0}{mg}$$

where k is Boltzmann's constant, T_0 is the temperature at sea-level, m is the mass of air and g is the gravity of Earth.

2.4 Cloud Enhancement

Cloud Enhancement is a known phenomenon where the irradiance exceeds the expected irradiance from a clear day. Cloud edges cause reflection, which enhanced the normal radiance. The observation of Cloud Enhancement is

worldwide, and the reason is a highly discussed matter. The Cloud Enhancement phenomenon is mainly due to strong forward Mie scattering inside the cloud, and the strongest CE events occur when thin clouds surround a narrow gap within 5° around the solar disk (Yordanov, 2013).

2.5 Clearness and Diffuse Index

The fraction between the extraterrestrial irradiance and global irradiance, I_g/I_{ex} , is known as the clearness index, k_t . The index normally ranges from 0 to 1, where a high index corresponds to a clear sky, and a low index corresponds to many clouds blocking the sun. Due to the phenomenon mentioned above, cloud enhancement, k_t can exceed the value of 1 in rare cases. Another useful relation is the diffuse index, k_d , which is the fraction between the diffuse irradiance and the global irradiance, I_d/I_g (Böhme, 2019). The index describes how much of the global irradiance measures as diffuse irradiance. The situation of a cloudy sky would result in high index number, close to 1. In contrast, a cloudless situation would result in almost no diffuse irradiance, and therefore a low index number close to 0.

2.6 Modelling Solar Irradiance

Measurements of solar irradiance components such as diffuse and direct irradiance are not available worldwide, as sufficient measurement equipment are both cost-intensive and require routine maintenance. In high-latitude regions above or below $\pm 60^\circ$ in the northern and southern hemisphere, such measurements are rare. Several models have been for estimating diffuse or direct fractions based on global irradiance data. These models require different input data, and the components can be estimated from different time intervals, ranging from minutes to monthly averaged data. There are two main categories of solar irradiance models: parametric and decomposition models.

The parametric model is based on physical principles and requires detailed information of atmospheric conditions such as atmospheric turbidity, type and amount of clouds, and precipitable water content (Wong, 2001). Decomposition models bases themselves on extracting solar irradiance parameters

and converting them into components from existing data. These models often use global irradiance as the only input parameter to predict the diffuse and direct components using the empirical correlation between GHI and DNI or DHI (Berstrand, 2015).

2.6.1 Skartveit and Olseth

Skartveit and Olseths decomposition model was developed using measurement data from Bergen, Norway. For higher latitudes above 60° in the northern hemisphere, (Böhme, 2019) found that Skartveit and Olseth gave the best results compared with the other investigated models: Reindl, Boland/Ridley/Lauret and Maxwell&Perez . The model only uses clearness index, k_t , computed from the global irradiance I_g and the suns height h as inputs variables. First the hourly variability index σ_3 is calculated. Different scenarios are then taken into account to further calculate the diffuse fraction index $k_{d,mod}$ (Böhme, 2019). The hourly variability index is defined as

$$\sigma_3 = \left(\frac{(\rho - \rho_{-1})^2(\rho - \rho_{+1})^2}{2} \right)^2 \quad (2.9)$$

and the clear sky index, ρ is defined as

$$\rho = \frac{k_t}{k_1} \quad \text{with} \quad k_1 = 0.83 - 0.56 \cdot \exp(-0.06 \cdot h) \quad (2.10)$$

k_1 is called the cloudless clearness index.

If either ρ_{+1} or ρ_{-1} is missing, σ_3 is defined as $|\rho - \rho_{\pm 1}|$. In the case that both are missing, σ is defined as

$$\begin{aligned} \sigma_3 &= 0.021 + 0.0397 \cdot \rho - 0.231\rho^2 - 0.13 \cdot \exp\left(-\left(\left(\frac{\rho - 0.931}{0.134}\right)^2\right)^{0.834}\right) & \text{for } \rho &\geq 0.14 \\ \sigma_3 &= 0.12 + 0.65 \cdot (\rho - 1.04) & \text{for } \rho &< 0.14 \end{aligned}$$

The model is then split into two different cases, dependent on the outcome of the hourly variability computation $\sigma_3 > 0$ and $\sigma_3 \approx 0$.

1. For the case of $\sigma \approx 0$, called invariable hours, four different equations arise for $k_{d,mod}$, dependent on the value of the clearness index k_t . The four different cases are:

I $k_t \leq 0.22$: S&O assume that for $k_t \leq 0.22$, many clouds with no direct beam are present and hence

$$k_{d,mod} = 1 \quad (2.11)$$

II $0.22 \leq k_t \leq k_2$: It is assumed that clouds partly obscure the sun and some direct beam is present. In that case

$$k_{d,mod} = 1 - (1 - d_1)(0.11\sqrt{K(k_t)} + 0.15K(k_t) + 0.74K(k_t)^2) \quad (2.12)$$

with

$$K(x) = 0.5 \left(1 + \sin \left(\pi \cdot \frac{x - 0.22}{k_1 - 0.22} - \frac{\pi}{2} \right) \right),$$

$$d_1 = 0.07 + 0.046 \cdot \frac{90 - h}{h + 3},$$

$$k_2 = 0.95 \cdot k_1$$

III $k_2 < k_t \leq k_{t,max}$: A nearly cloudiness sky is assumed that leads to a small diffuse index. Here $k_{d,mod}$ is given by

$$k_{d,mod} = \frac{d_2 \cdot k_2 \cdot (1 - k_t)}{k_t \cdot (1 - k_2)} \quad (2.13)$$

with

$$k_{t,max} = \frac{k_{b,max} + \frac{d_2 \cdot k_2}{1 - k_2}}{1 + \frac{d_2 \cdot k_2}{1 - k_2}},$$

$$d_2 = 1 - (1 - d_1)(0.11\sqrt{K(k_2)} + 0.15K(k_2) + 0.74K(k_2)^2),$$

$$k_{b,max} = 0.81^{\sin(h)^{-0.6}}$$

and k_2 is the same as above. Here d_2 resembles the upper bound value of $k_{d,mod}$ from the previous case (II). Using this in the new $k_{d,mod}$ ensures that it is continuous at $k_t = k_2$

IV $k_{t,max} < kt$: There are no obscuring from clouds, but diffuse irradiance is present due to clouds in the sky. Therefore:

$$k_{d,mod} = 1 - \frac{k_{t,max} \cdot (1 - k_{d,max})}{k_t} \quad (2.14)$$

with

$$k_{d,max} = \frac{d_2 \cdot k_2 \cdot (1 - k_{t,max})}{k_{t,max} \cdot (1 - k_2)}$$

The continuity is used here again, as $k_{d,max}$ equals the upper bound value from the previous case at $k_t = k_{t,max}$ in (2.14).

2. For $\sigma > 0$, called variable hours, a new term $\Delta(k_t, h, \sigma_3)$ dependent on three different cases is added to the equations (2.11), (2.12), (2.13) and (2.14) for $k_{d,mod}$ of the first case.

I $0.14 \leq kt \leq kx$: Then

$$\Delta(k_t, h, \sigma_3) = -3 \cdot k_l^2 \cdot (1 - k_l) \cdot \sigma_3^{1.3} \quad (2.15)$$

II $k_x < k_t \leq (kx + 0.71)$: Then

$$\Delta(k_t, h, \sigma_3) = 3 \cdot k_r \cdot (1 - k_r)^2 \cdot \sigma_3^{0.6} \quad (2.16)$$

III $k_t < 0.14$ or $k_t > (kx + 0.71)$: Then

$$\Delta(k_t, h, \sigma_3) = 0, \quad (2.17)$$

with

$$\begin{aligned} k_x &= 0.56 - 0.32 \cdot \exp(-0.06 \cdot h), \\ k_l &= \frac{k_t - 0.14}{k_x - 0.14}, \\ k_r &= \frac{k_t - k_x}{0.71} \end{aligned}$$

The overall equation for the second case then resembles the form:

$$k_{d,mod,2nd} = k_{d,mod} + \Delta(k_t, h, \sigma_3) \quad (2.18)$$

with k_{mod} from from (2.11)-(2.14) and Δ from (2.15)-(2.17)

The Skartveit&Olseth model has to take all these twelve different scenarios into account when computing the indexes.

Albedo Correction

In order to improve the model, Skartveit and Olseth also proposed correction for albedo factors r significantly differing from their assumed albedo r^* of 0.15. The model was developed in a snow-free environment corresponding to an albedo of 0.15. An albedo greater than 0.15 directly links to a higher clearness index and diffuse fraction due to reflections of irradiance from the ground and sky (Böhme, 2019).

The aim of the albedo corrected version is to find the clearness index k_r that corresponds to an albedo of 0.15. This corrected clearness index is then used as input for the model, and the diffuse fraction output is re-translated to the actual albedo. For this correction, the atmospheric absorption fraction A with a value of 0.2 is introduced, which is independent of solar elevation and cloudiness. The atmospheric albedo R is equal for ground reflected radiation and direct radiation from the sun for a solar height above the horizon of $h = 37^\circ$ (Böhme, 2019). The first step is the correction of the clearness index from solar height h' to the solar height $h=37^\circ$ for the actual albedo r by:

$$k'_r = k_r \cdot \frac{k_1(h')}{k_1(h)} \quad (2.19)$$

with the definition of k_1 in (2.7). This result allows the calculation of the cloud-dependent atmospheric albedo R with

$$R = \frac{1 - A - k'_r}{1 - k'_{r,r}} \quad (2.20)$$

while all resulting values below 0.08 are replaced with 0.08. Now it is possible to compute the clearness index k^*_r as

$$k^*_r = k_r \cdot \frac{1 - r \cdot R}{1 - r^* \cdot R} \quad (2.21)$$

This result is then input for the model, resulting in the diffuse fraction $k^*_{d,r}$. The actual diffuse fraction is then:

$$k_{d,r} = 1 - \frac{k^*_r \cdot (1 - k^*_{d,r})}{k_r} \quad (2.22)$$

The albedo correction is then used the station. The first step of this improvement of the model is to estimate the actual surface albedo from the stations' climate, region and yearly duration of snow cover. The deviation from $r^*=0.15$ to r is divided into two cases. A value above 0.1 is applied the correction of the clearness index. The correction method is only developed for significantly different r values from 0.15, and deviation values below 0.1 are therefore not corrected (Böhme, 2019).

2.7 Solar Elevation Angle

The sun's position over the horizon, here called solar elevation α , also referred to as solar height, is dependent on three variables: Hour Angle (HRA), declination angle δ , and latitude ϕ of the given location. The declination angle δ is a seasonal variability due to the tilt of the Earth around its own axis and the rotation around the sun. It varies from -23.45 to 23.45 and is only dependent on the day of the year (Bowden, 2019). It has two different definitions depending on what part of the hemisphere it is calculated for

$$\delta = 23.45^\circ \cdot \cos\left(\frac{360}{365} \cdot (d - 81)\right) \quad (2.23)$$

$$\delta = 23.45^\circ \cdot \cos\left(\frac{360}{365} \cdot (d + 284)\right) \quad (2.24)$$

Where d is the day of the year, starting at $d=1$ at January 1st. Formula 2.11 is for the northern hemisphere, while Formula 2.12 is for the southern hemisphere.

The Earth rotates 15° each over, and the Hour Angle converts the Local Solar Time, LST, into an angular motion of the sun in degrees. The Local Solar Time can be calculated with the difference in a timezone from Universal Coordinated Time(UTC) and a time correction factor. The time correction is defined as

$$TC = 4(Longitude - LST M) + EoT \quad (2.25)$$

where

$$EoT = 9.87\sin(2B) - 7.53\cos(B) - 1.5\sin(B) \quad (2.26)$$

for

$$B = \frac{360}{365}(d - 81)$$

and

$$LSTM = 15^\circ \cdot \Delta UTC \quad (2.27)$$

The Solar Angle can then be expressed as

$$\alpha = \arcsin \left[\sin(\delta) \sin(\phi) + \cos(\delta) \cos(\phi) \cos(HRA) \right] \quad (2.28)$$

2.8 Sun Tracker

A Sun Tracker is a device that accurately focuses on the sun at all times. They can be used to maximize the amount of irradiance collected by a solar collector, or in measurements of solar irradiance. There are two types of tracking available: Single-axis tracking or two-axis tracking (Solanki, 2015). Single-axis tracking, the device is rotated around a single axis, while a two-axis tracking system rotates two axes. The latter gives more precise tracking as the sun's insolation varies in two axes: solar azimuthal angle and solar elevation angle. A measurements device can be mounted to the sun tracker, thus resulting in the incidence angle of the sun always being zero (Solanki, 2015).

2.9 Pyranometer

A pyranometer is an instrument which measures the total irradiance hitting the surface of the Earth. The detectors in these instruments are independent of the wavelength for the incident photons across the solar spectrum. To efficiently measure direct irradiance at lower angles of the sun, they must have a response independent of angle of the incident photons (Duffie, 2019).

Most of the available data on solar irradiance is obtained from pyranometers (Duffie, 2019). There are several different types and models of pyranometers which have different characteristics. The two most common types of pyranometers is silicon photocells and thermopile (Duffie, 2019). The best-suited

pyranometer for measurement is based on the conditions of the location. For this project, two thermopile pyranometers are used.

Thermopile pyranometer

A thermopile pyranometer is a device that utilizes temperature measurements with thermocouples. A thermocouple consists of two dissimilar metals which are coupled together, creating a junction between the two metals. As the top metal is exposed to sunlight, causing a temperature change. This temperature change causes a voltage across the junction, which then can be used to calculate the temperature the thermocouple was exposed to. A thermopile pyranometer consists of several thermocouples connected in series or in parallel (Soluzione Solare, u.d). Another important element of the pyranometer is the protective glass dome, ensuring the correct sensitivity. The dome neglects the effect of wind cooling down the hot side of the junction, which alters the measurement of temperature and irradiance. Furthermore, pyranometers consist of a black coating on the surface to ensure the absorption of a wider spectre of the incoming irradiance (Kipp & Zonen, 2015). A thermopile pyranometer is capable of absorbing wavelengths of 300-3000nm (Soluzione Solare, u.d). While a silicon photocell based pyranometer only can absorb wavelengths of 300-1100nm, due to the bandgap value of Silicon (Hinckley, 2017).

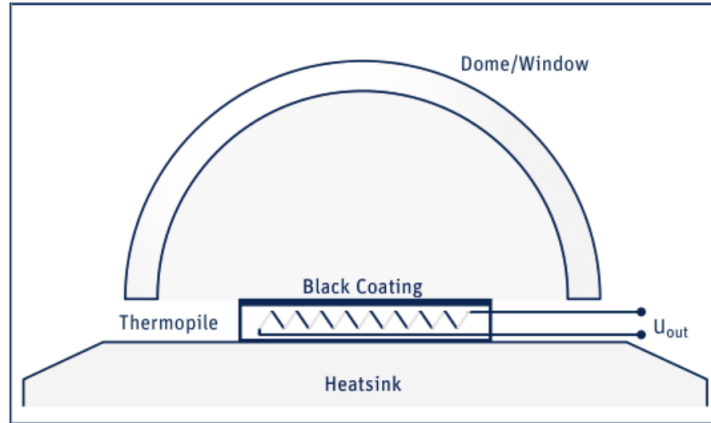


Figure 2.4: Schematic of Pyranometer (Kipp & Zonen, 2015)

2.10 Pyrheliometer

A pyrheliometer is a device used to measure the direct/beam irradiance. It uses the same principles of temperature change to measure the irradiance. However, the pyrheliometer consists of a collimating tube, with the thermocouple at the end of it (Duffie, 2019). The tube allows only a small portion of the diffuse irradiance, which is within the acceptance angle of the tube. The tube should be directed towards sun to allow the beam of direct irradiance to reach the sensor. Consequently, the pyrheliometer is very sensitive to the alignment of the instrument. This sensor is often a thermopile sensor (Solanki, 2015).

2.11 Calibration Standards

In order to measure the solar irradiance with the highest possible accuracy, the device has to be calibrated correctly. The existing methods for calibration are often with an existing pyrheliometer or pyranometer of a higher standard as reference (Kipp & Zonen, 2018). The challenges of calibration are the method of calibration that could alter the results and thus, errors for the device. The calibration should have a high-quality standard. The

International Organization for Standardization have developed the ISO-9000 standard, which is an accepted world standard for quality management of devices (ISO, u.d).

ISO-9846 is an accepted calibration method of pyranometers with pyrhelimeter as the reference. The method involves taking measurements with both devices under clear sky condition. Measurements need to include with and without shading for the pyranometer, and these are then compared to the reference pyrhelimeter (Eikeland, 2019).

Another calibration method which meets the ISO-9000 standards is the ISO-9847. This method could be conducted either indoors or outdoors. The indoors version include referencing the pyranometer with another pyranometer of same or higher quality. This is done indoors in a laboratory with a light source with a good approximation to the solar spectrum. For outdoors calibration, the pyranometer is referenced to a pyranometer with same conditions as the indoor version. The calibration should take place under a clear sky, and the position of incoming irradiance should be close to normal incident irradiance (Kipp & Zonen, 2018).

ISO-9059 is a calibration standard used for pyrhelimeters. This method uses a pyrhelimeter with similar or higher quality as reference. The pyrhelimeter used as a reference is ideally an absolute cavity pyrhelimeter, where the same procedure as pyranometer is used to calibrate (Kipp & Zonen, 2018).

Both the EKO Instruments MS-80 Thermopile Pyranometer and the MS-57 Thermopile Pyrhelimeter are calibrated to the ISO-9060 calibration standard. According to EKO Instruments, both the pyranometer and pyrhelimeter needs re-calibration at least every five years. The sun tracker does not require any re-calibrations or re-adjustments (EKO Instruments, u.d.).

Chapter 3

Method and Instruments



Figure 3.1: Setup of measurement system at Nordlysobservatoriet

3.1 EKO STR-22G Solar Tracker

The system setup at Nordlysobservatoriet uses EKO MS-80 Pyranometer in the measurement of both GHI and GNI. MS-80 is compliant to the "Fast response" and "Spectrally flat" sub-category under ISO 9060:2018 Class A

For the GNI, the second pyranometer is attached to the second arm swivel with a mounting plate. The GNI measurement is scarce often not included for stations worldwide(EKO, u.d.).

3.1.1 EKO STR-22G

EKO STR-22G is a compact two-arm sun tracker. It guarantees accurate sun tracking and pointing of the attached sensors by adjustment due to a closed-loop control system. It supports all kinds of global, diffuse and direct irradiance measurements, with various alternatives for mounting. It is equipped with an automated setup procedure through a GPS receiver, with a working range of 0 to 360 degrees for the azimuth angle and -15 to 95 degrees for the zenith angle(EKO,u.d.).

3.1.2 EKO MS-57 Pyrheliometer

EKO MS-57 Pyrheliometer is a direct normal irradiance sensor also compliant to the "Fast response" and "Spectrally flat" sub-category under ISO:9060:2018 Class A. The pyrheliometer has a fast thermopile response of fewer than 0.2 seconds and low thermal offset. The responsiveness to solar irradiance is ranging between 200 to 2000nm and works under temperatures between -40 to 80 degrees celsius. In five years, the longterm stability of the tensors responsivity is less than 0.5% (EKO, u.d.).

3.1.3 EKO MS-80 Pyranometer

Global horizontal irradiance and global normal irradiance were measured using an EKO Instruments MS-80 pyranometer. Another EKO Instruments MS-57 pyrheliometer was used for measurement of direct normal irradiance. All of the sensors used for measurement is mounted to an EKO Instruments STR-22G Sun Tracker. A Campbell Scientific CR-6 Datalogger is also included in the setup for storing and forwarding data to an external source.

The system setup for Nordlysobservatoriet is found in Table (3.1).

Table 3.1: Measurement Equipment installed at Nordlysobservatoriet

Equipment	Measurement Purpose
EKO MS-80 Thermopile Pyranometer	Global Horizontal Irradiance
EKO MS-80 Thermopile Pyranometer	Global Normal Irradiance
EKO MS-57 Thermopile Pyrheliometer	Direct Normal Irradiance
EKO STR-22G Sun Tracker	2-Axis Tracking
Campbell Scientific CR-6	Datalogger
Sierra Wireless Airlink RV50	Wireless Modem

3.2 Dataset

The dataset from Nordlysobservatoriet in Tromsø, Norway is provided by Microsoft Azure Storage access provided by University of Tromsø ITA department. The datasets consists of 15-minute intervals for Global Normal Irradiance, Global Horizontal Irradiance and Direct Normal Irradiance. The data is quality checked and reliable from 29th of April 2020, and the range of used data is from 1st of May 2020 to 1st of June 2020.

In order to compare the results, external datasets from different stations worldwide have been obtained. Range of latitudes of the external datasets is between 28.31° and 58.25° and were provided by Baseline Surface Radiation Network (BSRN)(Driemel, 2019).

BSRN is a project of the Data and Assessments Panel from the Global Energy and Water Cycle Experiment (GEWEX). The aim is to detect important changes in the radiation field at the Earth’s surface, which may be related to climate change (Driemel, 2019). The objective of BSRN is to provide observations of high quality for surface radiation fluxes with a high sampling rate. The collected observations are from a small number of selected stations in diverse climatic zones.

