FACULTY OF SCIENCE

LUND UNIVERSITY

Effects of stratospheric wildfire smoke on ozone depleting substances and ozone levels in the northern midlatitudes

by

Christopher Grüner

Thesis submitted for the degree of Master of Science
Project duration: 4 months
Supervised by Johan Friberg and Carl Svenhag

Department of Physics
Division of Nuclear Physics
March 2023

Abstract

The goal of this project was to examine whether there are correlations between recent wildfires in the northern hemisphere and stratospheric ozone anomalies in the midlatitudes following the creation of ozone depleting substances. To achieve this goal, data from the Aura satellite's Microwave Limb Sounder instrument was used and processed with self-written Python scripts.

Firstly, in order to analyze the data appropriately and to set it into context, results of previously published papers regarding the 2019-20 Australian wildfire were verified with the data. As a result, perturbations in hydrogen chloride, chloromethane, and ozone following the fires were found. The underlying chemical mechanism can possibly be explained by the hydrogenation of smoke particles and subsequent reactions on their surfaces. As an example for the northern hemisphere, the 2017 wildfire in British Columbia was analyzed in accordance with these findings. The process resulted in no connections between the Canadian wildfire and ozone destruction produced by ozone depleting substances being found in the data.

Therefore, in conclusion, there might be a certain threshold in injected smoke particle mass into the stratosphere. The injection of the Canadian wildfire was then potentially not enough in order to invoke sufficient production of ozone depleting substances compared to the initial production of ozone by smoke particles to deplete the midlatitude ozone layer measurably.

Acknowledgements

First and foremost, I would like to thank my supervisor Johan Friberg, who introduced me to this project, always helped and encouraged me, and gave me valuable feedback during the whole process.

In addition, I want to thank my co-supervisor Carl Svenhag, who reviewed my thesis and helped me to improve it immensely.

Last but not least, I would like to thank my family and friends for always supporting me in these stressful days of writing this thesis.

Contents

List of abbreviations

PSC polar stratospheric cloud

UVC ultraviolet C

UVB ultraviolet B

UVA ultraviolet A

TTL tropical transition layer

pyroCb pyrocumulonimbus cloud

NASA National Aeronautics and Space Administration

MLS Microwave Limb Sounder

1

Introduction

With the Montreal Protocol from 1987, the dramatic destruction of the Earth's protective ozone layer was addressed, and it was agreed by 198 countries to rule out the production of many of the strongest ozone depleting substances. A study confirmed the recovery of the ozone layer for the time period between 2004 and 2016 due to the Montreal Protocol [1], and the ozone layer over Antarctica is expected to fully recover by the 2050s. [2]

However, there are possible threats to this positive development. Substances that might cause ozone depletion can be transported to the stratosphere, where the ozone layer sits, by natural phenomena like volcanic explosions or large wildfires. Eruptions such as the one of Calbuco in Chile in April 2015 invoked a notable ozone destruction [3]. On the other hand, wildfires are likely to become more frequent in the future due to global warming. They transport smoke particles and trace gases into the stratosphere that can influence the ozone cycle. This could slow down the healing of the ozone layer or even revert its recovery. [4]

The ozone layer inside the stratosphere protects the Earth with all its living organisms from the harmful ultraviolet radiation. With increased destruction of this protective shield, humans could, for example, face severe increases in skin cancer cases, and entire ecosystems could suffer large damages. [5]

Previously, studies on wildfire-induced ozone destruction were mostly focused on the southern hemisphere and the big wildfires in Australia. Even though there is a slight increase in midlatitude stratospheric ozone reported since the late 1990s [6], studies found negative perturbations in the midlatitude stratospheric ozone abundance after the 2019-20 Australian bushfire season, which was called The Black Summer [3, 4, 7]. This fire was the biggest wildfire recorded in the 21st century, with a total area of around 243,000 km^2 burnt [8] and smoke particle injections into the stratosphere on 29-31 December 2019 and 4 January 2020 [9]. However, there are other wildfires worth considering, for example, the 2021 wildfires in Russia with a total burnt area of 62,600 km^2 [10], or the 2017 Canadian wildfire for which, by the end of the fires, around 12,000 km^2 [11] were burned in the province of British Columbia. It was decided to take the latter as the example for the northern hemisphere since the total aerosol mass that was injected into the stratosphere was one of the biggest in recent history, but this will be further elaborated in section 5.2.1. For this fire, smoke particles were lifted up into the stratosphere on 12 August 2017 [9]. The northern hemisphere, especially the northern midlatitudes, is also of particular interest to investigate since (as of 2011) 50% of the world's population live within the latitudes 20° and 40° [12] (positive latitude values in degree are in the northern hemisphere and negative in the southern hemisphere) and an ozone layer depletion in this region would, therefore, directly affect most of the world's population.

On the other hand, there are plans for geoengineering the stratosphere with sulfur dioxide (SO_2) in order to offset the warming effect of greenhouse gases [13]. The added SO_2 could increase the aerosol abundance in the stratospheric aerosol layer, which would reflect more of the incoming sunlight and, hence, cool the Earth's surface [13]. Investigating the stratospheric SO_2 changes and possible impacts on ozone for wildfires could therefore help predicting risks for depleting ozone by geoengineering.

In addition, the overall gained understanding could also help predicting impacts of other events that propel large amounts of substances that influence ozone into the stratosphere, like nuclear detonations. Since nuclear weapons exist, people have threatened to use them. Even for a locally restricted conflict with tactical nuclear weapons, possible ozone loss following the detonations would influence the whole planet. Model calculations predict a 20% global ozone loss in a time span of 5 years for a nuclear conflict involving 100 Hiroshima-size bombs [14]. In addition to the terrible aftermath of a nuclear conflict, skin cancer rates all over the globe would increase, food production would suffer, and famines could follow. Knowing the potential consequences of actions like geoengineering and nuclear detonations is, therefore, of great importance. This thesis is aiming for adding knowledge to this field.

To introduce the most essential background knowledge, the report continues with a brief theory part on ozone, its chemistry, and the measurement system used in chapter 2. Then, the aims of this thesis are presented in chapter 3; after that, the used methods are explained in detail in chapter 4; and next, the data is analyzed, divided into results and discussion for the selected wildfires in chapter 5. Finally, the thesis is finished with a conclusion and outlook in chapter 6.

2

Theory

2.1 The stratosphere

The atmosphere consists of multiple different layers, which are shown in Fig. 2.1. Only the lowest two layers are of importance in this thesis and are, therefore, explained further. The lowest layer, the so-called troposphere, reaches from the Earth's surface up until altitudes between 10 and 12 km. Most weather phenomena occur here, and about 75% of the atmospheric mass and nearly all of the atmospheric water is located within it. With rising altitude, the temperature drops (see Fig. 2.1) with about 6.5°C/km. This, in turn, results in the uplifting of warm air from the ground until it reaches the tropopause. This is the boundary layer between the troposphere and the next atmospheric layer, the stratosphere, and its altitude varies throughout the year but mainly depends on the latitudinal location. Beginning in the stratosphere, the temperature development is reversed. With increasing height, the temperature increases as well. From the tropopause, the stratosphere reaches up to about 50 km. Compared to the troposphere, the stratosphere is very dry and, therefore, there are no clouds to be found, except under the right conditions, there can be polar stratospheric clouds (PSCs) (more in section 2.3) [15]. Due to the missing precipitation and low vertical motion, substances stay inside this layer for very long periods of time. Vertical motions in the stratosphere require diabatic processes (heating or cooling). Most importantly, the stratosphere contains the ozone layer, which is further elaborated in the following section. [16]