The datasets are in line with data release guidelines of the BSRN, and the format is consistent for all station (Driemel, 2019). The time resolution for these datasets is one second for Global Normal Irradiance, Diffuse Horizontal Irradiance and Direct Normal Irradiance. BSRN offers Data Publisher for Earth Environmental Science service PANGAEA for data retrieval, while FTP-access is also a viable option. In order to gain access to PANGAEA, a read account can be obtained from Amelie Driemel. More information regarding data retrieval from BSRN is found at <https://bsrn.awi.de/?id=386>

Table 3.2: Overview of the stations for comparing the radiation measurements and indexes

Station	Latitude	Elevation Above Sea Level	Data Source
Tromsø, Norway	69.65	100	UiT
Cener, Spain	42.82	471	BSRN
Izana, Spain	28.31	2373	
Toravere, Estonia	58.25	70	
Lindenberg, Germany	52.21	125	

3.2.1 Data Quality Control

Measurement instruments are limited when measuring the irradiance at lower solar elevations. In order to improve the quality of the data, the datasets are filtered after calculating the needed parameters to remove bad or faulty measurements. The filter uses the parameters proposed by the European Commission Daylight I in 1993 (Jacovides, 1993), removing all data not

satisfying the five parameters:

$$0 < k_t \leq 1 \quad (3.1)$$

$$0 < k_d \leq 1 \quad (3.2)$$

$$\frac{I_d}{I_{ex}} > 0.8 \quad (3.3)$$

$$I_{gh} < 5 \frac{W}{m^2} \quad (3.4)$$

$$h < 5^\circ \quad (3.5)$$

The first two filters (3.1) and (3.2), filters out theoretically impossible clearness and diffuse indexes from measured data. (3.3) filters out too high diffuse irradiance in comparison to the total available irradiance. (3.4) filters out signal noise, while (3.5) filters out measurements at lower solar heights which would lead to high measurement errors.

Chapter 4

Results and Discussion

4.1 Irradiance Reduction Analysis

The reduction in solar irradiance from the extraterrestrial irradiance decreases with the solar elevation angle (h) above the horizon. The increase in Air Mass or length of atmosphere causes a decrease in the amount of sunlight remaining at the surface of Earth. This can be seen in figure (4.2). The reduction is defined as

$$Reduction_{GH} = 1 - \frac{I_{gh}}{I_{ex}} \quad (4.1)$$

$$Reduction_{DN} = 1 - \frac{I_{dn}}{I_{ex}} \quad (4.2)$$

The solar elevation angle is calculated from the time of measurement according to section 2.7. The data is focused at 5-degree intervals, including an error of $\pm 0.25^\circ$, which corresponds to a margin of error of 5 per cent. The AM for given solar elevation angles calculated using formula (2.5) is presented in Table (4.1).

The distance of atmosphere is another method of describing the Air Mass at different angles of elevation. The distance of atmosphere the sunlight have to pass through to reach the surface of the Earth can be calculated from Equation (2.8) and is presented in Table (4.1).

Table 4.1: Air mass and distance at given solar elevation angles

Elevation Angle [$^{\circ}$]	Air Mass	Distance of Atmosphere [m]
5	10.31	12818
10	5.59	12301
15	3.81	11814
20	2.90	11359
25	2.36	10937
30	1.99	10550
35	1.74	10198
40	1.55	9880
45	1.41	9595

The irradiance reduction analysis is based on days with clear skies. In order to obtain a dataset of only clear days, the data has to be filtered. All days have been plotted, and filtered manually into a separate file for this purpose.

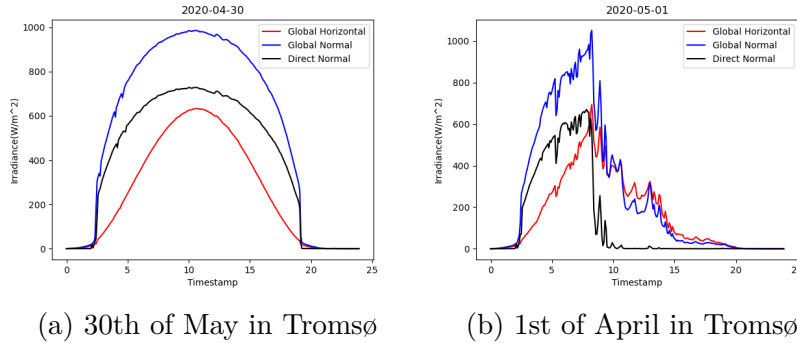


Figure 4.1: Difference between cloudless and clear sky day

Figure 4.1(a) is a typical plot of a clear sky condition. The plot is symmetrical with a smooth curve. Figure 4.1(b) shows a cloudy day, resulting in a random drops and peaks in irradiance. The number of days during May 2020 with completely clear weather for the different locations is presented in

Table (4.2). The elevation above sea level of the measurement site is also stated and is important for the coming irradiance analysis.

Table 4.2: Overview of clear days found after manually checking the dataset

Station Name	Clear day number	Elevation above sea level [m]
Tromsø, Norway	2	100
Cener, Spain	4	471
Izana, Spain	23	2373
Toravere, Estonia	2	70
Lindenberg, Germany	4	125

4.1.1 Reduction of Global Horizontal Irradiance

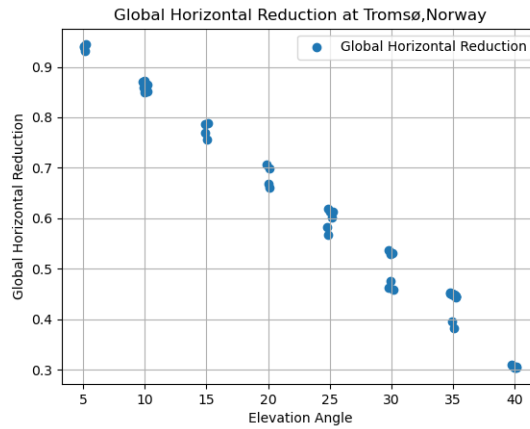
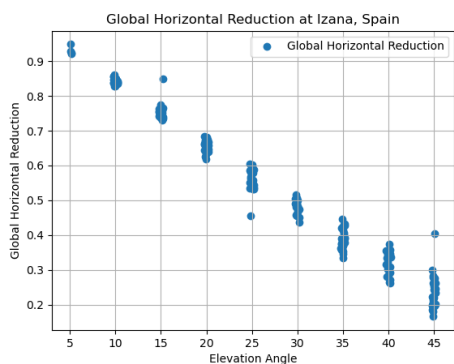


Figure 4.2: Reduction in Global Horizontal Irradiance for given angles in Tromsø, Norway

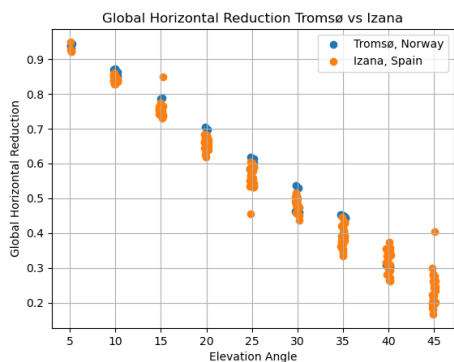
The reduction in Global Horizontal Irradiance shows a linear decrease with solar elevation angle. At 5 degrees solar elevation in Tromsø, the reduction is approximately 95%, thus almost no radiation passes through the

atmosphere. At the higher measured solar elevations of 40degrees, 70% of the radiation passes through the atmosphere. From Table (4.1), 5 degree elevation angle corresponds to an AM of 10.31, while at 40 degrees elevation angle there is only an AM of 1.55.

Izana, Spain



(a) Standalone



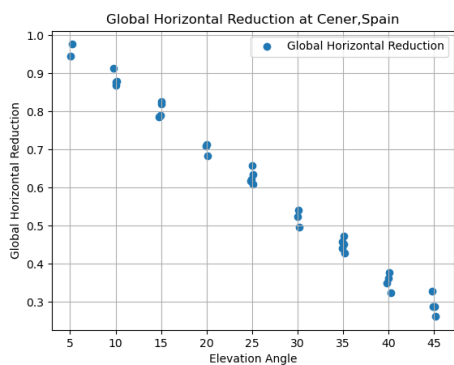
(b) Comparison

Figure 4.3: Reduction in Global Horizontal Irradiance for given angles in for Izana, Spain

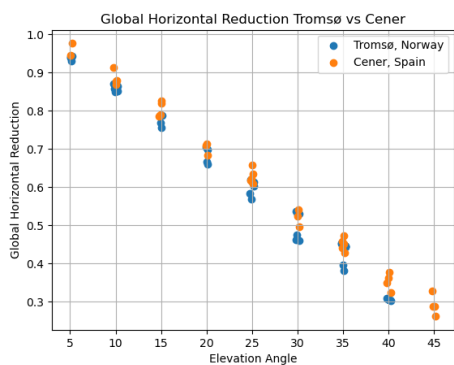
The result from Izana in the Pyrenees at an altitude of 2373m, shows a similar reduction as Tromsø. At higher solar elevation angles the reduction have a

high spread for Izana, ranging between 0.15 to 0.3 at 45 degrees. For lower elevation angles there is almost no spread in the reduction for a given solar elevation angle, ranging between 0.83 to 0.87 at 10 degrees solar elevation angle. Izana shows the same trend as Tromsø with a linear decrease with solar elevation angle.

Cener, Spain



(a) Standalone

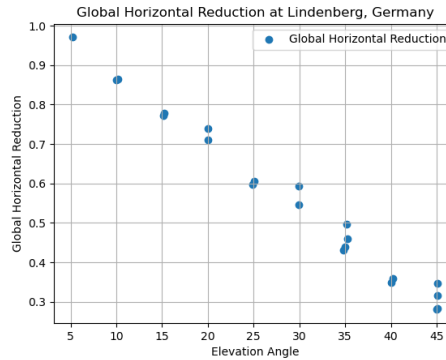


(b) Comparison

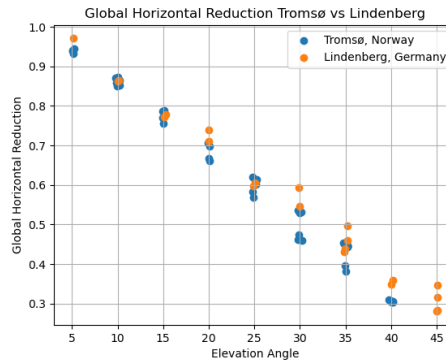
Figure 4.4: Reduction in Global Horizontal Irradiance for given angles in Cener, Spain

For Cener, at an altitude of 471m, the reduction of GHI is higher, compared to Tromsø, for all elevation angles. The increased reduction is approximately 0.05 absolute percentage for high sun elevations, while at low elevations the difference is significantly smaller. The data is more concentrated than Izana, which could be a result of fewer datapoints due to few of clear sky days.

Lindenberg, Germany



(a) Standalone

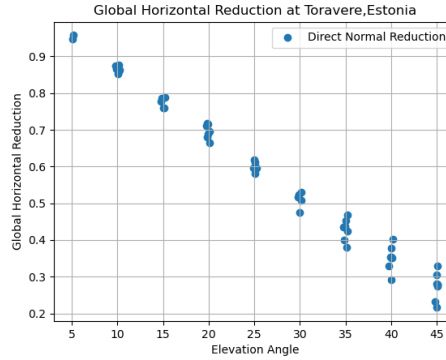


(b) Comparison

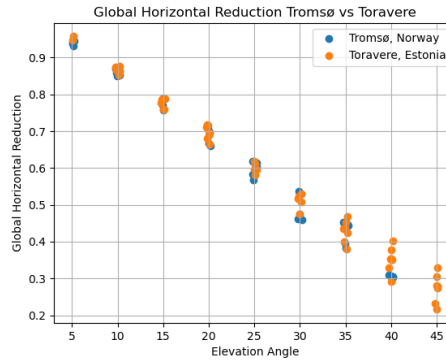
Figure 4.5: Reduction in Global Horizontal Irradiance for given angles in Lindenberg, Germany

Lindenberg shows similar results as Cener. The reduction is higher by 0.05 absolute percentage at the higher elevations, but the difference decreases for lower solar elevation angles. At 5-15 degrees solar elevation angles, the reduction is matching well. The spread of data is higher, almost counting at 10% reduction spread for 30-45 degrees solar elevation angles.

Toravere, Estonia



(a) Standalone



(b) Comparison

Figure 4.6: Reduction in Global Horizontal Irradiance for given angles in Toravere, Estonia

The comparison between Tromsø and Toravere shows similarities and matching reduction for all solar elevation angles. The datapoints from Tromsø is more concentrated than the Toravere. At 40 degrees solar elevation, Toraveres reduction is between 30-40% where Tromsø is steady at 30% reduction.

4.1.2 Reduction of Direct Normal Irradiance

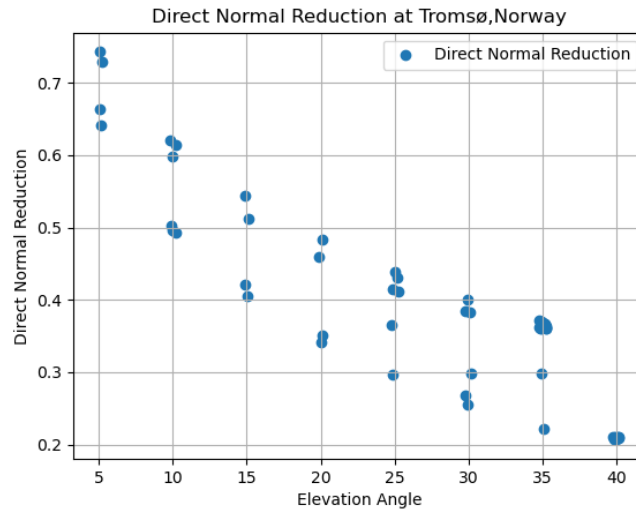
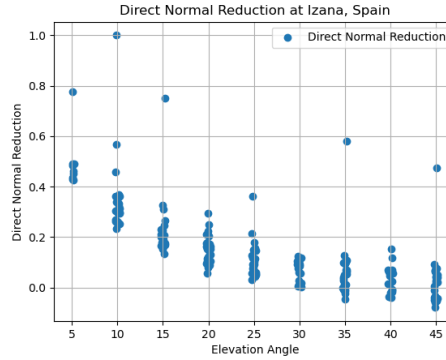


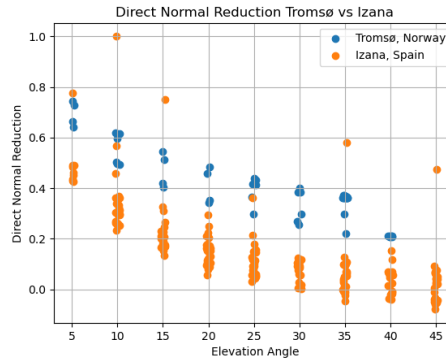
Figure 4.7: Reduction in Direct Normal Irradiance for given angles in Tromsø, Norway

The reduction in DNI in Tromsø show an exponential trend compared to the linear one for GHI in section 4.1.1. It is also noteworthy that the DNI reduction is lower for all angles compared to the GHI reduction. The reduction is 10 absolute percentages less for DNI, being at 0.2 at 40 degrees compared to the GHI reduction of 0.3. Increased solar elevation angle corresponds to a decrease in air mass and atmospheric pathway distance. From the curve of the DNI reduction, it seems like the reduction is not linearly dependent on the distance of atmosphere it travels through. Studies have shown that the direct normal irradiance is affected more by the atmospheric effects, than the global horizontal irradiance, and thus more dependent on the air mass (Brine, 1983).

Izana, Spain



(a) Standalone

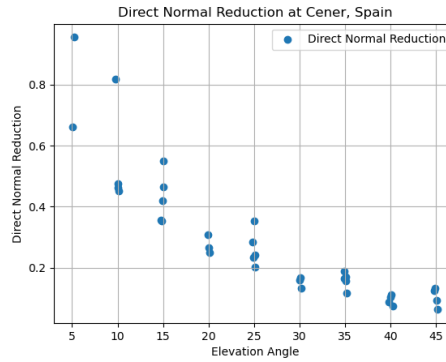


(b) Comparison

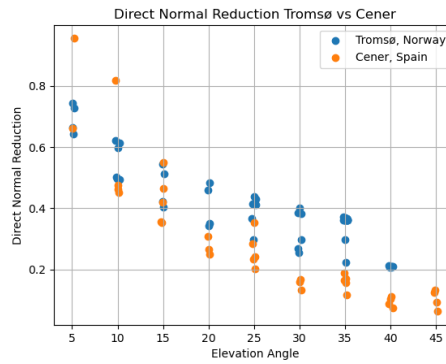
Figure 4.8: Direct Normal Irradiance for Izana, Spain

Izana has the same exponentially decaying trend. Most of the measurement are concentrated and consistently 20% lower than Tromsø. Due to Izana being located at 2373m above sea level in the Pyrenees, this result is expected. For 40 degrees the reduction is ranging from 0 up to 20%, while at 10 degrees it is ranging from 20 to 40 %. To deal with the height difference, an extra reduction is added to simulate it going through the same amount of atmosphere as Tromsø. These results will be provided in a later section.

Cener, Spain



(a) Standalone

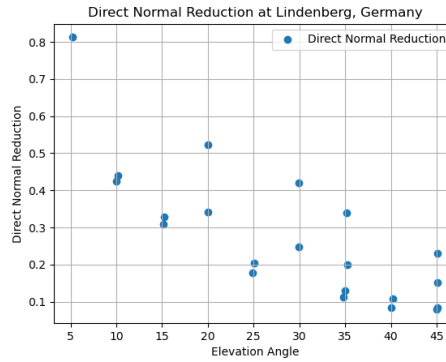


(b) Comparison

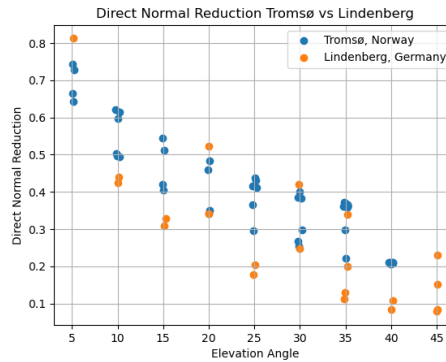
Figure 4.9: Direct Normal Irradiance for Cener, Spain

Ceners results has a sharper curve, with increased reduction at the lower solar elevation angles compared to Izana. The curve is similar to Tromsø, while the reduction at the higher solar elevation angles is lower. At 35 and 40 degrees the reduction is approximately 20% lower than Tromsø and for 35 degrees the reduction is 20% or lower.

Lindenberg, Germany



(a) Standalone

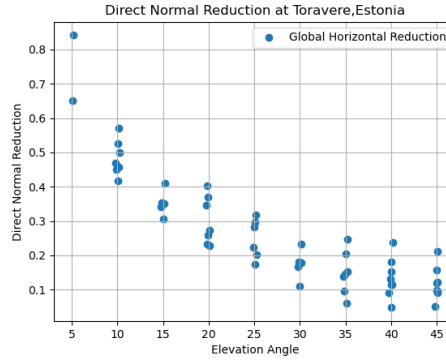


(b) Comparison

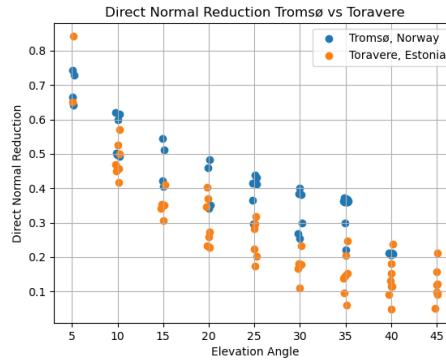
Figure 4.10: Direct Normal Irradiance for Lindenberg, Germany

The reduction results from Lindenberg are quite spread. The pattern seen from the other locations are not as apparent here. For most of the results, the same pattern shows from the previous locations with lower reduction compared to Tromsø. The reduction at 40 degrees at Lindenberg is 10%, which is significantly lower. The same difference of 10% seem constant for every elevation above the horizon.

Toravere, Estonia



(a) Standalone



(b) Comparison

Figure 4.11: Direct Normal Irradiance for Toravere, Estonia

As for the GHI at Toravere, Estonia, the DNI reduction have a big range of reductions for several elevations. The highest measured DNI reductions are lower than Tromsø's DNI reductions. As Lindenberg the reduction is generally 10% lower than Tromsø.

4.1.3 Reduction of Global Normal Irradiance

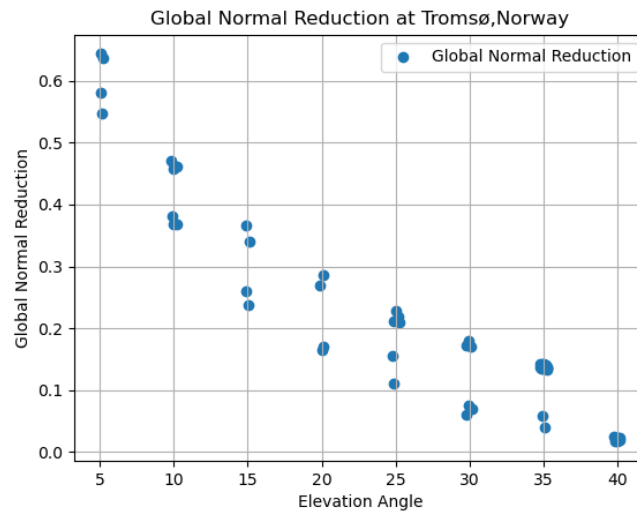


Figure 4.12: Reduction in Global Normal Irradiance for given angles in Tromsø, Norway

The Global Normal Reduction is an uncommon measurement, and it is not usual to have measurement systems obtaining such data. BSRN does not provide any measurements of GNI from any of their stations. GNI have a similar curve to DNI. The GNI reduction is lower compared to GHI for lower solar elevation angles in Tromsø. For higher elevation angles the reduction of GNI is similar to GHI.