2.2 Ozone

Ozone is a molecule made out of 3 oxygen atoms and is chemically referred to as O_3 [18]. As a gas, it is of a light blue color with a distinct smell [19]. It is a much stronger oxidant than the more common dioxygen (O_2) and, therefore, can be used for oxidation in many industrial and consumer applications [20]. However, the inhalation of ozone has negative health effects on humans, which in particular means that it can damage the cardiovascular, respiratory, and central nervous system [21]. For plants, ozone can hinder their growth and, thereby, can reduce the yield of crops [22]. For example, a study revealed that in East Asia, a total of 63 million U.S. dollars worth of crops is lost every year due to ozone pollution resulting from fossil fuel combustion [23]. Therefore, ground-level ozone concentrations should not exceed 60 $\mu g/m^3$ according to the WHO guidelines from 2021 [24]. Although ozone is seen as a pollutant when occurring in the troposphere, stratospheric ozone is crucial for life on Earth [25].

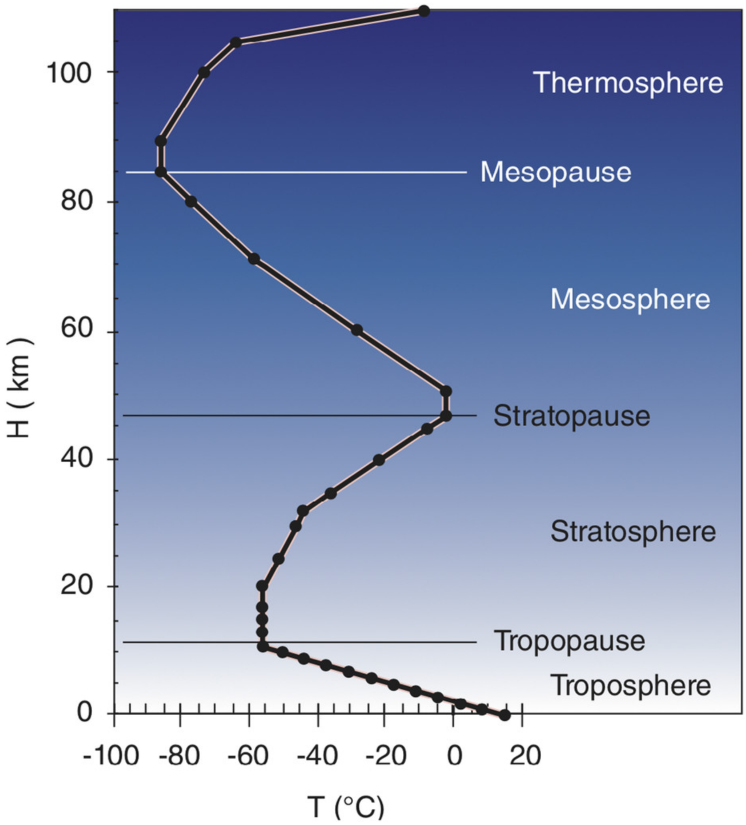

Figure 2.1: Temperature variation throughout the atmosphere and its according layers. Figure is acquired from [17].

The Sun's radiation that reaches Earth consists mostly of wavelengths from 100 nm to about 1 mm [26]. Those wavelengths are categorized into ultraviolet light from 100 to 400 nm, visible light from 380 to 700 nm, and infrared light from 700 nm to 1 mm [26]. In addition, ultraviolet light is subdivided into ultraviolet C (UVC) from 100 to 290 nm, ultraviolet B (UVB) from 290 to 320 nm, and ultraviolet A (UVA) from 320 to 400 nm [27].

For these three ultraviolet spectral ranges, stratospheric ozone comes into play. UVC light does not reach the Earth's surface since it is strongly absorbed by oxygen and ozone. Only a very small fraction of the UVA radiation gets blocked by the atmosphere and was found to cause cancer by indirectly damaging DNA [28]. The UVB range is the most dominant in its biological effects. On the positive side, it stimulates the production of Vitamin D for humans, but it can also damage DNA directly and result in sunburn and melanoma (skin cancer). For plants, it increases plant growth but also causes damage to the cell membranes [29], which outweighs the positive effects. [5]

The ozone in the stratosphere is able to prevent UVB radiation from reaching the ground, but to what extent it can do that depends on how much ozone there is in the stratosphere [30]. This concentration is either measured in volume mixing ratio (moles of certain gas per total moles of air) or, more commonly, in Dobson units, which gives the height of the ozone in a vertical column above the Earth's surface if it was compressed to a single layer at standard temperature (298 K) and pressure (1013 hPa) [30]. A layer thickness of 0.01 millimeter is equal to one Dobson unit, which in turn holds 2.69 $\times 10^{20}$ molecules per square meter [31]. In Fig. 2.2, it can be seen that with a stronger ozone column of 360 Dobson units compared to 270 and 180 Dobson units, the irradiance at

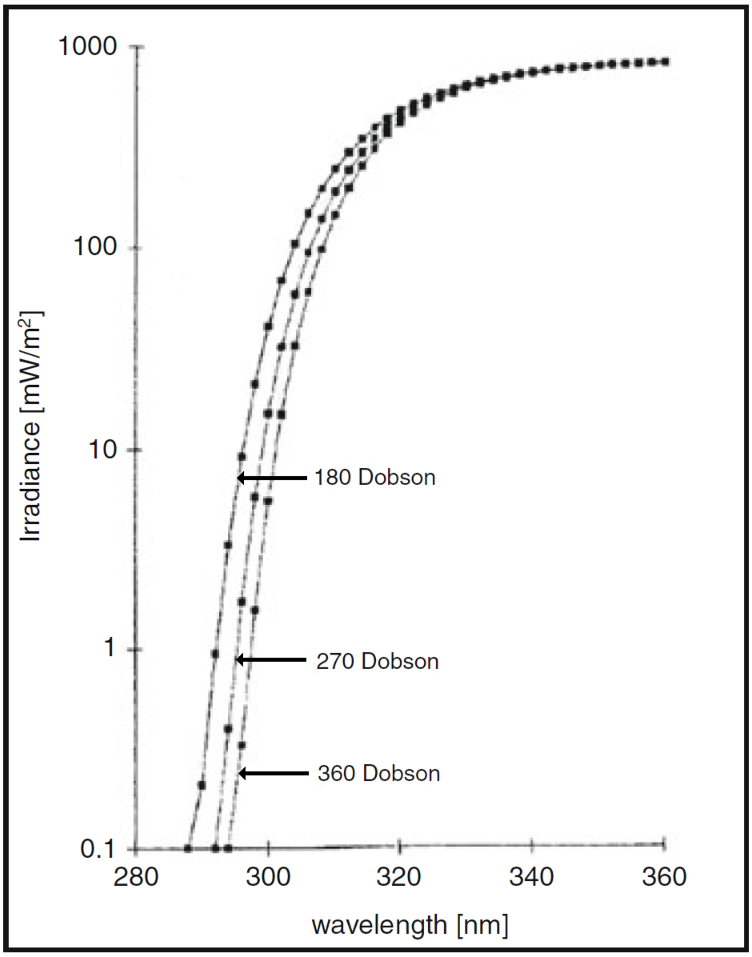

Figure 2.2: Irradiance-wavelength diagram that shows the ground level irradiance from the Sun for different total ozone columns on 21 June at a latitude of 49°. Figure is acquired from [5].

ground level for wavelengths that belong to the UVB range is lower.

The abundance of ozone throughout the stratosphere depends on the chemical production and destruction of ozone, as well as the transport of ozone in the atmosphere and the subsequent mixing of air masses with different ozone concentrations. The destruction of ozone occurs either naturally or fueled by catalyst substances containing hydrogen, nitrogen, chlorine, and bromine. [32]