4.1.4 Dealing with altitude difference of Izana and Cener

Izana is located at 2373m above sea level and Cener is located at 471m above sea level. The height difference between these two mentioned stations is high compared to the others located at 70-125m above sea level. The air mass the sunlight has to pass through to reach the surface is therefore approximately 2000m lower for Izana and 300m for Cener, with respect to all the other stations. In order to deal with this, the distance of atmosphere calculation given by equation (2.8) is altered with a new atmospheric distances which are respectively, 2000m and 300m lower. The results from this calculation is presented in table (4.3) and (4.4).

Table 4.3: Altered atmospheric distance and scaling at all solar elevation angles for Izana

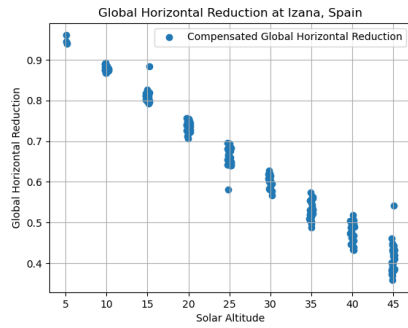
Elevation Angle [°]	y_{atm} [m]	$y_{atm,izana}$ [m]	$\frac{y_{atm,izana}}{y_{atm}}$
5	12818	10217	0.797
10	12301	9708	0.789
15	11814	9234	0.782
20	11359	8797	0.774
25	10937	8397	0.767
30	10550	8034	0.761
35	10198	7707	0.756
40	9880	7415	0.750
45	9595	7158	0.746
Mean value			0.769

Table 4.4: Altered atmospheric distance and scaling at all solar elevation angles for Cener

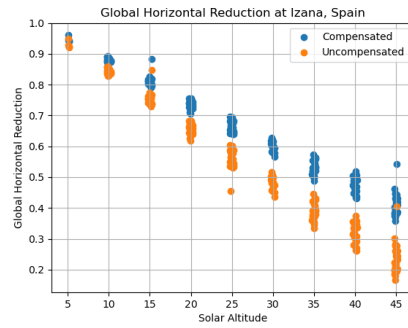
Elevation Angle [°]	y_{atm} [m]	$y_{atm,cener}$ [m]	$\frac{y_{atm,cener}}{y_{atm}}$
5	12818	12294	0.959
10	12301	11779	0.957
15	11814	11294	0.956
20	11359	10842	0.954
25	10937	10424	0.953
30	10550	10041	0.951
35	10198	9693	0.950
40	9880	9380	0.949
45	9595	9100	0.948
Mean value			0.953

The assumption made is that the atmospheric effects are constant with distance or air mass, which is not entirely correct. The atmospheric effects is dependent on the density of the air, which changes with altitude due to pressure and temperature differences (Hum, S.V, u.d.).

Izana, Spain

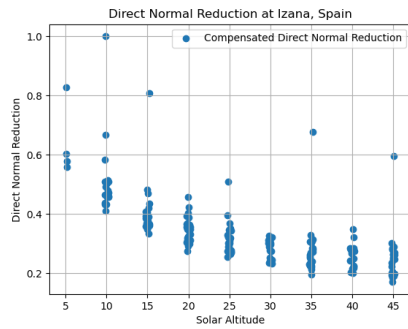


(a) Standalone

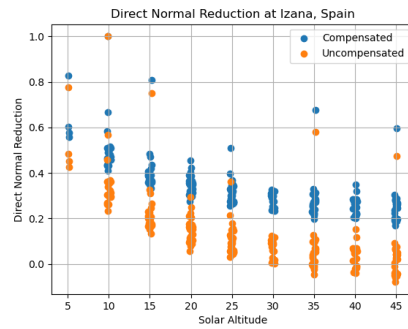


(b) Comparison

Figure 4.13: Compensated GHI reduction plots for Izana, Spain

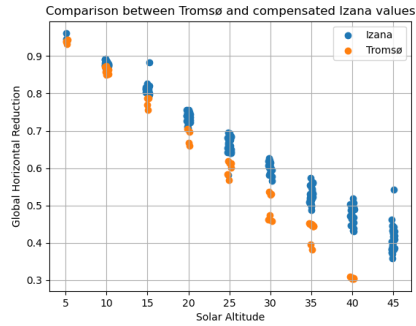


(a) Standalone

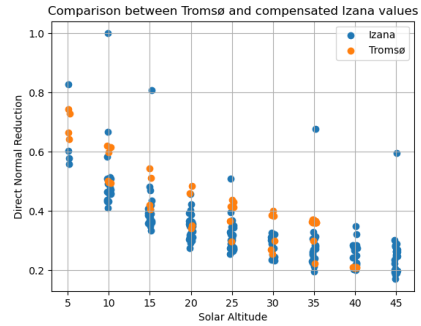


(b) Comparison

Figure 4.14: Compensated DNI reduction plots for Izana, Spain



(a) GHI comparison

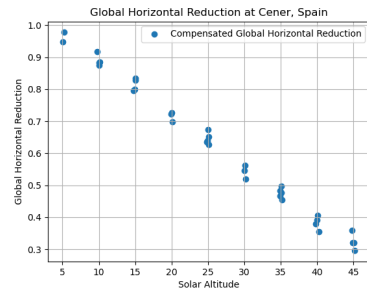


(b) DNI comparison

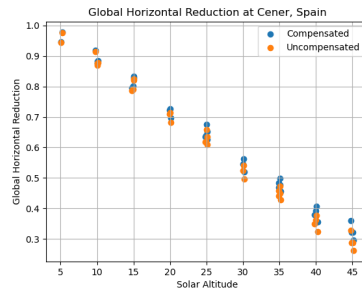
Figure 4.15: Comparison between Izana and Tromsø for compensated DNI and GHI values

The result, shown in figure (4.13)-(4.15), is that the global horizontal irradiance is increasing with the altitude above sea level. The increase varies between 3-104 % dependent on the solar elevation angle, with a mean value of 28.8% for Izana. The mean increase in reduction of DNI is 166% for Izana.

Cener, Spain

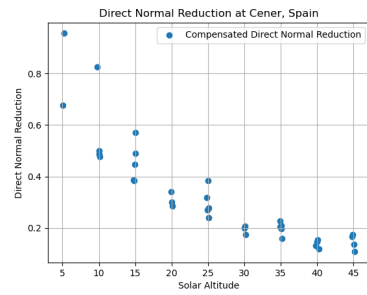


(a) Standalone

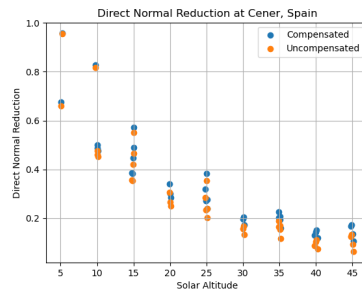


(b) Comparison

Figure 4.16: Compensated GHI reduction plots for Cener, Spain

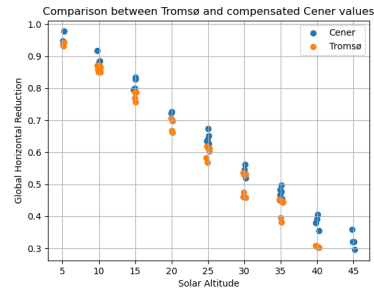


(a) Standalone

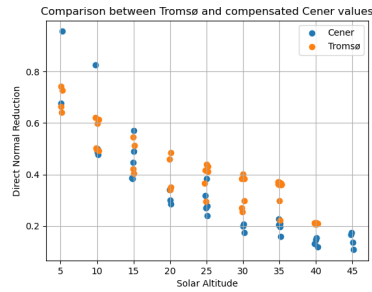


(b) Comparison

Figure 4.17: Compensated DNI reduction plots for Cener, Spain



(a) GHI comparison



(b) DNI comparison

Figure 4.18: Comparison between Cener and Tromsø for compensated DNI and GHI values

After compensating for the height of Cener, the increase in GHI reduction ranges from 0.2-13%, with a mean value of 4%. For DNI, the mean reduction increase is 21%, ranging from 2-70%. The GHI reduction is higher for Cener, while the DNI reduction is lower than Tromsø. This was the case before compensation, and is also the case for Izana, mentioned above.

4.2 Clearness and Diffuse Index Analysis

The clearness index, k_t , describes the ratio between the global horizontal irradiance and the extraterrestrial irradiance. A high clearness index is a result of a cloudless sky, while a low clearness index implies cloudy conditions. The diffuse index, k_d is the ratio between the direct radiation and the global radiation. The diffuse index is opposite of the clearness index, where a high diffuse index corresponds to a cloudy sky and vice versa.

In order to compare the clearness and diffuse index with modelled values for Tromsø, the Skartveit and Olseth model is used. The model was designed from measurements in Bergen, Norway and (Böhme, 2019) found out that it had the best fit for modelling higher latitudes. Skartveit & Olseth uses the surface albedo to incorporate the albedo correction in the model.

For Tromsø, the system is located on a roof with a black surface. According to (Li, H., 2016) the albedo of this surface is 0.2. The albedo is then set as an input in the Skartveit & Olseth model as described in section (2.6.1). Since the model is predicting the diffuse and clearness index for Tromsø and not the external stations, only the albedo for Tromsø is needed. The results are based on measurements from every day in May 2020. The measurements are filtered according to section (3.2.1) to remove signal noise and faulty measurement due to cloud enhancement or too low solar elevation angles.

Tromsø, Norway

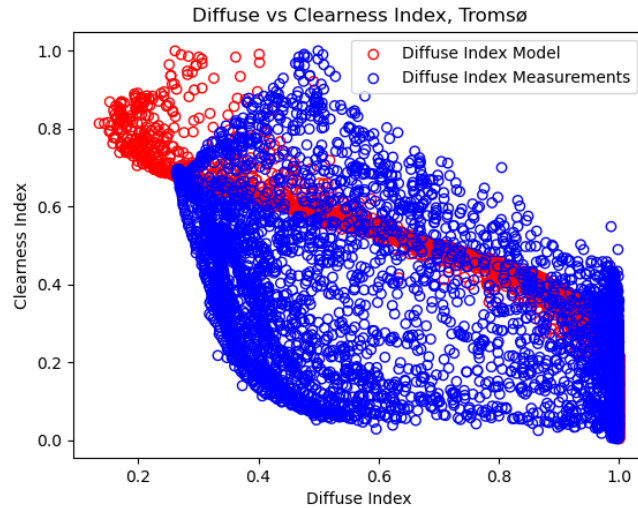
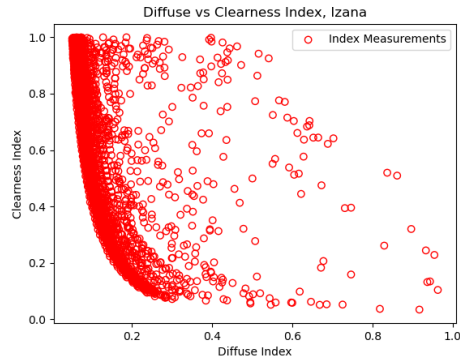


Figure 4.19: Skartveit Model of Tromsø

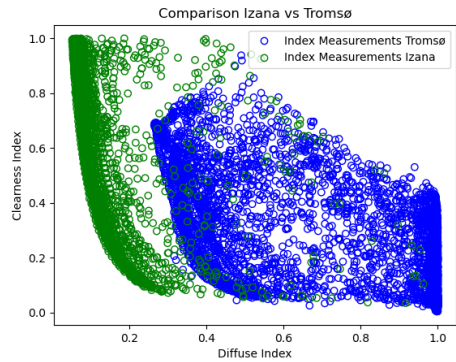
The k_d and k_t for Tromsø is illustrated in Figure (4.19). The Clearness index, k_t , for the real measurements shows a significantly higher spread in than the modelled values. Skartveit & Olseth seem to overestimate the clearness index for a wide spectre of the diffuse indexes. Between the diffuse index values of 0.4 to 0.6, this overestimation is highly present. Furthermore, the clearness index not even close towards 0.2 or lower at any point. A low clearness indicates a cloudy sky condition. Thus the Result implies no days with a high concentration of clouds. The modelled results are far from reality.

In the following text, measured clear and diffuse index values for the other measurement stations are compared with the Tromsø result.

Izana, Spain



(a) Standalone

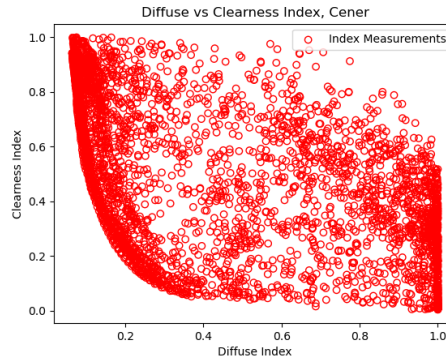


(b) Comparison

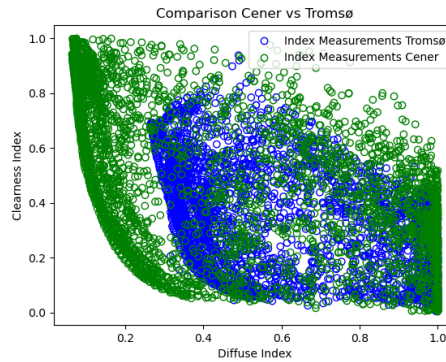
Figure 4.20: Clearness and Diffuse Index for Izana, Spain

Izanas ratio between the clearness index and the diffuse index has a steep curve. The comparison between Tromsø and Izana is illustrated in Figure (4.20). The results shows that Izana has concentrated offset towards lower diffuse indexes. A lower diffuse index is a result from less clouds. The offset is approximately 0.2 towards the left. The data from Izana show an extremely low spread, with very few random datapoints with no apparent pattern.

Cener, Spain



(a) Standalone

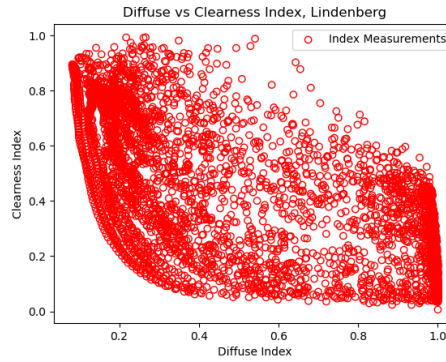


(b) Comparison

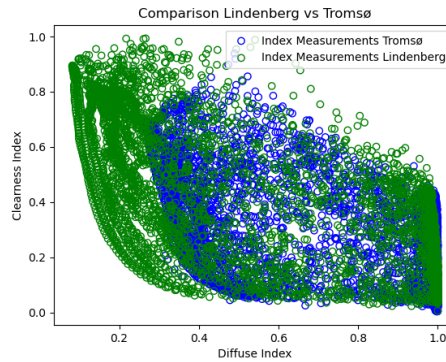
Figure 4.21: Clearness and Diffuse Index for Cener, Spain

Cener shows a more spread pattern, where parts of the data matches the diffuse and clearness index ratio from Tromsø. Cener shares the same offset difference as Izana. The difference is that the stagnation of diffuse index with increasingly clearness index starts earlier, at approximately 0.3 diffuse index instead of 0.2 in the Izana case.

Lindenberg, Germany



(a) Standalone

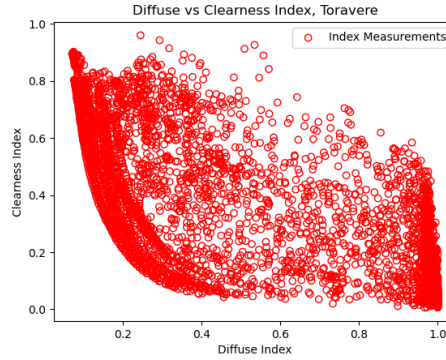


(b) Comparison

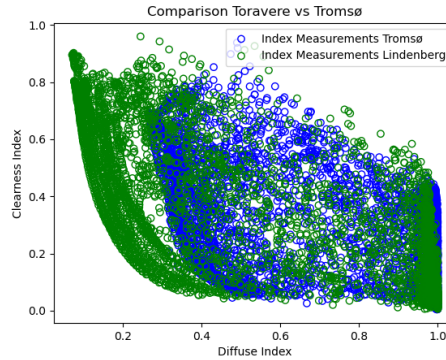
Figure 4.22: Clearness and Diffuse Index for Lindenberg, Germany

Lindenbergs data show a similar result as the two previous others. The same offset is seen in Figure (4.22). The change from almost no matching datapoints to more similarity in the index ratios is increased for Lindenberg. The offset is close to Ceners offset, with more datapoints at a clearness index of 0.3. The diffuse index have an increase concentration around 0.2 to 0.8 for this given clearness index.

Toravere, Estonia



(a) Standalone



(b) Comparison

Figure 4.23: Clearness and Diffuse Index for Toravere, Estonia

Toravere, Estonia shows more similarity towards Izana than Lindenberg. Most of the datapoints are concentrated from 0.1 to 0.2 Diffuse index with a clearness index between 0 and 0.8. The diffuse index is clearly lower for all clearness indexes compared to Tromsø, illustrated in Figure (4.23).

4.3 User Manual

The current setup at Nordlysobservatory could need maintenance or to be relocated in the future. The purpose of the manual presented is to easily change the programming or other parameters for the sun tracker if needed. The resulting programming of the datalogger and storage of measurement data was performed in cooperation with the University of Tromsø and the Department of IT.

4.3.1 Communication

In order to communicate with the CR6 Datalogger, a computer has to be connected through USB (Campbell Scientific, u.d.). It is also required to download the LoggerNet software from Campbell Scientific. More information is available at:

<https://www.campbellsci.com/loggernet>.

There is one specific program within loggernet that is crucial towards tweaking the datalogger: Device Configuration Utility. A second software, CRBasic is also helpful if the logger needs to be programmed but is not required. In order to connect to the datalogger after the USB connection is established, the Device Configuration Utility Software is used. First off, USB Drivers have to be installed on the computer. This installation can easily be done by clicking “Install USB Driver” in the middle of the program, seen in figure (4.24).

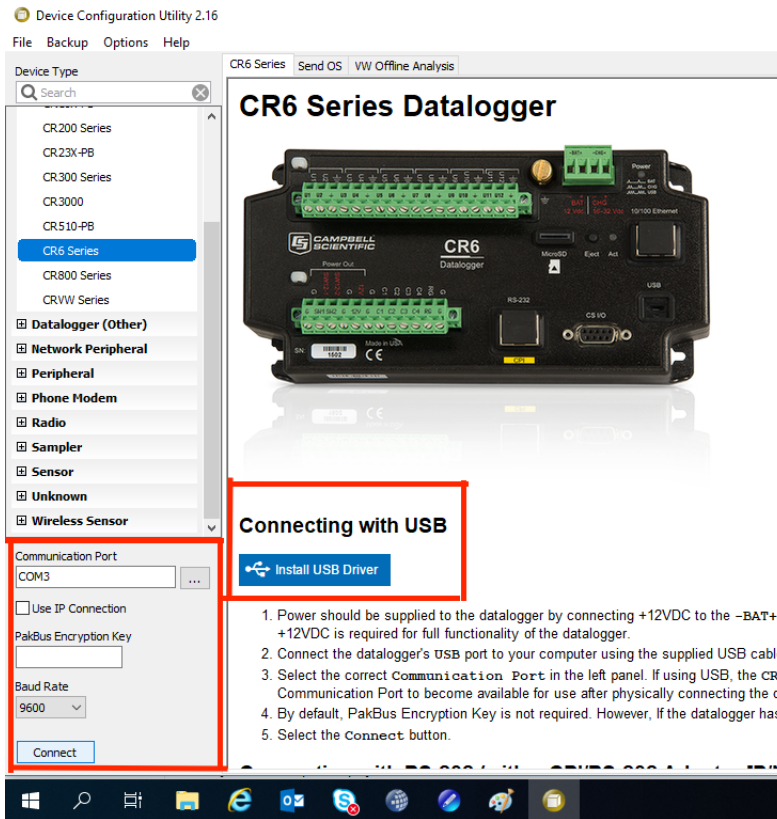


Figure 4.24: View of start window in Device Configuration Utility where connection is established

The COM-port used is selected before pressing the “Connect” button in the bottom left corner. The button should then change to “Disconnect”, signalling that the device is connected, shown in figure (4.25).