2.2.1 Stratospheric ozone formation

In the 1930s, the so-called Chapman Cycle was discovered by Sir Sydney Chapman. It consists of 4 main reactions,

$$\mathrm{O_2} + h\nu \longrightarrow 2\,\mathrm{O} \quad \lambda \leq 242\ \mathrm{nm}, \tag{2.1}$$

$$\mathrm{O} + \mathrm{O_2} \xrightarrow{\mathrm{M}} \mathrm{O_3}, \tag{2.2}$$

$$\mathrm{O_3} + h\nu \longrightarrow \mathrm{O} + \mathrm{O_2} \quad \lambda \lesssim 336\ \mathrm{nm}, \tag{2.3}$$

$$\mathrm{O} + \mathrm{O_3} \longrightarrow 2\,\mathrm{O_2}, \tag{2.4}$$

which ensure a steady concentration of ozone in the stratosphere. They result in the constant formation and destruction of ozone. First, an oxygen molecule is split into two single oxygen atoms (O) by absorbing UV radiation from the Sun (Eq. 2.1). Then those single oxygen atoms combine with oxygen molecules to form ozone while requiring another molecule for this reaction to conserve energy and momentum (Eq. 2.2). Because sunlight is needed for ozone production, it is strongest around the equator, where the sunlight hits the surface at a 90° angle and, hence, has to heat a smaller area compared to other latitudes polewards. For the destruction, on the other hand, UV light of a different wavelength splits ozone into an oxygen molecule and a single oxygen atom (Eq. 2.3). The oxygen atom then attaches to molecular oxygen to create ozone again (Eq. 2.2) or reacts with an ozone molecule in order to form two oxygen molecules (Eq. 2.4). While the Chapman Cycle overestimates the production of ozone in the tropics, it provides a basic understanding of the ozone creation and destruction cycle [32]. [33]

2.2.2 Stratospheric ozone depletion

The destruction of ozone via catalysts can be categorized into three cycles. The first cycle consists of two or more reactions in which the catalyst is restored in the end. As a result, one ozone molecule reacts with one atomic oxygen to form molecular oxygen:

$$\mathrm{XO} + \mathrm{O} \longrightarrow \mathrm{X} + \mathrm{O_2}, \tag{2.5}$$

$$\text{Cycle 1:} \qquad \mathrm{X} + \mathrm{O_3} \longrightarrow \mathrm{XO} + \mathrm{O_2}, \tag{2.6}$$

$$\text{Net: } \mathrm{O} + \mathrm{O_3} \longrightarrow 2\,\mathrm{O_2}. \tag{2.7}$$

Thus, a catalyst (X in the reaction) of that sort can deconstruct thousands of ozone molecules before it finally leaves the stratosphere and enters the troposphere, where it is eventually washed out by precipitation. [32]

These catalysts stem from source gases of natural or human emission that are turned into reactive gases by sunlight. One family of those reactive gases working as catalysts are HO_X radicals (including hydrogen (H), hydroxyl radical (OH), and hydroperoxyl radical (HO_2)) that are created from water (H_2O), hydrogen, or methane (CH_4) reacting with single oxygen atoms. Another type of radical powering cycle number one is nitric oxide (NO). It is either produced by the reaction of nitrous oxide created by bacteria in soils with single oxygen atoms or by the splitting of nitrogen molecules by sunlight. Counteracting the ozone depletion by nitric oxide is the transformation of nitrogen oxides into less active reservoir gases like nitric acid, chlorine nitrate, and bromine nitrate. The last group of catalysts for the first cycle are reactive halogen gases chlorine (Cl), bromine (Br), and their monoxides (ClO and BrO). The halogens are stored in reservoir gases as hydrogen chloride (HCl) that are converted to their reactive forms in the upper stratosphere. A large influence on the chlorine content of the stratosphere were chlorofluorocarbons which were manufactured industrially. They were ruled out with the Montreal Protocol by all members of the United Nations in 1987 because of causing massive ozone destruction. [32]

In the polar regions, two other cycles dominate ozone destruction:

$$ClO + ClO \xrightarrow{M} (ClO)_2, \quad (2.8)$$
$$(ClO)_2 + h\nu \longrightarrow ClOO + Cl, \quad (2.9)$$
$$\text{Cycle 2:} \quad ClOO \xrightarrow{M} Cl + O_2, \quad (2.10)$$
$$2\,(Cl + O_3 \longrightarrow ClO + O_2), \quad (2.11)$$
$$\text{Net: } O_3 + O_3 \longrightarrow 3\,O_2, \quad (2.12)$$

$$ClO + BrO \longrightarrow Cl + Br + O_2, \quad (2.13)$$
$$\text{Cycle 3:} \quad Cl + O_3 \longrightarrow ClO + O_2, \quad (2.14)$$
$$Br + O_3 \longrightarrow BrO + O_2, \quad (2.15)$$
$$\text{Net: } O_3 + O_3 \longrightarrow 3\,O_2, \quad (2.16)$$

This is due to the missing single atomic oxygen, which hinders cycle 1, and an enhanced occurrence of ClO due to reactions on the PSC surfaces (more in section 2.3). In cycle 2, the ClO produces ozone depleting Cl radicals by reacting with itself, whereas, in cycle 3, ClO reacts with BrO and creates Cl and Br that subsequently destroy ozone molecules. Only in winter times do the temperatures allow the creation of enough ClO (via PSCs) to fuel these two cycles. Because reservoir gases are being split into radicals through sunlight in the uppermost stratosphere, the halogen destruction cycles dominate there. In addition, they are the prevalent cycles in the lower stratosphere because of PSCs. Finally, the NO_X loss cycle dominates the middle, and the HO_X cycle dominates the lowermost stratosphere. [32]

2.2.3 Ozone transport

Because of the steady ozone formation through the Chapman Cycle and the loss of ozone through chemical reactions with catalyst gases, the ozone layer is formed, which has its maximum volume mixing ratio in the tropics between 30 and 35 km altitude. The black arrows in Fig. 2.3 show the flow of ozone in the stratospheric circulation from the tropical region towards polar regions as well as down into lower stratospheric altitudes during winter seasons. In addition, mixing takes place in the lower stratosphere between latitudes. The figure, moreover, shows the latitudinal varying height of the tropopause, which is marked with the black dashed line, having its maximum altitude around the tropics and getting lower polewards. Therefore, the height of the ozone layer ranges from 10 to 15 km in higher latitudes and 20 to 25 km around the equator. Although produced in the tropics at high altitudes, due to transportation, the highest ozone mass is found between the midlatitudes and the poles, about 10 km lower in altitude. Therefore, if the ozone mass is integrated vertically (usually measured in Dobson units), it also reaches its maximum there. [32]

2.3 Polar stratospheric clouds

As mentioned in section 2.1, there are no clouds in the stratosphere. In the lower part, though, there exists a thin layer of liquid supercooled aerosol droplets consisting mostly