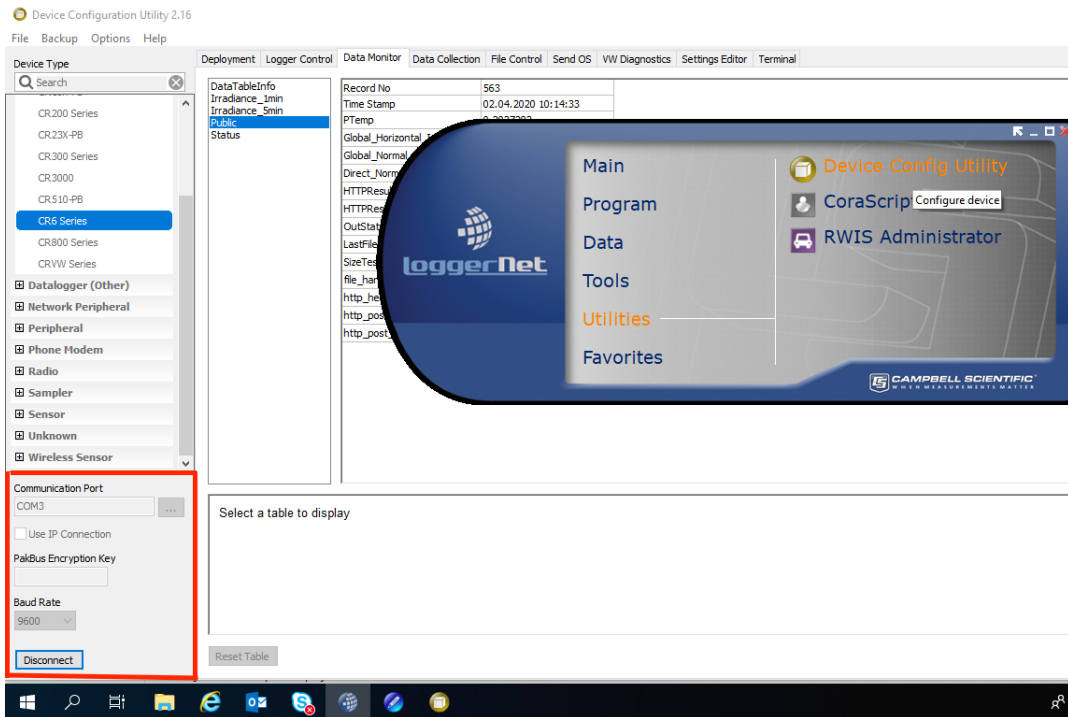


Figure 4.25: change in status in the start window from "Connect" to "Disconnect"

4.3.2 Wi-Fi Connection

In order to gain access to the internet, the data logger is connected to a Wireless Router. The router uses terminals of the datalogger to drain power. However, these terminals are usually shut off. In order to turn these on, commands are used in the CRBasic Script that runs on the datalogger. The specific command for turning on the chosen terminals is shown in figure (4.26).

```
NOBSPROJECT.cr6 - ...\duwap\Downloads
NOBSPROJECT.cr6
76 StdDev (1,Direct_Normal_Irradiance,FP2,False)
77 EndTable
78
79 'Main Program
80 BeginProg
81 'setStatus("usrDriveSize", 8192)
82 SW12 (1,1)
83 SW12 (2,1)
84
85 Scan (1,Sec,0,0)
86   PanelTemp (PTemp,15000)
87   VoltDiff (Global_Horizontal_Irradiance,1,mV200,U1,False,500,6,Sens_P1ra1,0)
88   VoltDiff (Global_Normal_Irradiance,1,mV200,U3,False,500,6,Sens_P1ra2,0)
89   VoltDiff (Direct_Normal_Irradiance,1,mV200,U5,False,500,6,Sens_P1rh1,0)
90
91   'CallTable Test
92   'CallTable Irradiance_1sec
93   CallTable Irradiance_1min
94   CallTable Irradiance_5min
95   NextScan
96
97 'file_handle = FileOpen("CPU:GHI.csv", "a", -1)
98 'FileWrite (file_handle, "2010-05-20T11:01:43Z," + Global_Horizontal_Irradiance + CHR(10), 0)
99 'FileWrite (file_handle, Irradiance_1min, 0)
100 'FileClose (file_handle)
101
102 SlowSequence
103 Do
104   Delay(1,5,Min) 'kontrollerer tiden mellom hver skrivning til sky
105   'Create file named FTP_Tutorial_1.csv and append data to the file every 5 minutes
106   'FTPResult=FTPClient ("computer-name.domain.com", "Tutorial", "Tutorial_PW", "FTPTest", "FTP_Tutorial_1
107   'HTTPResult=HTTPPut ("https://lagringforskningsdata2.blob.core.windows.net/sunblob/aztest.txt?sv=2019-0
108   http_header = "x-ms-blob-type: Blob8lob"
109   'file_handle = FileOpen("CPU:Irradiance_1min.csv", "a",-1)
110   'FileWrite (file_handle, Global_Normal_Irradiance + Global_Horizontal_Irradiance + CHR(10), 0)
111   'FileClose (file_handle)
112   FileRename(LastFileName,USR:SendingFile)
113
114   'http_post_tx = HTTPPost ("https://iftsundata.blob.core.windows.net/sundatablob/sundata.txt?sv=2019-02-
115   HTTPResult=HTTPPut ("https://iftsundata.blob.core.windows.net/trackerdata/latest.csv?comp=appendblob&
```

Figure 4.26: SW command to turn on the terminals used for powering the router

The SW command has two arguments. The first one specifies which terminal is used. The second argument is either 1 or 0, where 1 is True, and 0 is False for turning on the power. This command must be used within the main program.

In this setup, an ethernet connection is used between the router and the datalogger. This yields an automatic setup, which does not require further changes in order for the datalogger to receive a stable connection.

4.3.3 Time and Date

Using the “Logger Control” tab, illustrated in figure (4.27), the current station and reference time can be observed. There is also an option to change the station time by clicking ”Reference Clock Setting”. The station is currently using UTC for compatibility purposes.

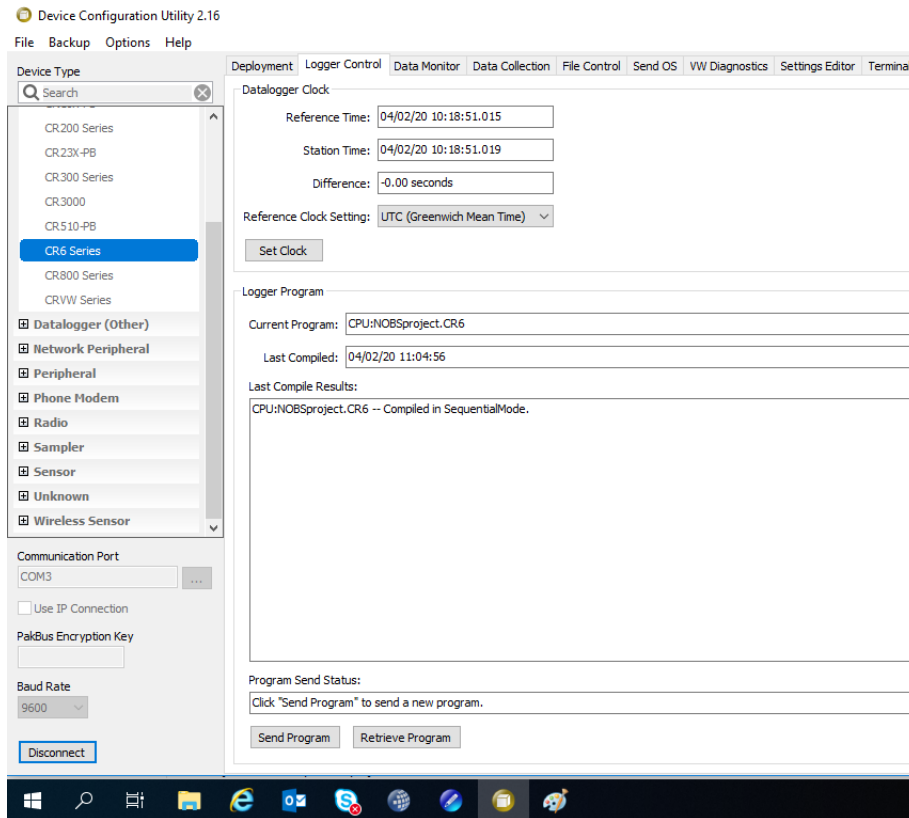
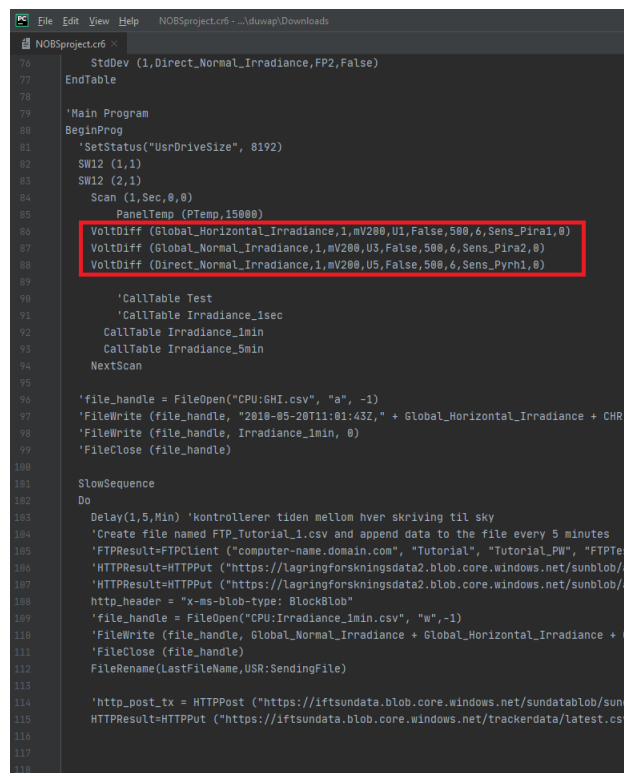


Figure 4.27: Time settings in Device Configuration Utility

4.3.4 CR6 Programming

The data logger uses CR6 programs to determine what tasks it should perform. Making these programs is the most comprehensive work of the project. The current program is taking measurements of DNI,GHI and GNI as voltage differences and uses a prefixed command called "VoltDiff" for calculation of these measurements. The variables used for each instrument can be found in figure (4.28), and came finished programmed by EKO.



```
File Edit View Help NOBProject.cr6 - ...\Downloads
NOBProject.cr6
76 StdDev (1,Direct_Normal_Irradiance,FP2,False)
77 EndTable
78
79 *Main Program
80 BeginProg
81 'setStatus("usrDriveSize", 8192)
82 SW12 (1,1)
83 SW12 (2,1)
84 Scan (1,Sec,0,0)
85 PanelTemp (PTemp,15000)
86 VoltDiff (Global_Horizontal_Irradiance,1,mV200,U1,False,500,6,Sens_Pira1,0)
87 VoltDiff (Global_Normal_Irradiance,1,mV200,U3,False,500,6,Sens_Pira2,0)
88 VoltDiff (Direct_Normal_Irradiance,1,mV200,U5,False,500,6,Sens_Pyrh1,0)
89
90 'CallTable Test
91 'CallTable Irradiance_1sec
92 CallTable Irradiance_1min
93 CallTable Irradiance_5min
94 NextScan
95
96 'file_handle = FileOpen("CPU:GHI.csv", "a", -1)
97 'FileWrite (file_handle, "2010-05-20T11:01:43Z," + Global_Horizontal_Irradiance + CHR(10))
98 'FileWrite (file_handle, Irradiance_1min, 0)
99 'FileClose (file_handle)
100
101 SlowSequence
102 Do
103 Delay(1,5,Min) 'kontrollerer tiden mellom hver skrivning til sky
104 'Create file named FTP_Tutorial_1.csv and append data to the file every 5 minutes
105 'FTPResult=FTPClient ("computer-name.domain.com", "Tutorial", "Tutorial_PW", "FTPTest")
106 'HTTPResult=HTTPPut ("https://lagringforskningsdata2.blob.core.windows.net/sunblob/az")
107 'HTTPResult=HTTPPut ("https://lagringforskningsdata2.blob.core.windows.net/sunblob/az")
108 http_header = "x-ms-blob-type: BlockBlob"
109 'file_handle = FileOpen("CPU:Irradiance_1min.csv", "w", -1)
110 'FileWrite (file_handle, Global_Normal_Irradiance + Global_Horizontal_Irradiance + CHR(10))
111 'FileClose (file_handle)
112 FileRename (LastFileName,USR:SendingFile)
113
114 'http_post_tx = HTTPPost ("https://iftsundata.blob.core.windows.net/sundatablob/sundatablob")
115 HTTPResult=HTTPPut ("https://iftsundata.blob.core.windows.net/trackerdata/latest.csv")
116
117
118
```

Figure 4.28: Prefixed voltage difference command in CRBasic

In order to collect these measurements in a data table, DataTable and CallTabler commands are used. At the beginning of the script, the DataTable function has to be called in order to define a data table. Furthermore, the DataInterval is used within the DataTable loop to define the interval between data table values. The program also has prefixed commands for finding the average, maximum and minimum within these intervals. These commands are called "Average", "Maximum" and "Minimum" respectively. Figure (4.29) shows an example of the programming used for the data table in this project.

```

DataTable(Irradiance_5min,1,-1)
DataInterval(0,5,Min,10)
TableFile("USR:Datatable5min",12,1,0,5,Min,OutStat,LastFileName)
Average(1,Global_Horizontal_Irradiance,IEEE4,False)
Maximum(1,Global_Horizontal_Irradiance,FP2,False,False)
Minimum(1,Global_Horizontal_Irradiance,FP2,False,False)
StdDev(1,Global_Horizontal_Irradiance,FP2,False)
Average(1,Global_Normal_Irradiance,IEEE4,False)
Maximum(1,Global_Normal_Irradiance,FP2,False,False)
Minimum(1,Global_Normal_Irradiance,FP2,False,False)
StdDev(1,Global_Normal_Irradiance,FP2,False)
Average(1,Direct_Normal_Irradiance,IEEE4,False)
Maximum(1,Direct_Normal_Irradiance,FP2,False,False)
Minimum(1,Direct_Normal_Irradiance,FP2,False,False)
StdDev(1,Direct_Normal_Irradiance,FP2,False)
EndTable

```

Figure 4.29: DataTable function used to create 5minute average datatable

For the data table to be sent to external storage, it has to be saved to the USR allocation in the datalogger. This saving can be done by using the TableFile command, used within the data table command shown in figure (4.29), in CRBasic. The "File Control" tab and "USR Drive" option can be used to check if the file is indeed saved, illustrated in figure (4.30). In order to not exceed the memory of the USR, the number of saved data tables can be reduced. The easiest practice is only to allow one single file to be stored at an instance.

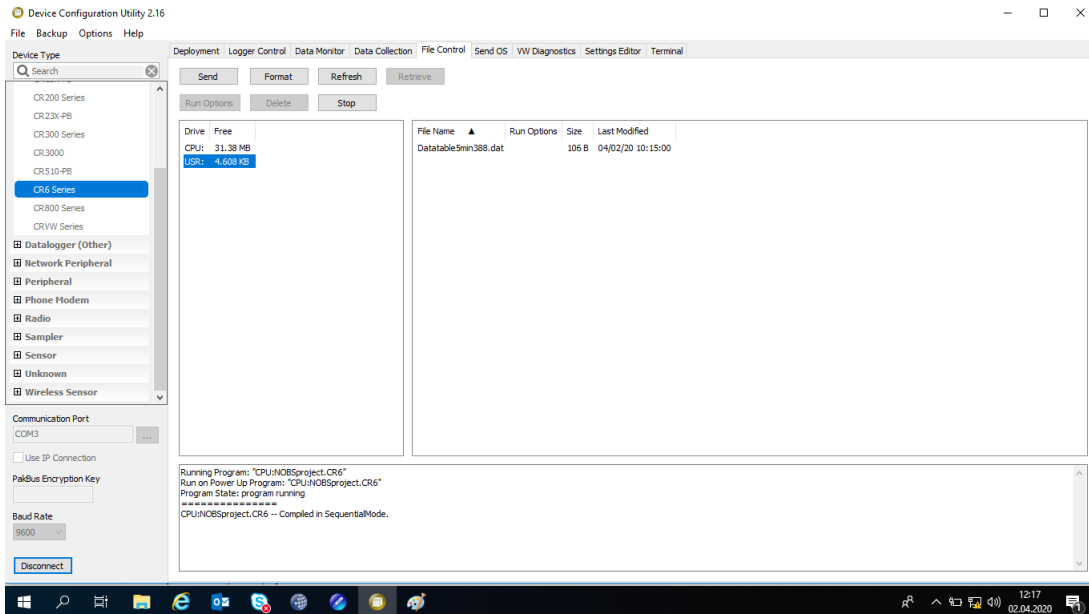


Figure 4.30: USB directory in the CR6 Datalogger

The Campbell Scientific supports several methods for sending wanted data from the datalogger (Campbell Scientific, u.d.). These include methods such as FTP Streaming and HTTPPut/HTTPGet. Due to security reasons, HTTPPut command is used to send the data from the datalogger to a virtual storage location at the University of Tromsø every 5 minutes. The reasoning behind the 5minute interval for sending is to resend every 5 minutes, due to a new table being created at the same sampling rate. The interval of sending data can easily be changed in the HTTPPut command. The service of storing data at ITA is valid to 10th of March 2025 and have to be revised at this point to continue. More information on different methods of sending data is available at <https://www.campbellsci.com/cr6>.

4.3.5 Public Variables

The data logger is also instructed with a “Data Monitor” tab. The status on public variables such as the different measurement and timestamp can be found here. This is especially useful for identifying the HTTP Result. This indicates whether the HTTPPut function manages to transfer the file. A number higher than 100 indicates that a file has successfully been trans-

ferred to the wanted location. These variables are found under “Public” in the “Data Monitor” tab. seen in figure (4.31).

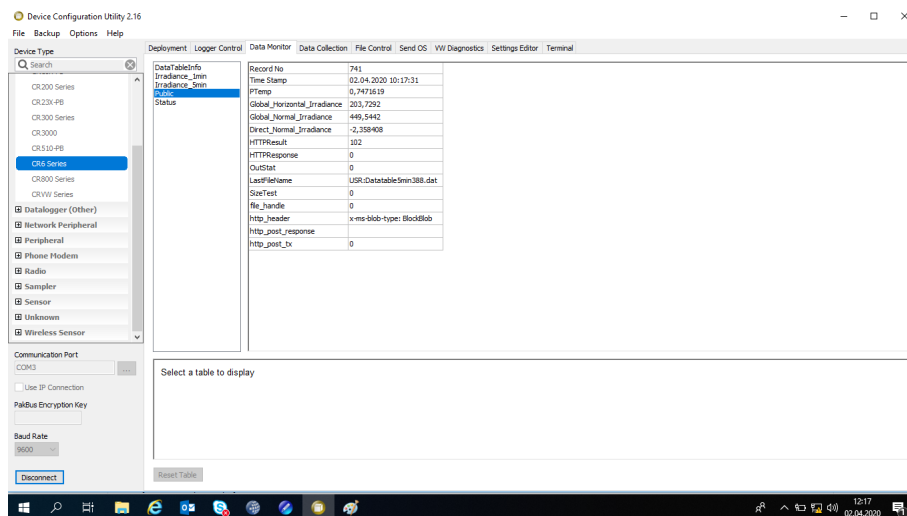


Figure 4.31: Public Variables directory in Device Configuration Utility

Included in the other subtabs are the different data tables that have been programmed. These tabs can be used to click into and check how the current data table looks and can be used to export it to the computer that is connected through USB. The data table is illustrated in figure (4.32).

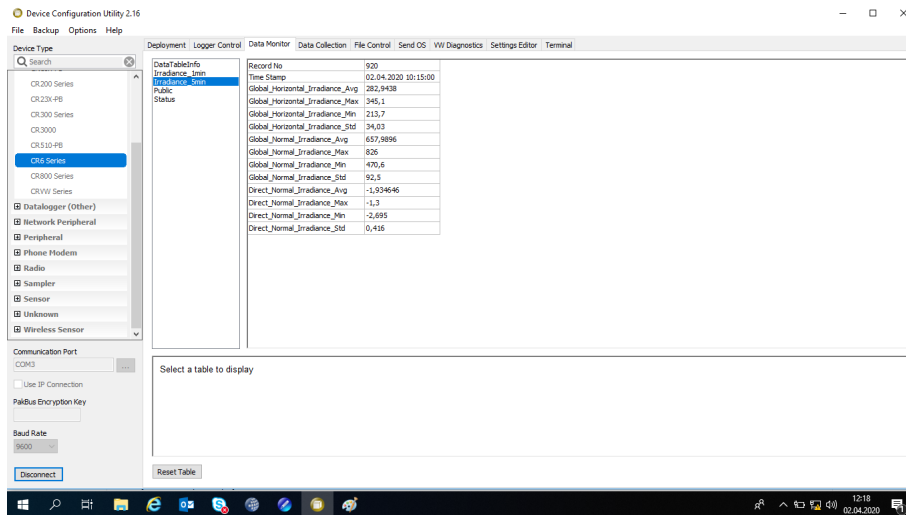


Figure 4.32: Illustration of the data table in device configuration utility

4.3.6 Uploading CR6 Program

In order for the CRBasic program to run on the logger, it has to be uploaded and started manually. The “File Control” tab is used for this. Under the CPU subtab, all the currently uploaded programs are shown, illustrated in figure (4.33). It also has a “Run Options” tab, where the status on the given program is given. In order to upload a new program to the CPU, the “Send” button is used. After uploading the program with Send button, the run option for the given program needs to be modified. The most useful setting is “Run on Power Up”. This option causes the program to run automatically. If the program is unable to run due to any faults in the CRBasic code, an error message will be shown in the Console window at the bottom of the screen.