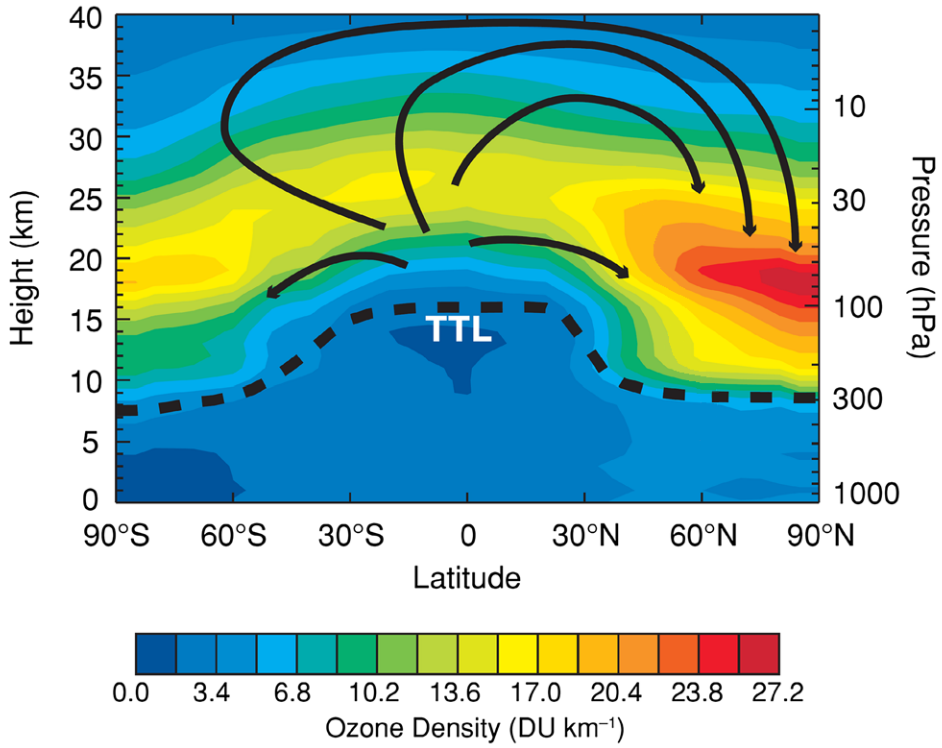

Figure 2.3: Color map of ozone density in Dobson units per km for height/pressure over the latitudes from January until March, which in addition highlights stratospheric circulation with black arrows, the tropopause with a black dashed line, and the TTL stands for the tropical transition layer. Figure is acquired from [32].

of sulfuric acid. Under the right conditions, these droplets have vapors condense on them and form PSCs. There exist two types of PSCs, with type I forming at temperatures below 190-195 K having liquid particle sizes of about one micrometer and type II forming below 185-187 K having water-ice crystals of about 10 micrometers in diameter. Altogether, these conditions are predominantly found during winter over the polar regions inside a polar vortex (latitudes north of 60° and south of -60° [34]) and, therefore, do not occur in the midlatitudes, which are the focus of this study. However, the reactions on the particles of the PSCs play an important role in the massive ozone depletion over the polar regions. Firstly, there are chlorine activation reactions (see Eq. 2.17, 2.18, and 2.19) that transform inactive reservoir gases such as chlorine nitrate ($ClONO_2$) and HCl into chlorine molecules that can be split into chlorine radicals by sunlight:

$$ClONO_2 + HCl \longrightarrow Cl_2 + HNO_3, \tag{2.17}$$

$$ClONO_2 + H_2O \longrightarrow HOCl + HNO_3, \tag{2.18}$$

$$HOCl + HCl \longrightarrow Cl_2 + H_2O, \tag{2.19}$$

Secondly, reactions exist that remove nitrogen oxide gases by transferring them into nitric acid inside the cloud particles. Otherwise, those nitrogen oxides would bind to ozone depleting radicals and, thereby, interfere with the catalytic ozone cycles. [15]

2.4 Wildfires

Due to climate change, subsequent droughts, and extreme heat, the frequency and intensity of wildfire events are projected to increase [35]. Therefore, in addition to their immediate damages, possible effects on the atmospheric composition following the fires would increase as well since they release black carbon, organics, and different trace gases into the atmosphere.

2.4.1 Smoke transport to the stratosphere

Large wildfires can create so-called pyrocumulonimbus clouds or short pyroCbs. They are thunderstorms caused by the uplift of air with water, black carbon, organics, and other trace gases because of the heat of the fire. [36] The air rises, expands, and cools down due to the uplifting and subsequently forms clouds by condensation. Like conventional thunderstorms, they can produce lightning, which can initiate more fire, but they lack precipitation. The heat from the fire lifts the formed clouds even past the point of condensation, and because of extra heat released by the condensation, they can even pass the tropopause and reach the stratosphere. [37]

The wildfire emission products can survive there for months. They form layers of only a few km in thickness that can organize in bubble-like structures of a thousand km in diameter [38], which are maintained compact by a vortex. The black carbon absorbs a lot of sunlight, and the produced heat, therefore, lifts the layer and consequently keeps it in the stratosphere for a larger amount of time. [36]