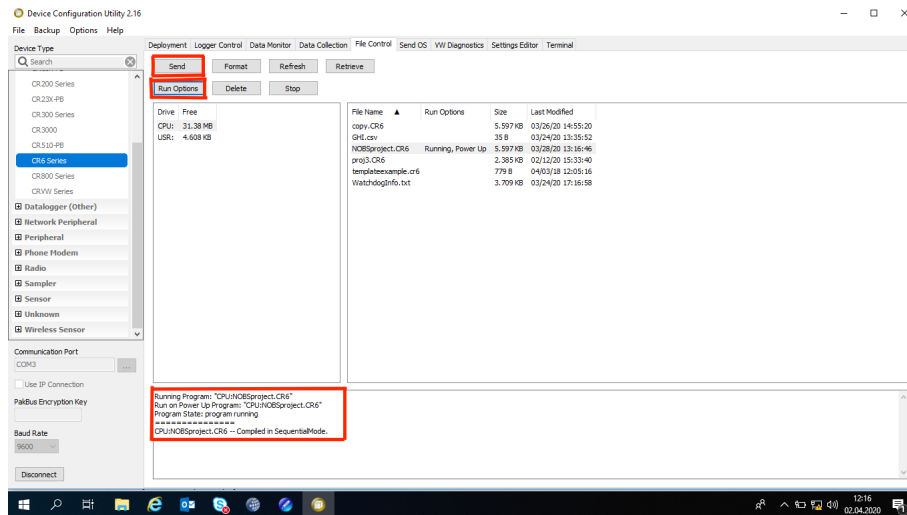


Figure 4.33: Running and uploading a new program to the CR6 Datalogger

4.3.7 Azure Storage

Azure Storage is the preferred storage location for the University of Tromsø. In order to gain access to measurement data, access has to be given by the ITA department at the University. The data is located in “Archives” under iftsundata in Azure Storage Application, illustrated in figure (4.34). The data is labelled with dates and contains the measurements with 5minute intervals of that whole day. In order to change the sampling rate, the second and third parameter in DataTable command can be used. In figure (4.29) the datatable has the second parameter as 5, and the third as Min, resulting in 5-minute intervals.

A plot of the measurement of the day can also be found at <http://suntracker.azure.uit.no/>, with a small delay. It is also important to note that the time in the given plot is UTC and not local time. The plot for a given date is updated at approximately 02:00 AM and uploaded to the archives.

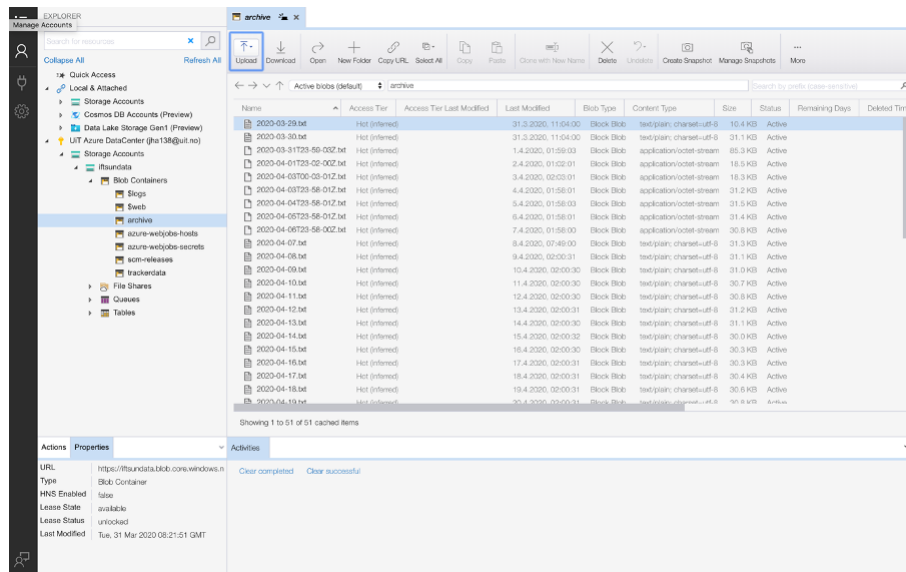
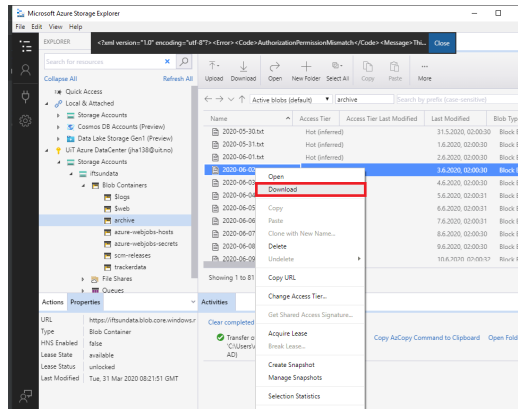
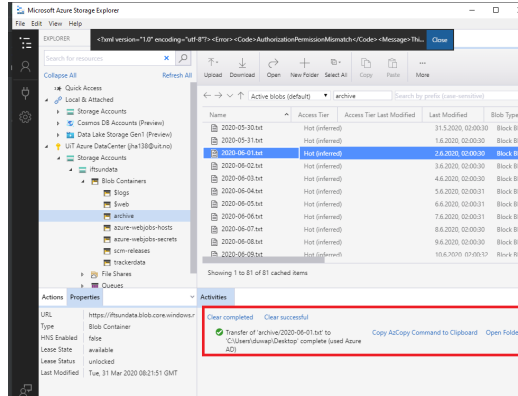


Figure 4.34: Layout of Azure Storage Software for accessing the files from the sun tracker

In order to download the files from Azure Storage, the download function in the program is utilized by right-clicking the given file to download. The "activity" window will provide information about whether the download was successful or not, and the location of the downloaded file. Both of the functions are shown in figure (4.35).



(a) Download File function



(b) Activity Window

Figure 4.35: Specific commands for downloading a file from Azure Storage

4.4 Work Distribution

Several parties have been included in the setup of the sun tracker system. EKO Instruments, the company behind the sun tracker, did parts of the programming in CRBasic. EKO made an example file of how to make a data table in CRBasic and the specific command for which terminal was to be used in the different "VoltDiff" commands. The fixed sensitivities for the given pyranometers and pyrhelimeter used in the programming part was obtained from EKO. The setup of the Datalogger terminals with cables fixed for using on the different measurement devices, such as the cable for measuring DNI,

was prearranged as well.

The programming of the wanted data tables, interval between streaming the measurements to Azure storage, was handled by myself in cooperation with the ITA department of UiT. The conversion of the streamed datasets to the website, <http://suntracker.azure.uit.no/>, and storage of the datasets was set up by Per Ivar Emmanulesen and Rolf Andersen at the University of Tromsø. All other work, including the assembly of the sun tracker itself, cabling, calibration and finding a location, was done by myself as a part of this master thesis project.

Chapter 5

Final discussion and conclusions

This thesis is analyzing the solar irradiance in Tromsø, at 69.6 degrees latitude, and comparing it with a few central and middle Europe locations. The hypothesis is that the atmosphere is less particle dense due to less pollution in Tromsø. The sunlight could therefore be less affected by the atmospheric effects such as scattering and absorption due to particles. The reason for investigating this is to gain a better understanding of the energy potential at higher latitudes, and further to utilize this for better extraction of solar energy.

In order to investigate the atmospheric effects on the sunlight in Tromsø, a two-axis sun tracker was installed at Nordlysobservatoriet in Tromsø. The measurements were collected, and in cooperation with ITA at UiT, a website has been created to visualize the daily plots of Global Normal Irradiance, Global Horizontal Irradiance and Direct Normal Irradiance. A daily plot of the measurements is found at <http://suntracker.azure.uit.no/>. Baseline Surface Radiation Network provides measurements from four different stations of variable latitude, and climate (Driemel, 2019). A diffuse and clearness index analysis has been performed. The reduction from extraterrestrial radiation to Global Horizontal Irradiance and Direct Normal Irradiance have been established for all stations aforementioned. All datasets from BSRN was in 1-second intervals, while the measurements from Tromsø were of 5-minute intervals originally. All datasets are resampled to mean values of 5-minute intervals and filtered to remove faulty or wrong measurements. These mea-

surements could either be affected by the cloud enhancement phenomenon or for being measured when the solar elevation angle was below 5 degrees. This chapter provides a final discussion of the methodology and results in this thesis.

Section (4.1) give the results from picking out days with cloudless condition from daily plots of GHI, GNI and DNI measurements in May. The reduction of GHI and DNI is analyzed with a 5-degree interval of solar elevation angle. Iznas measurement station is located 2373m above sea level, and the distance of the atmosphere is, therefore, lower between the start of the atmosphere and the measurement sensors. This height above sea level has to be compensated for in order to investigate the atmospheric effects. Using equation (2.8) to Iznas at 2373m and Cener at 471m above sea level, a new compensated reduction is produced. Subsection (4.1.4) shows the result of this compensation, and after adding extra atmosphere to Iznas and Cener. The method of compensating for this height difference is not entirely true, due to the assumption of a homogeneous spherical atmosphere model. Using the scale height formula found in Appendix A, the result from using the homogenous spherical atmosphere model is close and considered viable for the analysis in this thesis.

5.1 Reduction of Global Horizontal Irradiance and Direct Normal Irradiance

The results show a higher reduction of Global Horizontal Irradiance for all external stations used in this thesis. The GHI reduction is decreasing linearly with solar elevation angle, and Tromsø have the lowest reduction from extraterrestrial irradiance to global horizontal irradiance for all solar elevation angles. The result shows an increase in GHI reduction with a mean value of 3-5%. The reduction is increasing with the solar elevation angles, resulting in a higher reduction at higher solar elevation angles. At the lower angles, the reduction is high, and the difference is not as apparent.

The reduction of Direct Normal Irradiance decreases exponentially with the solar elevation angle. In Tromsø, the Global Horizontal Irradiance reduction at 40 degrees solar elevation angle is approximately 30-40%, while the reduc-

tion of Direct Normal Irradiance is 0-20% for the same solar elevation angle. The difference in the reduction of DNI varies with the solar elevation angle but is in total higher for Tromsø. The reduction of DNI from extraterrestrial irradiance has an increased mean value of 10% higher at the higher solar elevation angles for Tromsø compared to external stations.

Looking at the result of compensation of height above sea level for Izana and Cener, the result seems to fit well with the results from the other stations. The result from compensations could be used as a basis for comparing the reduction of both GHI and DNI. Since the thesis only includes data from May, all stations except Izana have a small dataset of days with cloudless conditions. It is therefore hard to come to any conclusion from this small dataset, but the behaviour is shown. The measurements from Tromsø can only be used from 29th of April and forward, and it was attempted to find clear days in June in order to compensate for the low number. This was not successful as no days were matching, and it was decided not to take more months into account at the other locations.

The result of the reduction calculations supports the hypothesis of higher irradiance in Tromsø due to a less particle dense atmosphere. The global irradiance consists of both diffuse and direct irradiance. The reduction of direct normal irradiance is higher in Tromsø, which would lead to more global irradiance being diffuse irradiance. If the pyrhelimeter is not perfectly directed at the sun, the measurement will also have an error which will affect the result. If not correctly levelled, the pyrhelimeter is more prone to erroneous measurements than the pyranometer. The pyrhelimeter only has a 5° acceptance angle, which makes it more sensitive to imperfect measurements. It has been observed that the sun tracker have to be re-levelled depending on the snow condition. This un-levelling is due to the type of surface the roof is made of, and therefore the pyrhelimeter might give some bad measurements if not maintained routinely. Another influencing factor in the direct irradiance is the particle size of aerosols. An atmosphere containing a large number of smaller particles scatters more than an atmosphere containing an equivalent mass of larger particles (Brine, 1983).

(Enoksen, 2020) found that when using real measurement data from Svalbard, located at 79.0° latitude, as an input for solar energy simulation program PVSyst, it did not accept the data due to having too high irradiance

measurements. This finding is in line with the results of this thesis, showing an increased GHI irradiance for high latitude locations.

5.2 Diffuse and Clearness Index

From section (4.2), the result shows that Skartveit&Olseth model overestimates the clearness index compared to the measured index in Tromsø. Especially for lower diffuse indexes, the clearness index is increasingly different. For comparison between the datasets, the model only uses measurement data from May as input, which is a small data set. The Skartveit&Olseth model is based on hourly-averaged data consisting of several years worth of datasets. It could lead to the difference seen, and comparing bigger data sets might give other results.

The result takes all days into account, and show an apparent increase in the diffuse index for Tromsø compared to the other external locations. This implies that there is a higher diffuse irradiance component for a given clearness index. The error could occur due to a slight fault in the levelling of the pyrliometer concerning the solar angle. From the result of the S&O model with May month as input, the model seems to correspond poorly to the actual clearness and diffuse index. The albedo corrected model also uses an albedo value as input. The value is estimated from the surface cover, and using a poorly estimated surface albedo could occur. The Skartveit&Olseth model is therefore not used to any extent. Instead, the comparison between the external stations and Tromsø is used in the thesis.

5.3 Final Summary

The resulting comparison plots show a clear indication that the global irradiance is higher at Tromsø, compared to other european locations. From the DNI plots, it is clear that the reduction of DNI is higher for all solar elevation angles in Tromsø. Due to the pyrliometers increased sensitivity of levelling, there is a higher chance that the DNI measurements in Tromsø are incorrect. The increased global horizontal irradiance in Tromsø could be due to an atmosphere with fewer particles.

The result from running the Skartveit&Olseth model for diffuse and clearness index in Tromsø shows a low ability to predict these for May month. The model overestimates the clearness index, and the overall results do not match with the result from the real measurements.

5.4 Further work

During this thesis, a sun tracker has been installed at Nordlysobservatoriet in Tromsø, where the measurements have been used to investigate the atmospheric effects in Tromsø compared to middle and central Europe. There are several possibilities to further work on this basis. Some suggestions are:

- **Update location of the system**

The roof the sun tracker is located and need re-levelling for different snow conditions. There is a possibility that a measurement platform is being installed at Nordlysobservatoriet by UiT. The measurement platform is a preferred location, and there is required less maintenance of the equipment due to stable conditions. The whole setup needs to be reassembled and calibrated for this location. The GNI mounting plate did not fit with the MV-01 ventilation/heating unit preventing snow and ice to form. EKO has been informed and tare working on a solution.

- **Investiagtion over a longer period**

The dataset used in the thesis is only from the start of May to the start of June. In order to get a better analysis of the measurement data, it is preferred to analyse measurement over a longer period. A future assignment could be to investigate further the reduction of GHI and DNI, including more months.

Chapter 6

Bibliography

Bertrand, C., Vanderveken, G. and Journée, M., 2015. Evaluation of decomposition models of various complexity to estimate the direct solar irradiance over Belgium. *Renewable Energy*, pp.618-626.

Blanc, P., Espinar, B., Gauder, N. et al., 2014, Direct Normal Irradiance Related Definitions and Applications: The Circumsolar Issue. *Solar Energy*, 110, pp.561-577.

Bowden, S. Honsberg, C., 2019, Air Mass [Online]
url: <https://www.pveducation.org/pvcdrom/properties-of-sunlight/air-mass>

Bowden, S. Honsberg, C., 2019, Atmospheric Effects [Online]
url: <https://www.pveducation.org/pvcdrom/properties-of-sunlight/atmospheric-effects>

Bowden, S. Honsberg, C., 2019, Energy of Photon [Online]
url: <https://www.pveducation.org/pvcdrom/properties-of-sunlight/energy-of-photon>

Bowden, S. Honsberg, C., 2019, Properties of Light [Online]
url: <https://www.pveducation.org/pvcdrom/properties-of-sunlight/properties-of-light>

Brine, D.T., Iqbal M., 1983, Diffuse and Global Solar Spectral Irradiance

Under Cloudless Skies, Solar Energy, pp. 447-453

Böhme, L., 2019, Modelling Diffuse Irradiance at High Latitudes, Biele University

Campbell Scientific, u.d., Data Loggers [Online]
url: <https://www.campbellsci.com/data-loggers>

Campbell Scientific, u.d., CR6 [Online]
url: <https://www.campbellsci.com/cr6>

Duffie, J. A. Beckman, W. A., 2013, Solar Engineering of Thermal Processes, Fourth Edition, Hobouken, New Jersey, pp. 44-53

Driemel, A., Augustine, J., Behrens, K. et al., 2018, Baseline Surface Radiation Network (BSRN): Structure and Data Description (1992–2017). Earth System Science Data

EKO Instruments, u.d., MS-80 Pyranometer [Online]
url: <https://eko-eu.com/products/solar-energy/pyranometers/ms-80-pyranometer>

EKO Instruments, u.d., MS-57 Pyrheliometer [Online]
url: <https://eko-eu.com/products/solar-energy/pyrheliometers/ms-57-pyrheliometer>

EKO Instruments, u.d., STR-22G Sun Tracker [Online]
url: <https://eko-eu.com/products/solar-energy/sun-trackers/str-22g-sun-trackers>

EKO Instruments, u.d., Re-calibration Interval of Solar Sensors [Online]
url: <https://eko-eu.com/support/faq/maintenance/>

Enoksen, T.O., 2020, Evaluation of a Solar Power Plant at Longyearbyen, UiT - The Arctic University of Norway

Eikeland, O.D., 2019, Investigation of Photovoltaic Energy Yield on Tromsøya by Mapping Solar Potential in ArcGIS, UiT – The Arctic University of Norway

Garner, R., 2008, Solar Irradiance [Online]

url: https://www.nasa.gov/mission_pages/sdo/science/solar-irradiance.html

Hinckley, A., 2017, Pyranometers: What you need to know [Online]
url: <https://www.campbellsci.com/blog/pyranometers-need-to-know>

Hum, S.V., Atmospheric Effects, Radio and Microwave Wireless Systems [Online]
url: <http://www.waves.utoronto.ca/prof/svhum/ece422/notes/20b-atmospheric.pdf>

ISO, u.d., ISO 9000 family – Quality Management [Online]
url: <https://www.iso.org/iso-9001-quality-management.html>

Jacovides, C., Kallos, G., Steven, M., 1993, Spectral Band Resolution of Solar Radiation in Athens, Greece. International Journal Of Climatology, 689-697

Kipp Zonen, 2015, The working principle of a thermopile pyranometer [Online]
url: <https://www.kippzonen.com/News/572/The-Working-Principle-of-aThermopile-Pyranometer>

Kipp Zonen, 2018, Calibration Standards [Online]
url: <https://www.kippzonen.com/ProductGroup/112/Calibration-Standards>

Li, H., 2016, Pavement Materials For Heat Island Mitigation, Oxford: Butterworth-Heinemann.

Merchant, E. F., (2018), IPCC: Renewables to supply 70% of electricity by 2050 to avoid worst impacts of climate change [Online], url: <https://www.greentechmedia.com/articles/read/ipcc-renewables-85-electricity-worst-impacts-climate-change>

Multiconsult, 2018, Solkraft Løfter Norge inn i Framtiden [Online]
<https://www.multiconsult.no/solkraft-lofter-norge-inn-i-framtiden/>

Sterken, C., Manfroid, J., 1992, Astronomical Photometry, Dordrecht: Kluwer

Solanki, C.S., 2015, Solar Photovoltaics: Fundamentals, Technologies and Applications, Third Edition, Prentice-Hall of India

Soluzione Solare, u.d., Thermopile Pyranometers [Online]

url: <https://www.solarwind-sensor.com/pyranometers/thermopile-pyranometers/>

UiT, 2019, ARC- Arctic Centre for Sustainable Energy [Online]

url: https://en.uit.no/forskning/forskningsgrupper/gruppe?p_document_id=453700

Yordanov, G., Midtgård, O., Saetre, T. et al., 2013, Overirradiance (Cloud Enhancement) Events at High Latitudes, IEEE Journal of Photovoltaics, pp.271-277.

Chapter 7

Appendices

7.1 Appendix A

Taking height into account, scale height H can be used to find a scale of decrease in air mass due to increase in height. Assuming uniform temperature and composition, the density can be treated proportional to pressure.

$$P(x) = P_0 \exp(-x/H) \quad (7.1)$$

where x is the height and H is the scale height. Normalizing the sea level to 1 yields that the relative height air mass can be expressed as

$$AM_{ratio} = \exp(-x/H) \quad (7.2)$$

where H is approximately 8000m.

7.2 Appendix B

All python codes used for the analysis. Due to different format of the documents, several scripts for reading and resampling the data is needed. A script for the skartveit model is also given, as well as a script for analyzing the data gained.

```
#Script to convert .txt file from NOBS into plot of the  
→ measurements
```

```

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import csv

def dailyplot(filename):
    #reading given filename using pandas
    #using function to easily call for another file to
    → import
    data = pd.read_csv(filename, sep= ",")

    #making the different header columns and adds the given
    → values to it, i_dn = direct normal, i_gn = global
    → normal, i_gh = global horizontal
    #all of the above uses the average value

    current_date = 'NO DATA'
    if len(data) != 0:
        current_date = data['TIMESTAMP'][0][0:10]
        data.TIMESTAMP = pd.DatetimeIndex(data.TIMESTAMP)
        data = data.rename(columns={'TIMESTAMP': 'Timestamp'})
        doy = data['Timestamp'].dt.dayofyear
        doy = doy.values.tolist()
        data = data.set_index('Timestamp')
        data.index = data.index.hour + data.index.minute / 60
        # return timestamp for further use
        time_utc = pd.Index.to_numpy(data.index)

    print(doy)
    #adding the different measurements into arrays
    i_gh = np.array(data['Global_Horizontal_Irradiance_Avg'])
    i_gn = np.array(data['Global_Normal_Irradiance_Avg'])
    i_dn = np.array(data['Direct_Normal_Irradiance_Avg'])

    #plotting data
    plt.figure(1)
    plt.plot(data.index,i_gh,color='red',label='Global
    → Horizontal')

```

```

plt.plot(data.index,i_gn,color='blue',label='Global
→ Normal')
plt.plot(data.index,i_dn,color='black',label='Direct
→ Normal')
plt.title(current_date)
plt.xlabel('Timestamp')
plt.ylabel('Irradiance(W/m^2)')
plt.legend()
plt.show()
return time_utc,i_gh,i_gn,doy,i_dn

def exradcalc(time_utc,i_gh,i_gn,i_dn,doy):
    # extraterrestrial radiation constant
    I_exsc = 1367 #W/m^2
    length=len(time_utc)
    b=np.zeros(length)
    R= np.zeros(length)
    I_ex = np.zeros(length)

    for i in range(0,length):
        b[i] = (2 * np.pi / 365) * doy[i]
        R[i] = 1.00011 + 0.34221 * np.cos(b[i]) + 0.00128 *
        → np.sin(b[i])\
            + 0.000719 * np.cos(2 * b[i])\
            + 0.000077 * np.sin(2 * b[i])
        I_ex[i] = I_exsc * R[i]

    #reading given
    #calculating the reduction in radiation from
    → extraterrestrial to global

    I_reduction_gh =(1- i_gh/I_ex)
    I_reduction_gn = (1-i_gn/I_ex)
    I_reduction_dn = (1-i_dn/I_ex)

    return I_reduction_gh, I_reduction_gn,I_reduction_dn,
    → I_ex

```



```

def
→ solarangle(doy,time_utc,igh,idn,latitude_deg,longitude_deg):
    length = len(time_utc)
    altitude_deg = np.zeros(length)
    zenith_deg = np.zeros(length)
    zenith_rad=np.zeros(length)
    latitude_rad = np.deg2rad(latitude_deg)
    LSTM = np.zeros(length)
    time_local = np.zeros(length)
    TC = np.zeros(length)
    LST = np.zeros(length)
    HRA = np.zeros(length)
    HRA_rad = np.zeros(length)
    altitude_rad =np.zeros(length)
    B=np.zeros(length)
    EoT=np.zeros(length)
    dec_angle=np.zeros(length)
    dec_angle_rad=np.zeros(length)
    i_dhi=np.zeros(length)

    for i in range(length):
        #local standard time meridian
        delta_utc = 1
        LSTM[i] = 15*delta_utc
        time_local[i] = time_utc[i] + delta_utc

        #Equation of Time
        # EOT - np.sin uses radians as argument
        B[i] = (360/365)*(doy[i]-81)
        EoT[i] = 9.87 * np.sin(np.deg2rad(2 * B[i]))\
                - 7.53 * np.cos(np.deg2rad(B[i])) \
                -1.5 * np.sin(np.deg2rad(B[i]))

        #time correction
        TC[i] = 4*(longitude_deg - LSTM[i]) + EoT[i]

        #Local Solar Time

```

```

LST[i] = time_local[i] + TC[i]/60

#Hour Angle
HRA[i] = 15*(LST[i]-12)

HRA_rad[i] = np.deg2rad(HRA[i])

dec_angle[i] = 23.45*np.sin(np.deg2rad(B[i]))
dec_angle_rad[i] = np.deg2rad(dec_angle[i])

altitude_rad[i] = np.arcsin(np.cos(latitude_rad)
                            *np.cos(dec_angle_rad[i])
                            *np.cos(HRA_rad[i])
                            +np.sin(latitude_rad)
                            *np.sin(dec_angle_rad[i]))
altitude_deg[i] = np.rad2deg(altitude_rad[i])
zenith_deg[i] = 90-altitude_deg[i]
zenith_rad[i] = np.deg2rad(zenith_deg[i])
i_dhi[i] = igh[i] - idn[i] * np.cos(zenith_rad[i])

return altitude_deg,i_dhi

def anglefinder(i_gh, i_dn, altitude_deg, I_reduction_gh,
→ I_reduction_gn, I_reduction_dn):
loc= np.zeros(len(altitude_deg))
alt_list=[]
altloc_list=[]
red_gh_list=[]
red_gn_list=[]
red_dn_list=[]
i_gh_list=[]
i_dn_list=[]

for k in range(1,10):
    for i in range(len(altitude_deg)):
        if altitude_deg[i] < (5*k + 0.25) and
→ altitude_deg[i] > ((5*k)-0.25):
            loc[i] = 1

```

```

        alt_list.append(altitude_deg[i])
        altloc_list.append(i)
        red_gh_list.append(I_reduction_gh[i])
        red_gn_list.append(I_reduction_gn[i])
        red_dn_list.append(I_reduction_dn[i])
        i_gh_list.append(i_gh[i])
        i_dn_list.append(i_dn[i])

    #plt.figure(2)
    #plt.scatter(alt_list, red_list)
    #plt.title('REDUCTION')
    #plt.xlabel('Solar Altitude')
    #plt.ylabel('Reduction')
    #plt.show()

    return
    ↪ i_gh,i_dn,alt_list,red_gh_list,red_gn_list,red_dn_list

def writedata(filename2, time_utc, day, i_gh, i_gn, i_dhi,
    ↪ i_dni, iex, alt, I_gh_reduction, I_gn_reduction,
    ↪ I_dn_reduction, firstfile=0):
    with open(filename2 + '_filtered.csv', 'a', newline='')
        ↪ as csvfile:
            fields = ['time', 'doy', 'igh', 'ign', 'idhi', 'idni',
                ↪ 'i_gh_reduction', 'i_gn_reduction',
                ↪ 'i_dn_reduction', 'iex', 'alt']
            writer = csv.DictWriter(csvfile, fieldnames=fields)
            if firstfile==1:
                writer.writeheader()
                for i in range(len(time_utc)):
                    writer.writerow(

```

```

        {'time': time_utc[i], 'doy': day[i],
         ↪ 'igh':
         ↪ i_gh[i], 'ign':i_gn[i], 'idhi':i_dhi[i],
         ↪ 'idni': i_dni[i], 'i_gh_reduction':
         ↪ I_gh_reduction[i], 'i_gn_reduction':
         ↪ I_gn_reduction[i], 'i_dn_reduction':
         ↪ I_dn_reduction[i], 'iex': iex[i],
         ↪ 'alt': alt[i]})
    else:
        for i in range(len(time_utc)):
            writer.writerow(
                {'time': time_utc[i], 'doy': day[i],
                 ↪ 'igh': i_gh[i], 'ign':i_gn[i], 'idhi':
                 ↪ i_dhi[i], 'idni': i_dni[i],
                 ↪ 'i_gh_reduction':
                 ↪ I_gh_reduction[i], 'i_gn_reduction':
                 ↪ I_gn_reduction[i], 'i_dn_reduction':
                 ↪ I_dn_reduction[i], 'iex': iex[i],
                 ↪ 'alt': alt[i]})

def writedataclear(filename2,i_gh,i_dn,
↪ alt_list,red_gh_list,red_gn_list,red_dn_list,firstfile=0):
    with open(filename2 +
        ↪ '_altitudereduction.csv','a',newline='') as csvfile:
        fields = ['i_gh', 'i_dn','altitude' , 'gh_reduction',
        ↪ 'gn_reduction', 'dn_reduction']
        writer = csv.DictWriter(csvfile,fieldnames=fields)
        if firstfile==1:
            writer.writeheader()
            for i in range(len(alt_list)):
                writer.writerow({'i_gh': i_gh[i], 'i_dn':
                    ↪ i_dn[i], 'altitude': alt_list[i],
                    ↪ 'gh_reduction':red_gh_list[i],
                    ↪ 'gn_reduction': red_gn_list[i],
                    ↪ 'dn_reduction': red_dn_list[i]})
        else:
            for i in range(len(alt_list)):

```

```

        writer.writerow({'i_gh': i_gh[i], 'i_dn':
            ↪ i_dn[i], 'altitude':
            ↪ alt_list[i], 'gh_reduction':
            ↪ red_gh_list[i],
            ↪ 'gn_reduction': red_gn_list[i],
            ↪ 'dn_reduction': red_dn_list[i]})

def filtertromso(time_utc, day, i_gh, i_dni, i_dhi, iex,
    ↪ I_gh_reduction, I_gn_reduction, I_dn_reduction, alt):
    #filtering out values for ig < 5
    for i in reversed(range(len(time_utc))):
        if i_gh[i] < 5:
            i_gh = np.delete(i_gh, i)
            time_utc = np.delete(time_utc, i)
            day = np.delete(day, i)
            I_gh_reduction = np.delete(I_gh_reduction, i)
            I_gn_reduction = np.delete(I_gn_reduction, i)
            I_dn_reduction = np.delete(I_dn_reduction, i)
            iex = np.delete(iex, i)
            alt = np.delete(alt, i)
            i_dhi = np.delete(i_dhi, i)
            i_dni = np.delete(i_dni, i)

    #filtering out all values below 5 degrees
    for i in reversed(range(len(time_utc))):
        if alt[i] < 5:
            i_gh = np.delete(i_gh, i)
            time_utc = np.delete(time_utc, i)
            day = np.delete(day, i)
            I_gh_reduction = np.delete(I_gh_reduction, i)
            I_gn_reduction = np.delete(I_gn_reduction, i)
            I_dn_reduction = np.delete(I_dn_reduction, i)
            iex = np.delete(iex, i)
            alt = np.delete(alt, i)
            i_dhi = np.delete(i_dhi, i)
            i_dni = np.delete(i_dni, i)

```

```

for i in reversed(range(len(time_utc))):
    if i_dhi[i]/iex[i] > 0.8:
        i_gh = np.delete(i_gh,i)
        time_utc =np.delete(time_utc,i)
        day=np.delete(day,i)
        I_gh_reduction = np.delete(I_gh_reduction, i)
        I_gn_reduction = np.delete(I_gn_reduction, i)
        I_dn_reduction = np.delete(I_dn_reduction,i)
        iex=np.delete(iex,i)
        alt=np.delete(alt,i)
        i_dhi=np.delete(i_dhi,i)
        i_dni=np.delete(i_dni,i)

return time_utc, day, i_gh, i_dni, i_dhi, iex,
    ↪ I_gh_reduction, I_gn_reduction, I_dn_reduction, alt

def analysisandsave(filename, clear, firstfile_clear=0,
    ↪ firstfile_all=0):
    [timestamp,i_gh,i_gn,doy,i_dni] = dailyplot(filename)
    [I_gh_reduction, I_gn_reduction,I_dn_reduction, I_ex] =
    ↪ exradcalc(timestamp,i_gh,i_gn,i_dni,doy)
    [altitude_deg,i_dhi] = solarangle(doy, timestamp, i_gh,
    ↪ i_dni, latitude_deg=69.65, longitude_deg=18.96)
    [timestamp, doy, i_gh, i_dni, i_dhi, I_ex, I_gh_reduction,
    ↪ I_gn_reduction, I_dn_reduction, altitude_deg] =
    ↪ filtertromso(timestamp, doy, i_gh, i_dni, i_dhi, I_ex,
    ↪ I_gh_reduction, I_gn_reduction, I_dn_reduction,
    ↪ altitude_deg)
    if clear==1:
        [i_gh_angle, i_dn_angle, alt_list, red_gh_list,
        ↪ red_gn_list, red_dn_list] = anglefinder(i_gh,
        ↪ i_dni, altitude_deg, I_gh_reduction,
        ↪ I_gn_reduction, I_dn_reduction)
        writedataclear('NOBS', i_gh_angle, i_dn_angle,
        ↪ alt_list, red_gh_list, red_gn_list, red_dn_list,
        ↪ firstfile=firstfile_clear)

```

```

writedata('NOBS_alldays', timestamp, doy, i_gh, i_gn,
  → i_dhi, i_dni, I_ex, altitude_deg, I_gh_reduction,
  → I_gn_reduction, I_dn_reduction,
  → firstfile=firstfile_all)

#return timestamp, doy, i_gh, iex, i_reduction,
  → altitude_deg, altitude_list, reduction_list

def analysisandsavefiles():
  #scanning all days from NOBS, only first file is
  → firstfile_clear and firstfile_all to make the list
  → (30.april was a clear day luckily)
  #change clear according to if the day was clear or
  → cloudy
analysisandsave('NOBS/2020-04-30.txt', clear=1,
  → firstfile_clear=1, firstfile_all=1)
analysisandsave('NOBS/2020-05-01.txt', clear=0)
analysisandsave('NOBS/2020-05-02.txt', clear=0)
analysisandsave('NOBS/2020-05-03.txt', clear=0)
analysisandsave('NOBS/2020-05-04.txt', clear=0)
analysisandsave('NOBS/2020-05-05.txt', clear=0)
analysisandsave('NOBS/2020-05-06.txt', clear=0)
analysisandsave('NOBS/2020-05-07.txt', clear=0)
analysisandsave('NOBS/2020-05-08.txt', clear=0)
analysisandsave('NOBS/2020-05-09.txt', clear=0)
analysisandsave('NOBS/2020-05-10.txt', clear=0)
analysisandsave('NOBS/2020-05-11.txt', clear=0)
analysisandsave('NOBS/2020-05-12.txt', clear=0)
analysisandsave('NOBS/2020-05-13.txt', clear=0)
analysisandsave('NOBS/2020-05-14.txt', clear=0)
analysisandsave('NOBS/2020-05-15.txt', clear=0)
analysisandsave('NOBS/2020-05-16.txt', clear=0)
analysisandsave('NOBS/2020-05-17.txt', clear=0)
analysisandsave('NOBS/2020-05-18.txt', clear=0)
analysisandsave('NOBS/2020-05-19.txt', clear=0)
analysisandsave('NOBS/2020-05-20.txt', clear=0)
analysisandsave('NOBS/2020-05-21.txt', clear=0)
analysisandsave('NOBS/2020-05-22.txt', clear=1)

```

```
analysisandsave('NOBS/2020-05-24.txt', clear=0)
analysisandsave('NOBS/2020-05-25.txt', clear=0)
analysisandsave('NOBS/2020-05-26.txt', clear=0)
analysisandsave('NOBS/2020-05-27.txt', clear=0)
analysisandsave('NOBS/2020-05-28.txt', clear=0)
analysisandsave('NOBS/2020-05-29.txt', clear=0)
analysisandsave('NOBS/2020-05-30.txt', clear=0)
analysisandsave('NOBS/2020-05-31.txt', clear=0)
```

```
analysisandsavefiles()
```

```
#Python Script to resample and save/filter the external
→ files from BSRN

import pandas as pd
import numpy as np
import csv
import datetime
import matplotlib.pyplot as plt

def plot_external(filename, location):
    #data = pd.read_csv(filename, sep='\t')
    data = pd.read_csv(filename, sep=',')

    #getting day of year for every point
    #data = data.rename(columns={'Date/Time': 'Timestamp'})
    data.Timestamp = pd.DatetimeIndex(data.Timestamp)
    #data.time=pd.DatetimeIndex(data.time)
    data['Timestamp'].astype('Datetime64')
    doy = data['Timestamp'].dt.dayofyear
    doy = doy.values.tolist()
    day = np.array(doy)

    #getting timestamp as integer value
    data = data.set_index('Timestamp')
    data.index = data.index.hour + data.index.minute / 60
    time_utc = pd.Index.to_numpy(data.index)
```



```

# adding the different measurements into arrays
i_gh = np.array(data.iloc[:,2])
i_direct = np.array(data.iloc[:, 6])
i_diffuse = np.array(data.iloc[:, 10])

#making lists and appending the values for the same day
→ of years
i_gh_plot = []
i_direct_plot = []
i_diffuse_plot = []
time_utc_plot = []
startday = day[0]

for i in range(len(day)):
    if day[i] == startday:
        i_gh_plot.append(i_gh[i])
        i_diffuse_plot.append(i_diffuse[i])
        i_direct_plot.append(i_direct[i])
        time_utc_plot.append(time_utc[i])
    else:
        savefilename = str(location) + '_' + str(startday)
        → + '.png'
        plt.figure()
        plt.plot(time_utc_plot, i_gh_plot, label='Global
        → Horizontal', color='red')
        plt.plot(time_utc_plot, i_direct_plot,
        → label='Direct Normal', color='blue')
        plt.plot(time_utc_plot, i_diffuse_plot,
        → label='Diffuse Horizontal', color='green')
        plt.legend()
        plt.title(startday)
        plt.savefig(location + '/' + savefilename)
        #wiping data for new data
        i_gh_plot.clear()
        i_diffuse_plot.clear()
        i_direct_plot.clear()
        time_utc_plot.clear()
        startday = startday + 1

```

```

def resample(filename):
    #reading given filename using pandas
    #using function to easily call for another file to
    → import
    data = pd.read_csv(filename, sep='\t')

    #making the different header columns and adds the given
    → values to it, i_dn = direct normal, i_gn = global
    → normal, i_gh = global horizontal
    #all of the above uses the average value
    #timezone= ['']
    data = data.rename(columns={'Date/Time': 'Timestamp'})
    data.Timestamp = pd.DatetimeIndex(data.Timestamp)
    data['Timestamp'].astype('Datetime64')
    #data.index = data.index.tz_localize('UTC')
    #data.index = data.index.tz_convert(timezone)

    data = data.set_index('Timestamp')
    data = data.resample('5Min').mean()
    data.to_csv(filename+'_resampled.csv')

    data = data.reset_index()
    #formatted_time =
    → data['Timestamp'].dt.strftime("%H:%M:%S")
    doy = data['Timestamp'].dt.dayofyear
    doy = doy.values.tolist()
    day=np.array(doy)

    data = data.set_index('Timestamp')
    data.index = data.index.hour + data.index.minute / 60
    time_utc = pd.Index.to_numpy(data.index)

    data2=pd.read_csv(filename+'_resampled.csv', sep=',')
    # adding the different measurements into arrays
    i_gh = np.array(data2['SWD [W/m**2]'])
    i_direct = np.array(data2['DIR [W/m**2]'])

```

```

i_diffuse = np.array(data2['DIF [W/m**2]'])

return time_utc, day, i_gh, i_direct, i_diffuse

def solarangle_monthly(latitude_deg, longitude_deg, timezone,
→ time_utc, doy):
    length = len(time_utc)
    altitude_deg = np.zeros(length)
    zenith_deg = np.zeros(length)
    latitude_rad = np.deg2rad(latitude_deg)
    LSTM = np.zeros(length)
    time_local = np.zeros(length)
    TC = np.zeros(length)
    LST = np.zeros(length)
    HRA = np.zeros(length)
    HRA_rad = np.zeros(length)
    altitude_rad = np.zeros(length)
    B = np.zeros(length)
    EoT = np.zeros(length)
    dec_angle = np.zeros(length)
    dec_angle_rad = np.zeros(length)

    for i in range(length):
        # local standard time meridian
        delta_utc = timezone
        LSTM[i] = 15 * delta_utc
        time_local[i] = time_utc[i] + delta_utc

        # Equation of Time
        # EOT - np.sin uses radians as argument
        B[i] = (360 / 365) * (doy[i] - 81)
        EoT[i] = 9.87 * np.sin(np.deg2rad(2 * B[i])) - 7.53 *
→ np.cos(np.deg2rad(B[i])) - 1.5 *
→ np.sin(np.deg2rad(B[i]))

        # time correction

```

```

TC[i] = 4 * (longitude_deg - LSTM[i]) + EoT[i]

# Local Solar Time
LST[i] = time_local[i] + TC[i] / 60

# Hour Angle
HRA[i] = 15 * (LST[i] - 12)

HRA_rad[i] = np.deg2rad(HRA[i])

dec_angle[i] = 23.45 * np.sin(np.deg2rad(B[i]))
dec_angle_rad[i] = np.deg2rad(dec_angle[i])

altitude_rad[i] = np.arcsin(
    np.cos(latitude_rad) * np.cos(dec_angle_rad[i]) *
    ↪ np.cos(HRA_rad[i]) + np.sin(latitude_rad) *
    ↪ np.sin(
        dec_angle_rad[i]))
altitude_deg[i] = np.rad2deg(altitude_rad[i])
zenith_deg[i] = 90 - altitude_deg[i]

return altitude_deg

def exrad_monthly(time_utc, doy, i_gh, i_dn):
    length = len(time_utc)
    b = np.zeros(len(doy))
    I_ex = np.zeros(length)
    R = np.zeros(length)
    # extraterrestrial radiation constant
    I_exsc = 1367 # W/m^2
    for i in range(len(doy)):
        b[i] = 2 * np.pi * (doy[i] / 365)
        R[i] = 1.00011 + 0.34221 * np.cos(b[i]) + 0.00128 *
            ↪ np.sin(b[i]) + 0.000719 * np.cos(2 * b[i]) +
            ↪ 0.000077 * np.sin(2 * b[i])
        I_ex[i] = I_exsc * R[i]

```

```

# reading given
# calculating the reduction in radiation from
→ extraterrestrial to global

I_reduction_gh = 1-(i_gh / I_ex)
I_reduction_dn = 1-(i_dn / I_ex)

return I_reduction_gh,I_reduction_dn,I_ex

def clearnessdiffusecalc(igh, i_diffuse, iex):
    length=len(igh)
    kt_list=[]
    kd_list=[]
    # Filling Clearness Index
    for i in range(length):
        try:
            kt_list.append(igh[i] / iex[i])
        except ZeroDivisionError:
            kt_list.append(0)
            continue

    # Filling Diffuse Index
    for i in range(length):
        try:
            kd_list.append(i_diffuse[i] / igh[i])
        except ZeroDivisionError:
            kd_list.append(0)
            continue

    return kd_list,kt_list

#running resampling and filters out low global radiation and
→ too high diffuse radiation
def runandfilter(filename, location):
    if location=='Lindenberg':
        Timezone= 2

```

```

        Longitude= 14.122000
        Latitude= 52.210000
    if location=='Cener':
        Timezone = 2
        Longitude = -1.601000
        Latitude = 42.816000
    if location=='Toravere':
        Timezone = 3
        Longitude = 26.462000
        Latitude = 58.254000
    if location=='Izana':
        Timezone = 2
        Longitude = -16.499260
        Latitude = 28.309350

    [time_utc, day, i_gh, i_direct, i_diffuse] =
    → resample(filename)
    alt = solarangle_monthly(latitude_deg=Latitude ,
    → longitude_deg=Longitude , timezone=Timezone,
    → time_utc=time_utc, doy=day)
    [i_reduction_gh, i_reduction_dn, iex] =
    → exrad_monthly(time_utc, day, i_gh, i_direct)

    for i in reversed(range(len(time_utc))):
        if i_gh[i] < 5:
            i_gh = np.delete(i_gh,i)
            time_utc =np.delete(time_utc,i)
            i_direct=np.delete(i_direct,i)
            i_diffuse=np.delete(i_diffuse,i)
            day=np.delete(day,i)
            i_reduction_gh=np.delete(i_reduction_gh,i)
            i_reduction_dn = np.delete(i_reduction_dn, i)
            iex=np.delete(iex,i)
            alt=np.delete(alt,i)
    for i in reversed(range(len(time_utc))):
        if alt[i] < 5:
            i_gh = np.delete(i_gh, i)
            time_utc = np.delete(time_utc, i)

```

```

        i_direct = np.delete(i_direct, i)
        i_diffuse = np.delete(i_diffuse, i)
        day = np.delete(day, i)
        i_reduction_gh = np.delete(i_reduction_gh, i)
        i_reduction_dn = np.delete(i_reduction_dn, i)
        iex = np.delete(iex, i)
        alt = np.delete(alt, i)

    for i in reversed(range(len(time_utc))):
        if (i_diffuse[i]/iex[i]) > 0.8:
            i_gh = np.delete(i_gh,i)
            time_utc =np.delete(time_utc,i)
            i_direct=np.delete(i_direct,i)
            i_diffuse=np.delete(i_diffuse,i)
            day=np.delete(day,i)
            i_reduction_gh=np.delete(i_reduction_gh,i)
            i_reduction_dn = np.delete(i_reduction_dn,i)
            iex=np.delete(iex,i)
            alt=np.delete(alt,i)

    return time_utc, day, i_gh, i_direct, i_diffuse, iex,
        ↪ i_reduction_gh, i_reduction_dn, alt

def anglefinder_external(altitude_deg, i_gh, i_dn,
    ↪ I_reduction_gh, I_reduction_dn, doy):
    loc= np.zeros(len(altitude_deg))
    alt_list=[]
    altloc_list=[]
    red_gh_list=[]
    red_dn_list=[]
    doy_list=[]
    i_gh_list=[]
    i_dn_list=[]

    for k in range(1,10):
        for i in range(len(altitude_deg)):

```

```

        if altitude_deg[i] < (5*k + 0.25) and
        ↪ altitude_deg[i] > ((5*k)-0.25):
            loc[i] = 1
            alt_list.append(altitude_deg[i])
            altloc_list.append(i)
            red_gh_list.append(I_reduction_gh[i])
            red_dn_list.append(I_reduction_dn[i])
            doy_list.append(doy[i])
            i_gh_list.append(i_gh[i])
            i_dn_list.append(i_dn[i])
    return alt_list, i_gh_list, i_dn_list, red_gh_list,
    ↪ red_dn_list, doy_list

def savevalues(filename2, time_utc, day, i_gh, i_diffuse,
    ↪ i_direct, iex, alt, i_reduction_gh, i_reduction_dn):
    with open(filename2 + '_resampled_filtered.csv', 'a',
        ↪ newline='') as csvfile:
        fields = ['time', 'doy', 'igh', 'idni', 'idhi',
            ↪ 'ired_gh', 'ired_dn', 'iex', 'alt']
        writer = csv.DictWriter(csvfile,
            ↪ fieldnames=fields)
        writer.writeheader()
        for i in range(len(time_utc)):
            writer.writerow({'time': time_utc[i], 'doy':
                ↪ day[i], 'igh': i_gh[i], 'idni':
                ↪ i_direct[i], 'idhi': i_diffuse[i],
                ↪ 'ired_gh': i_reduction_gh[i], 'ired_dn':
                ↪ i_reduction_dn[i], 'iex': iex[i],
                ↪ 'alt':alt[i]})

def savevalues_clearday(filename2, altitude, i_gh, i_dn,
    ↪ i_reduction_gh, i_reduction_dn, firstfile=0):
    with open (filename2 + '_cleardays.csv', 'a', newline='')
        ↪ as csvfile:
        fields = ['altitude', 'i_gh', 'i_dn', 'reduction_gh',
            ↪ 'reduction_dn']
        writer = csv.DictWriter(csvfile, fieldnames=fields)

```



```

    if firstfile==1:
        writer.writeheader()
    for i in range(len(altitude)):
        writer.writerow({'altitude': altitude[i], 'i_gh':
            ↪ i_gh[i], 'i_dn': i_dn[i], 'reduction_gh':
            ↪ i_reduction_gh[i], 'reduction_dn':
            ↪ i_reduction_dn[i]})

def resamplesaveall():
    [time_utc, day, i_gh, i_direct, i_diffuse, iex,
    ↪ i_reduction_gh, i_reduction_dn, alt] =
    ↪ runandfilter('Lindenberg_radiation_mai.tab',
    ↪ location='Lindenberg')
    savevalues('Lindenberg_mai', time_utc, day, i_gh,
    ↪ i_diffuse, i_direct, iex, alt, i_reduction_gh,
    ↪ i_reduction_dn)

    [time_utc, day, i_gh, i_direct, i_diffuse, iex,
    ↪ i_reduction_gh, i_reduction_dn, alt] =
    ↪ runandfilter('cener_radiation_mai.tab',
    ↪ location='Cener')
    savevalues('Cener_mai', time_utc, day, i_gh, i_diffuse,
    ↪ i_direct, iex, alt, i_reduction_gh, i_reduction_dn)

    [time_utc, day, i_gh, i_direct, i_diffuse, iex,
    ↪ i_reduction_gh, i_reduction_dn, alt] =
    ↪ runandfilter('Izana_radiation_mai.tab',
    ↪ location='Izana')
    savevalues('Izana_mai', time_utc, day, i_gh, i_diffuse,
    ↪ i_direct, iex, alt, i_reduction_gh, i_reduction_dn)

    [time_utc, day, i_gh, i_direct, i_diffuse, iex,
    ↪ i_reduction_gh, i_reduction_dn, alt] =
    ↪ runandfilter('Toravere_radiation_mai.tab',
    ↪ location='Toravere')

```

```

savevalues('Toravere_mai', time_utc, day, i_gh, i_diffuse,
→ i_direct, iex, alt, i_reduction_gh, i_reduction_dn)

def savecleardays():
    cleardays_lind = np.array([139,147])
    cleardays_cener = [125, 133, 142, 150]
    cleardays_izana =[121, 122, 123, 124, 125, 126, 127, 129,
→ 130, 133, 134, 135, 137, 139, 140, 141, 142, 143, 144,
→ 145, 146, 147, 148, 149, 150]
    cleardays_toravere = [129, 136, 138, 150]

    data =
→ pd.read_csv('Lindenberg_mai_resampled_filtered.csv',
→ sep=',')
    # adding the different measurements into arrays
    doy = data['doy']
    doy = doy.values.tolist()
    i_red_gh=data['ired_gh']
    i_red_gh = i_red_gh.values.tolist()
    i_red_dn = data['ired_dn']
    i_red_dn = i_red_dn.values.tolist()
    altitude=data['alt']
    altitude = altitude.values.tolist()
    i_gh = np.array(data['igh'])
    i_dn = np.array(data['idni'])

    [alt,gh,dn,red_gh,red_dn,doy_list] =
→ anglefinder_external(altitude, i_gh, i_dn, i_red_gh,
→ i_red_dn, doy)

    with open('Lindenberg_cleardays.csv', 'a', newline='') as
→ csvfile:
        fields = ['altitude', 'i_gh', 'i_dn', 'reduction_gh',
→ 'reduction_dn']
        writer = csv.DictWriter(csvfile, fieldnames=fields)
        writer.writeheader()
        for k in range(len(cleardays_lind)):

```

```

        for i in range(len(doy_list)):
            if doy_list[i] == cleardays_lind[k]:
                writer.writerow({'altitude': alt[i],
                                ↪ 'i_gh': gh[i], 'i_dn': dn[i],
                                ↪ 'reduction_gh': red_gh[i],
                                ↪ 'reduction_dn': red_dn[i]})

data = pd.read_csv('Cener_mai_resampled_filtered.csv',
    ↪ sep=',')
# adding the different measurements into arrays
doy = data['doy']
doy = doy.values.tolist()
i_red_gh=data['ired_gh']
i_red_gh = i_red_gh.values.tolist()
i_red_dn = data['ired_dn']
i_red_dn = i_red_dn.values.tolist()
altitude=data['alt']
altitude = altitude.values.tolist()
i_gh = np.array(data['igh'])
i_dn = np.array(data['idni'])

[alt, gh, dn, red_gh, red_dn, doy_list] =
    ↪ anglefinder_external(altitude, i_gh, i_dn, i_red_gh,
    ↪ i_red_dn, doy)
with open('Cener_cleardays.csv', 'a', newline='') as
    ↪ csvfile:
    fields = ['altitude', 'i_gh', 'i_dn', 'reduction_gh',
    ↪ 'reduction_dn']
    writer = csv.DictWriter(csvfile, fieldnames=fields)
    writer.writeheader()
    for k in range(len(cleardays_cener)):
        for i in range(len(doy_list)):
            if doy_list[i]==cleardays_cener[k]:
                writer.writerow({'altitude': alt[i],
                                ↪ 'i_gh': gh[i], 'i_dn': dn[i],
                                ↪ 'reduction_gh': red_gh[i],
                                ↪ 'reduction_dn': red_dn[i]})

```

```

data = pd.read_csv('Izana_mai_resampled_filtered.csv',
    ↪ sep=',')
# adding the different measurements into arrays
doy = data['doy']
doy = doy.values.tolist()
i_red_gh=data['ired_gh']
i_red_gh = i_red_gh.values.tolist()
i_red_dn = data['ired_dn']
i_red_dn = i_red_dn.values.tolist()
altitude = data['alt']
altitude = altitude.values.tolist()
i_gh = np.array(data['igh'])
i_dn = np.array(data['idni'])

[alt, gh, dn, red_gh, red_dn, doy_list] =
    ↪ anglefinder_external(altitude, i_gh, i_dn, i_red_gh,
    ↪ i_red_dn, doy)

with open('Izana_clear_days.csv', 'a', newline='') as
    ↪ csvfile:
    fields = ['altitude', 'i_gh', 'i_dn', 'reduction_gh',
    ↪ 'reduction_dn']
    writer = csv.DictWriter(csvfile, fieldnames=fields)
    writer.writeheader()
    for k in range(len(clear_days_izana)):
        for i in range(len(doy_list)):
            if doy_list[i] == clear_days_izana[k]:
                writer.writerow({'altitude': alt[i],
                    ↪ 'i_gh': gh[i], 'i_dn': dn[i],
                    ↪ 'reduction_gh': red_gh[i],
                    ↪ 'reduction_dn': red_dn[i]})

data = pd.read_csv('Toravere_mai_resampled_filtered.csv',
    ↪ sep=',')
# adding the different measurements into arrays
doy = data['doy']
doy = doy.values.tolist()
i_red_gh=data['ired_gh']

```

```

i_red_gh = i_red_gh.values.tolist()
i_red_dn = data['ired_dn']
i_red_dn = i_red_dn.values.tolist()
altitude = data['alt']
altitude = altitude.values.tolist()
i_gh = np.array(data['igh'])
i_dn = np.array(data['idni'])

[alt, gh, dn, red_gh, red_dn, doy_list] =
↳ anglefinder_external(altitude, i_gh, i_dn, i_red_gh,
↳ i_red_dn, doy)

with open('Toravere_cleardays.csv', 'a', newline='') as
↳ csvfile:
    fields = ['altitude', 'i_gh', 'i_dn', 'reduction_gh',
↳ 'reduction_dn']
    writer = csv.DictWriter(csvfile, fieldnames=fields)
    writer.writeheader()
    for k in range(len(cleardays_toravere)):
        for i in range(len(doy_list)):
            if doy_list[i] == cleardays_toravere[k]:
                writer.writerow({'altitude': alt[i],
↳ 'i_gh': gh[i], 'i_dn': dn[i],
↳ 'reduction_gh': red_gh[i],
↳ 'reduction_dn': red_dn[i]})

def indexcalc(filename, savename):
    data = pd.read_csv(filename, sep=',')
    i_gh = np.array(data['igh'])
    i_dh = np.array(data['idhi'])
    i_ex = np.array(data['iex'])

    [kd_list, kt_list] = clearnessdiffusecalc(i_gh, i_dh,
↳ i_ex)
    #checking for faults
    #making nan values 0.0
    kd_list = np.nan_to_num(kd_list)
    kt_list = np.nan_to_num(kt_list)

```

```

#taking away these NaN values(which now is zero)
for i in reversed(range(len(kd_list))):
    if kd_list[i] == 0:
        kd_list = np.delete(kd_list, i)
        kt_list = np.delete(kt_list,i)
for i in reversed(range(len(kd_list))):
    if kt_list[i] == 0:
        kd_list = np.delete(kd_list, i)
        kt_list = np.delete(kt_list,i)

#filtering for impossibly high or low kt/kd values
for i in reversed(range(len(kd_list))):
    if kd_list[i] > 1:
        kd_list = np.delete(kd_list, i)
        kt_list = np.delete(kt_list,i)
for i in reversed(range(len(kd_list))):
    if kd_list[i] < 0:
        kd_list = np.delete(kd_list, i)
        kt_list = np.delete(kt_list, i)
for i in reversed(range(len(kd_list))):
    if kt_list[i] > 1:
        kt_list=np.delete(kt_list,i)
        kd_list = np.delete(kd_list, i)
for i in reversed(range(len(kd_list))):
    if kt_list[i] < 0:
        kt_list=np.delete(kt_list,i)
        kd_list = np.delete(kd_list, i)

with open(savename, 'a', newline='') as csvfile:
    fields = ['kd_list', 'kt_list']
    writer = csv.DictWriter(csvfile, fieldnames=fields)
    writer.writeheader()
    for i in range(len(kd_list)):
        writer.writerow({'kd_list': kd_list[i], 'kt_list':
            ↪ kt_list[i]})

#resamplesaveall()
#savecleardays()

```

```

#indexcalc('Izana_mai_resampled_filtered.csv',
→ savename='Izana_indexes.csv')
#indexcalc('Toravere_mai_resampled_filtered.csv',
→ savename='Toravere_indexes.csv')
#indexcalc('Cener_mai_resampled_filtered.csv',
→ savename='Cener_indexes.csv')
#indexcalc('Lindenberg_mai_resampled_filtered.csv',
→ savename='Lindenberg_indexes.csv')

plot_external('Lindenberg_radiation_mai.tab_resampled.csv',
→ location='Lindenberg')

```

```

#Altered Script with the Skartveit Albedo Corrected Model
→ from Lukas Böhme

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import datetime as dt

def model_Skartveit(time_utc,
i_gh,
i_dhi,
iex,
altitude_deg):

    # time
    time_list = []

    # solar altitude
    sh_list = []

    # azimuth angle
    az_list = []

```

```

# global radiation
ig_list = []

# i_diffuse horizontal
id_list = []

# i_extraterrestrial
iex_list = []

ib_list = []
ib2_list = []

# Clearness index
kt_list = []

# Diffuse index
kd_list = []

o3_list = []
rho_list = []
k1_list = []

# length of arrays
length = len(time_utc)

for i in range(length):
    ig_list.append(i_gh[i])
    time_list.append(time_utc[i])
    iex_list.append(iex[i])

for i in range(length):
    sh_list.append(altitude_deg[i])
    id_list.append(i_dhi[i])

# Calculate k_t and k_d and fill in their list
"""k_t"""
for line in range(len(ig_list)):

```



```

    try:
        kt_list.append(ig_list[line] / iex_list[line])
    except ZeroDivisionError:
        kt_list.append(0)
        continue

"""k_d"""
for line in range(len(time_list)):
    try:
        kd_list.append(id_list[line] / ig_list[line])
    except ZeroDivisionError:
        kd_list.append(0)
        continue

# The albedos are rough estimates dependent on the
↪ duration of a snow cover at the station.
albedo = 0.2

r = albedo
r_star = 0.15

if (np.abs(r - r_star) > 0.1):
    A = 0.2
    h = np.array(sh_list)
    h_prime = np.copy(h)
    h_prime[:] = 37
    k_r = np.array(kt_list)

    def k_1(x):
        k_1 = 0.83 - 0.56 * np.exp(-0.006 * x)

        return k_1

    k_r_prime = k_r * k_1(h_prime) / k_1(h)

    R = (1 - A - k_r_prime) / (1 - k_r_prime * r)
    R[R < 0.08] = 0.08

```

```

k_r_star = k_r * ((1 - r * R) / (1 - r_star * R))

kt_list = k_r_star

# The standard SED model

"""k1"""
for line in range(len(time_list)):
    try:
        k1_list.append(0.83 - 0.56 * np.exp(-0.06 *
            ↪ sh_list[line]))
    except ZeroDivisionError:
        k1_list.append(0)
        continue

"""rho"""
for line in range(len(time_list)):
    try:
        rho_list.append(kt_list[line] / k1_list[line])
    except ZeroDivisionError:
        rho_list.append(0)
        continue

"""o3"""
o3_list = [None] * len(time_list)
for line in range(len(time_list)):
    if line == 0:
        o3_list[line] = np.abs(rho_list[line] -
            ↪ rho_list[line + 1])
        continue

    if line == len(time_list) - 1:
        o3_list[line] = np.abs(rho_list[line] -
            ↪ rho_list[line - 1])
        continue

```

```

o3_list[line] = (((rho_list[line] - rho_list[line -
↪ 1]) ** 2 + (
                    rho_list[line] - rho_list[line + 1]) ** 2)
↪ / 2) ** 0.5

kd_model = [None] * len(kd_list)

k2_list = []
for line in range(len(k1_list)):
    k2_list.append(k1_list[line] * 0.95)

alpha = []
for line in range(len(k1_list)):
    if sh_list[line] == 0:
        alpha.append(0)
    else:
        try:
            alpha.append((1 / (np.sin(sh_list[line] *
↪ np.pi / 180)) ** 0.6))
        except:
            alpha.append(0)
            continue

kbmax = []
for line in range(len(k1_list)):
    kbmax.append(0.81 ** alpha[line])

d1_list = []
for line in range(len(k1_list)):
    if sh_list[line] <= 1.4:
        d1_list.append(1)
    else:
        d1_list.append(0.07 + 0.046 * ((90 -
↪ sh_list[line]) / (sh_list[line] + 3)))

d2_list = []
for line in range(len(k1_list)):

```

```

K2 = 0.5 * (1 + np.sin(np.pi * ((k2_list[line] - 0.22)
↪ / (k1_list[line] - 0.22)) - np.pi / 2))

d2_list.append(1 - (1 - d1_list[line]) * (0.11 *
↪ np.sqrt(K2) + 0.15 * K2 + 0.74 * K2 ** 2))

K2 = None

ktmax = []
for line in range(len(k1_list)):
    ktmax.append((kbmax[line] + (d2_list[line] *
↪ k2_list[line]) / (1 - k2_list[line])) / (
        1 + (d2_list[line] * k2_list[line]) / (1 -
↪ k2_list[line])))

kdmax = []
for line in range(len(k1_list)):
    kdmax.append((d2_list[line] * k2_list[line] * (1 -
↪ ktmax[line])) / (ktmax[line] * (1 -
↪ k2_list[line])))

kx_list = []
for line in range(len(k1_list)):
    kx_list.append(0.56 - 0.32 * np.exp(-0.06 *
↪ sh_list[line]))

kl_list = []
for line in range(len(k1_list)):
    kl_list.append((kt_list[line] - 0.14) / (kx_list[line]
↪ - 0.14))

kr_list = []
for line in range(len(k1_list)):
    kr_list.append((kt_list[line] - kx_list[line]) / 0.71)

"""Invariable hours"""

for line in range(len(ig_list)):

```

```

if o3_list[line] <= 0.0000001:
    if kt_list[line] <= 0.22:
        kd_model[line] = 1

    elif 0.22 < kt_list[line] <= k2_list[line]:
        K = 0.5 * (1 + np.sin(np.pi * ((kt_list[line]
        ↪ - 0.22) / (k1_list[line] - 0.22)) - np.pi
        ↪ / 2))

        kd_model[line] = 1 - (1 - d1_list[line]) *
        ↪ (0.11 * np.sqrt(K) + 0.15 * K + 0.74 * K
        ↪ ** 2)

    elif k2_list[line] < kt_list[line] <=
    ↪ ktmax[line]:
        kd_model[line] = (d2_list[line] *
        ↪ k2_list[line] * (1 - kt_list[line])) / (
            kt_list[line] * (1 -
            ↪ k2_list[line]))

    elif kt_list[line] > ktmax[line]:
        kd_model[line] = 1 - ((ktmax[line] * (1 -
        ↪ kdmax[line])) / (kt_list[line]))

"""Variable hours"""

if o3_list[line] > 0.0000001:
    """Define delta"""
    if 0.14 <= kt_list[line] <= kx_list[line]:
        delta = -3 * kl_list[line] ** 2 * (1 -
        ↪ kl_list[line]) * o3_list[line] ** 1.3

    elif kx_list[line] < kt_list[line] <=
    ↪ (kx_list[line] + 0.71):
        delta = 3 * kr_list[line] * (1 -
        ↪ kr_list[line]) ** 2 * o3_list[line] ** 0.6

    elif kt_list[line] < 0.14:

```

```

        delta = 0

elif kt_list[line] > (kx_list[line] + 0.71):
    delta = 0

else:
    print('Line: ', line, ' k_t is: ',
          ↪ kt_list[line])

"""Now the old code with delta"""

if kt_list[line] <= 0.22:
    kd_model[line] = 1 + delta

elif 0.22 < kt_list[line] <= k2_list[line]:
    K = 0.5 * (1 + np.sin(np.pi * ((kt_list[line]
    ↪ - 0.22) / (k1_list[line] - 0.22)) - np.pi
    ↪ / 2))

    kd_model[line] = 1 - (1 - d1_list[line]) *
    ↪ (0.11 * np.sqrt(K) + 0.15 * K + 0.74 * K
    ↪ ** 2) + delta

elif k2_list[line] < kt_list[line] <=
    ↪ ktmax[line]:
    kd_model[line] = (d2_list[line] *
    ↪ k2_list[line] * (1 - kt_list[line])) / (
    ↪ kt_list[line] * (1 -
    ↪ k2_list[line])) + delta

elif kt_list[line] > ktmax[line]:
    kd_model[line] = 1 - ((ktmax[line] * (1 -
    ↪ kdmax[line])) / (kt_list[line])) + delta

# Now again the correction for albedo
if (np.abs(r - r_star) > 0.1):
    d_r_star = np.array(kd_model)

```

```

        d_r = 1 - (k_r_star * (1 - d_r_star) / k_r)

        kd_model = d_r

    return kd_model,
           kd_list,
           kt_list

```

```

#making a file to analyze the finished files from the
→ resample_external.py and ExRad.py

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from SKARTVEIT import model_Skartveit
#for clear days, the reduction / angle

def skartveit_do(filename,location):
    #idn = direct normal, idh = diffuse horizontal
    data = pd.read_csv(filename, sep=",")
    time = np.array(data['time'])
    doy = np.array(data['doy'])
    igh = np.array(data['igh'])
    ired = np.array(data['i_gh_reduction'])
    iex = np.array(data['iex'])
    alt = np.array(data['alt'])
    idhi = np.array(data['idhi'])
    [kd_model, kd_list, kt_list] = model_Skartveit(time_utc =
    → time, i_gh = igh, i_dhi = idhi, iex = iex,
    → altitude_deg=alt, location=location)

    # filtering for impossibly high or low kt/kd values
    for i in reversed(range(len(kd_list))):
        if kd_list[i] > 1:
            kd_list = np.delete(kd_list, i)
            kt_list = np.delete(kt_list, i)
            kd_model = np.delete(kd_model,i)

```

```

for i in reversed(range(len(kd_list))):
    if kd_list[i] < 0:
        kd_list = np.delete(kd_list, i)
        kt_list = np.delete(kt_list, i)
        kd_model = np.delete(kd_model, i)
for i in reversed(range(len(kd_list))):
    if kt_list[i] > 1:
        kt_list = np.delete(kt_list, i)
        kd_list = np.delete(kd_list, i)
        kd_model = np.delete(kd_model, i)
for i in reversed(range(len(kd_list))):
    if kt_list[i] < 0:
        kt_list = np.delete(kt_list, i)
        kd_list = np.delete(kd_list, i)
        kd_model = np.delete(kd_model, i)
return kd_model, kd_list, kt_list

def clearday_analysis_tromso(filename, location):
    # idn = direct normal, idh = diffuse horizontal

    data = pd.read_csv(filename, sep=",")
    alt_list = np.array(data['altitude'])
    reduction_gh = np.array(data['gh_reduction'])
    reduction_gn = np.array(data['gn_reduction'])
    reduction_dn = np.array(data['dn_reduction'])

    return alt_list, reduction_gh, reduction_gn, reduction_dn

def clearday_analysis_ext(filename):

    data = pd.read_csv(filename, sep=",")
    altitude = np.array(data['altitude'])
    reduction_gh = np.array(data['reduction_gh'])
    reduction_dn = np.array(data['reduction_dn'])
    return altitude, reduction_gh, reduction_dn

def airmass_dn_gh_calc(filename):
    def AM(x):

```



```

    am = 1 / (np.cos(np.deg2rad(x)) + 0.50572 * (96.07995
    ↪ - x) ** (-1.6364))
    return am

data = pd.read_csv(filename, sep=",")
altitude = np.array(data['altitude'])
i_dn = np.array(data['i_dn'])
i_gh = np.array(data['i_gh'])
zenith = np.array(90 - altitude)
AM = np.array(AM(zenith))
Direct_AM = 1353 * 0.7 ** (AM*0.678)
Global_AM = 1.1 * Direct_AM
print(Direct_AM)
print(Global_AM)
plt.plot()
plt.scatter(altitude, Global_AM, label='Global Horizontal
↪ Radiation Calculated')
plt.scatter(altitude, Direct_AM, label='Direct Normal
↪ Radiation Calculated')
plt.scatter(altitude, i_gh, label='Global Horizontal
↪ Radiation Measured')
plt.scatter(altitude, i_dn, label='Direct Normal Radiation
↪ Measured')
plt.title('GHI and DNI Measured vs Calculated')
plt.legend()
plt.show()
return Direct_AM,Global_AM

def altitudeplots():
    [alt_tromso, red_gh_tromso, red_gn_tromso, red_dn_tromso]
    ↪ =
    ↪ clearday_analysis_tromso('NOBS_altitudereduction.csv',
    ↪ location='Tromso')

#figure of GHI reduction @ Tromsø

```

```

plt.figure()
plt.scatter(alt_tromso, red_gh_tromso, label='Global
→ Horizontal Reduction')
plt.title('Global Horizontal Reduction at Tromsø,Norway')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#figure of GNI reduction @ Tromsø
plt.figure()
plt.scatter(alt_tromso, red_gn_tromso, label='Global
→ Normal Reduction')
plt.title('Global Normal Reduction at Tromsø,Norway')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Normal Reduction')
plt.legend()
plt.grid()
plt.show()

#figure of DNI reduction @ Tromsø
plt.figure()
plt.scatter(alt_tromso, red_dn_tromso, label='Direct
→ Normal Reduction')
plt.title('Direct Normal Reduction at Tromsø,Norway')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

[alt_izana, red_gh_izana, red_dn_izana] =
→ clearday_analysis_ext('Izana_cleardays.csv')
[alt_tora, red_gh_tora, red_dn_tora] =
→ clearday_analysis_ext('Toravere_cleardays.csv')

```

```

[alt_cener, red_gh_cener, red_dn_cener] =
→ clearday_analysis_ext('Cener_cleardays.csv')
[alt_lind, red_gh_lind, red_dn_lind] =
→ clearday_analysis_ext('Lindenberg_cleardays.csv')

#Global Horizontal Reduction @ Lindenberg
plt.figure()
plt.scatter(alt_lind, red_gh_lind, label='Global
→ Horizontal Reduction')
plt.title('Global Horizontal Reduction at Lindenberg,
→ Germany')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#Global Horizontal Reduction @ Cener
plt.figure()
plt.scatter(alt_cener, red_gh_cener, label='Global
→ Horizontal Reduction')
plt.title('Global Horizontal Reduction at Cener,Spain')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#Global Horizontal Reduction @ Izana
plt.figure()
plt.scatter(alt_izana, red_gh_izana, label='Global
→ Horizontal Reduction')
plt.title('Global Horizontal Reduction at Izana, Spain')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()

```

```

plt.show()

#Global Horizontal Reduction @ Toravere
plt.figure()
plt.scatter(alt_tora, red_gh_tora, label='Direct Normal
→ Reduction')
plt.title('Global Horizontal Reduction at
→ Toravere,Estonia')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#Global Horizontal Reduction @Tromsø/Linden
plt.figure()
plt.scatter(alt_tromso, red_gh_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_lind, red_gh_lind, label='Lindenberg,
→ Germany')
plt.title('Global Horizontal Reduction Tromsø vs
→ Lindenberg')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

# Global Horizontal Reduction @Tromsø/Cener
plt.figure()
plt.scatter(alt_tromso, red_gh_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_cener, red_gh_cener, label='Cener, Spain')
plt.title('Global Horizontal Reduction Tromsø vs Cener')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()

```

```

plt.show()

# Global Horizontal Reduction @Tromsø/Izana
plt.figure()
plt.scatter(alt_tromso, red_gh_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_izana, red_gh_izana, label='Izana, Spain')
plt.title('Global Horizontal Reduction Tromsø vs Izana')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

# Global Horizontal Reduction @Tromsø/Toravere
plt.figure()
plt.scatter(alt_tromso, red_gh_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_tora, red_gh_tora, label='Toravere,
→ Estonia')
plt.title('Global Horizontal Reduction Tromsø vs
→ Toravere')
plt.xlabel('Elevation Angle')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#Direct Normal Reduction @Tromsø/Linden
plt.figure()
plt.scatter(alt_tromso, red_dn_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_lind, red_dn_lind, label='Lindenberg,
→ Germany')
plt.title('Direct Normal Reduction Tromsø vs Lindenberg')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()

```

```

plt.grid()
plt.show()

#Direct Normal Reduction @Tromsø/Cener
plt.figure()
plt.scatter(alt_tromso, red_dn_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_cener, red_dn_cener, label='Cener, Spain')
plt.title('Direct Normal Reduction Tromsø vs Cener')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

# Direct Normal Reduction @Tromsø/Izana
plt.figure()
plt.scatter(alt_tromso, red_dn_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_izana, red_dn_izana, label='Izana, Spain')
plt.title('Direct Normal Reduction Tromsø vs Izana')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

# Direct Normal Reduction @Tromsø/Toravere
plt.figure()
plt.scatter(alt_tromso, red_dn_tromso, label='Tromsø,
→ Norway')
plt.scatter(alt_tora, red_dn_tora, label='Toravere,
→ Estonia')
plt.title('Direct Normal Reduction Tromsø vs Toravere')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()

```

```

plt.show()

# Direct Normal Reduction @ Lindenberg
plt.figure()
plt.scatter(alt_lind, red_dn_lind, label='Direct Normal
→ Reduction')
plt.title('Direct Normal Reduction at Lindenberg,
→ Germany')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

# Direct Normal Reduction @ Cener
plt.figure()
plt.scatter(alt_cener, red_dn_cener, label='Direct Normal
→ Reduction')
plt.title('Direct Normal Reduction at Cener, Spain')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

# Direct Normal Reduction @ Izana
plt.figure()
plt.scatter(alt_izana, red_dn_izana, label='Direct Normal
→ Reduction')
plt.title('Direct Normal Reduction at Izana, Spain')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

# Direct Normal Reduction @ Toravere
plt.figure()

```

```

plt.scatter(alt_tora, red_dn_tora, label='Global
→ Horizontal Reduction')
plt.title('Direct Normal Reduction at Toravere,Estonia')
plt.xlabel('Elevation Angle')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

def plotall():
    # getting indexes + model index for tromsø
    [kd_mod_tromso, kd_tromso, kt_tromso] =
    → skartveit_do('NOBS_alldays_filtered.csv',
    → location='Tromso')
    data_lind = pd.read_csv('Lindenberg_indexes.csv', sep=",")
    data_tora = pd.read_csv('Toravere_indexes.csv', sep=",")
    data_izana = pd.read_csv('Izana_indexes.csv', sep=",")
    data_cener = pd.read_csv('Cener_indexes.csv', sep=",")

    #getting indexes from external location
    kd_lind = np.array(data_lind['kd_list'])
    kt_lind = np.array(data_lind['kt_list'])

    kd_tora = np.array(data_tora['kd_list'])
    kt_tora = np.array(data_tora['kt_list'])

    kd_cener = np.array(data_cener['kd_list'])
    kt_cener = np.array(data_cener['kt_list'])

    kd_izana = np.array(data_izana['kd_list'])
    kt_izana = np.array(data_izana['kt_list'])

    #SKARTVEIT
    #TROMSØ
    plt.figure()

```



```

plt.scatter(kd_mod_tromso, kt_tromso, label='Index Model',
    ↪ facecolors='none', edgecolors='r')
plt.scatter(kd_tromso, kt_tromso, label='Index
    ↪ Measurements', facecolors='none', edgecolors='b')
plt.title('Diffuse vs Clearness Index, Tromsø')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#LINDENBERG
plt.figure()
plt.scatter(kd_lind, kt_lind, label='Index Measurements',
    ↪ facecolors='none', edgecolors='r')
plt.title('Diffuse vs Clearness Index, Lindenberg')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#IZANA
plt.figure()
plt.scatter(kd_izana, kt_izana, label='Index
    ↪ Measurements', facecolors='none', edgecolors='r')
plt.title('Diffuse vs Clearness Index, Izana')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#CENER
plt.figure()
plt.scatter(kd_cener, kt_cener, label='Index
    ↪ Measurements', facecolors='none', edgecolors='r')
plt.title('Diffuse vs Clearness Index, Cener')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()

```

```

plt.show()

#TORAVERE
plt.figure()
plt.scatter(kd_tora, kt_tora, label='Index Measurements',
    → facecolors='none', edgecolors='r')
plt.title('Diffuse vs Clearness Index, Toravere')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#TROMSØ/LINDEN
plt.figure()
plt.scatter(kd_tromso, kt_tromso, label='Index
    → Measurements Tromsø', facecolors='none',
    → edgecolors='b')
plt.scatter(kd_lind, kt_lind, label='Index Measurements
    → Lindenberg', facecolors='none', edgecolors='g')
plt.title('Comparison Lindenberg vs Tromsø')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#TROMSØ/CENER
plt.figure()
plt.scatter(kd_tromso, kt_tromso, label='Index
    → Measurements Tromsø', facecolors='none',
    → edgecolors='b')
plt.scatter(kd_cener, kt_cener, label='Index Measurements
    → Cener', facecolors='none', edgecolors='g')
plt.title('Comparison Cener vs Tromsø')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

```

```

#TROMSØ/IZANA
plt.figure()
plt.scatter(kd_tromso, kt_tromso, label='Index
→ Measurements Tromsø', facecolors='none',
→ edgecolors='b')
plt.scatter(kd_izana, kt_izana, label= 'Index Measurements
→ Izana', facecolors='none', edgecolors='g')
plt.title('Comparison Izana vs Tromsø')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#TROMSØ/TORAVERE
plt.figure()
plt.scatter(kd_tromso, kt_tromso, label='Index
→ Measurements Tromsø', facecolors='none',
→ edgecolors='b')
plt.scatter(kd_tora, kt_tora, label='Index Measurements
→ Lindenberg', facecolors='none', edgecolors='g')
plt.title('Comparison Toravere vs Tromsø')
plt.xlabel('Diffuse Index')
plt.ylabel('Clearness Index')
plt.legend()
plt.show()

#function for plotting new izana values.
def izana_new_reduction():
    data = pd.read_csv('izana_cleardays.csv')
    alt = np.array(data['altitude'])
    i_gh = np.array(data['i_gh'])
    i_dn = np.array(data['i_dn'])
    reduction_gh = np.array(data['reduction_gh'])
    reduction_dn = np.array(data['reduction_dn'])

    i_dh = i_gh - i_dn * np.cos(np.deg2rad(90 - alt))

```

```

data3 = pd.read_csv('Cener_cleardays.csv')
alt_cener = np.array(data3['altitude'])
i_gh_cener = np.array(data3['i_gh'])
i_dn_cener = np.array(data3['i_dn'])
reduction_gh_cener = np.array(data3['reduction_gh'])
reduction_dn_cener = np.array(data3['reduction_dn'])

data2 = pd.read_csv('NOBS_altitudereduction.csv')
gh_red_tromso = np.array(data2['gh_reduction'])
alt_tromso = np.array(data2['altitude'])
dn_red_tromso = np.array(data2['dn_reduction'])

#finding original Iex value to not have to change the
→ whole program

I_ex = i_gh / (1-reduction_gh)
I_ex = np.nan_to_num(I_ex)

I_ex_cener = i_gh_cener/(1-reduction_gh_cener)
I_ex_cener = np.nan_to_num(I_ex_cener)

I_dn_red = i_dh / I_ex

plt.figure()
plt.scatter(alt, I_dn_red)
plt.show()

for i in reversed(range(len(I_ex))):
    if I_ex[i] == 0:
        i_gh = np.delete(i_gh, i)
        i_dn = np.delete(i_dn, i)
        I_ex = np.delete(I_ex, i)
        alt = np.delete(alt, i)
        reduction_dn = np.delete(reduction_dn, i)
        reduction_gh = np.delete(reduction_gh, i)

for i in reversed(range(len(I_ex_cener))):

```

```

    if I_ex_cener[i] == 0:
        i_gh_cener = np.delete(i_gh_cener, i)
        i_dn_cener = np.delete(i_dn_cener, i)
        I_ex_cener = np.delete(I_ex_cener, i)
        alt_cener = np.delete(alt_cener, i)
        reduction_dn_cener = np.delete(reduction_dn_cener,
                                        ↪ i)
        reduction_gh_cener = np.delete(reduction_gh_cener,
                                        ↪ i)

#calculating new i_gh for izana
i_gh_new = i_gh*0.769
i_dn_new = i_dn*0.769
I_reduction_gh_new = 1- (i_gh_new / I_ex)
I_reduction_dn_new = 1 - (i_dn_new/I_ex)

Increase = I_reduction_gh_new / reduction_gh
Increase_mean = np.mean(Increase)
Increase2 = I_reduction_dn_new / reduction_dn
Increase2_mean = np.mean(Increase2)
print(Increase)
print(Increase_mean)
print(Increase2)
print(Increase2_mean)

#calculating new i_gh for cener
i_gh_new_cener = i_gh_cener*0.953
i_dn_new_cener = i_dn_cener*0.953
I_reduction_gh_new_cener = 1- (i_gh_new_cener /
↪ I_ex_cener)
I_reduction_dn_new_cener = 1 - (i_dn_new_cener /
↪ I_ex_cener)

#standalone plot GHI Izana
plt.figure()

```

```

plt.scatter(alt, I_reduction_gh_new, label='Compensated
→ Global Horizontal Reduction')
plt.title('Global Horizontal Reduction at Izana, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

# standalone plot GHI Cener
plt.figure()
plt.scatter(alt_cener, I_reduction_gh_new_cener,
→ label='Compensated Global Horizontal Reduction')
plt.title('Global Horizontal Reduction at Cener, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#comparison plot GHI for izana
plt.figure()
plt.scatter(alt, I_reduction_gh_new, label='Compensated')
plt.scatter(alt, reduction_gh, label='Uncompensated')
plt.title('Global Horizontal Reduction at Izana, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#comparison plot GHI for cener
plt.figure()
plt.scatter(alt_cener, I_reduction_gh_new_cener,
→ label='Compensated')
plt.scatter(alt_cener, reduction_gh_cener,
→ label='Uncompensated')
plt.title('Global Horizontal Reduction at Cener, Spain')

```

```

plt.xlabel('Solar Altitude')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#standalone plot DNI Izana
plt.figure()
plt.scatter(alt, I_reduction_dn_new, label='Compensated
↳ Direct Normal Reduction')
plt.title('Direct Normal Reduction at Izana, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

#standalone plot DNI Cener
plt.figure()
plt.scatter(alt_cener, I_reduction_dn_new_cener,
↳ label='Compensated Direct Normal Reduction')
plt.title('Direct Normal Reduction at Cener, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

#comparison plot DNI for Izana
plt.figure()
plt.scatter(alt, I_reduction_dn_new, label='Compensated')
plt.scatter(alt, reduction_dn, label='Uncompensated')
plt.title('Direct Normal Reduction at Izana, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

```

```

#comparison plot DNI for Cener
plt.figure()
plt.scatter(alt_cener, I_reduction_dn_new_cener,
    → label='Compensated')
plt.scatter(alt_cener, reduction_dn_cener,
    → label='Uncompensated')
plt.title('Direct Normal Reduction at Cener, Spain')
plt.xlabel('Solar Altitude')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

#comparison between compensated values izana and tromsø
→ GHI
plt.figure()
plt.scatter(alt, I_reduction_gh_new, label='Izana')
plt.scatter(alt_tromso, gh_red_tromso, label='Tromsø')
plt.title('Comparison between Tromsø and compensated Izana
    → values')
plt.xlabel('Solar Altitude')
plt.ylabel('Global Horizontal Reduction')
plt.legend()
plt.grid()
plt.show()

#comparison between compensated values cener and tromsø
→ GHI
plt.figure()
plt.scatter(alt_cener, I_reduction_gh_new_cener,
    → label='Cener')
plt.scatter(alt_tromso, gh_red_tromso, label='Tromsø')
plt.title('Comparison between Tromsø and compensated Cener
    → values')
plt.xlabel('Solar Altitude')
plt.ylabel('Global Horizontal Reduction')
plt.legend()

```



```

plt.grid()
plt.show()

# comparison between new values izana and tromsø DNI
plt.figure()
plt.scatter(alt, I_reduction_dn_new, label='Izana')
plt.scatter(alt_tromso, dn_red_tromso, label='Tromsø')
plt.title('Comparison between Tromsø and compensated Izana
→ values')
plt.xlabel('Solar Altitude')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

# comparison between new values cener and tromsø DNI
plt.figure()
plt.scatter(alt_cener, I_reduction_dn_new_cener,
→ label='Cener')
plt.scatter(alt_tromso, dn_red_tromso, label='Tromsø')
plt.title('Comparison between Tromsø and compensated Cener
→ values')
plt.xlabel('Solar Altitude')
plt.ylabel('Direct Normal Reduction')
plt.legend()
plt.grid()
plt.show()

#altitudeplots()
izana_new_reduction()
#plotall()

```