2.4.2 Resulting stratospheric chemistry

In stratospheric altitudes, normally already a layer with sulfuric acid and water particles exists, playing an essential role in the ozone destruction process. Although not yet completely understood, studies suggested that soot particles and other organic compounds released by wildfires become coated with sulfuric acid [7, 39]. They, therefore, can enhance the surface area of the aerosol particles in the aforementioned layer, similar to the chemistry invoked by volcanic eruptions, which release additional sulfur and, thereby, also increase this surface area. The reactions on the surfaces of those acid-covered particles have not yet been studied in the laboratory [4], but they influence the abundance of reactive nitrogen, which includes NO and NO_2. These two species normally transform at daytime into one another, and these reactions depend on ozone concentration, temperature, and photolysis rates, but if both of the nitrogen species reduce due to the particle surface reactions, the abundance of ClO and OH radicals is affected. All of those are part of the catalytic ozone destruction cycles (see section 2.2.2), and this change in the chemistry is expected to cause stratospheric ozone loss. [40]

Other trace gases playing a role in wildfire chemistry also analyzed in this thesis are SO_2, chloromethane (CH_3Cl), and HCl. The concentration of sulfur in dry plants varies between 0.1% and 0.9%. When burnt, the plants will release this sulfur in the form of SO_2 [41]. CH_3Cl as well is released during the combustion of biomass [42] but results in the increase of chlorine radicals due to photochemical separation into methyl radicals and atomic chlorine via UVC radiation [43]. HCl, on the other hand, is not emitted by wildfires but is very easily soluble in the aforementioned acidic, hydrated soot particles [44].

As already mentioned in section 2.1, there is minimal mixing vertically and no precipitation in the stratosphere. Therefore, once injected into the stratosphere, the substances will form layers and stay there for longer times. [33]

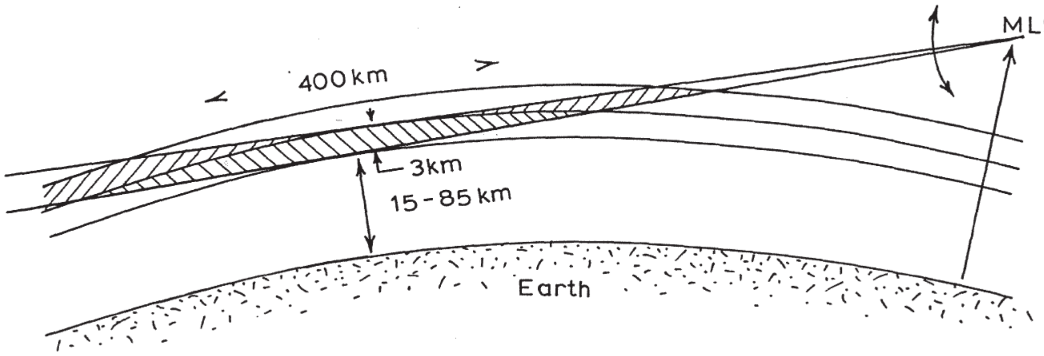

Figure 2.4: Measurement geometry of the MLS instrument, where the tangent height is measured at the closest passing point of the ray to the Earth's surface. Figure is acquired from [48].

2.5 Microwave Limb Sounder

The Aura (formerly EOS CH-1) satellite is a scientific research satellite from NASA that orbits the Earth at a 710 km altitude, is tilted 98.22°, and has a near-global coverage [45]. It followed the Upper Atmosphere Research Satellite but is more focused on the lower stratosphere and troposphere [46]. Aboard there are four instruments: a High Resolution Dynamics Limb Sounder, an Ozone Monitoring Instrument, a Tropospheric Emission Spectrometer, and a Microwave Limb Sounder (MLS) [46]. The last of which, or rather the data from this instrument, is used for the analysis in this thesis.

The MLS is used for recording the abundance of several different trace gases, as can be read in the MLS data manual by the Jet Propulsion Laboratory at the California Institute of Technology: BrO, CH_3Cl, CH_3CN, CH_3OH, ClO, CO, H_2O, HCl, HCN, HO_2, HNO_3, HOCl, N_2O, O_3, and OH and SO_2. The temperature, the geopotential altitude, as well as two more water abundance related parameters are also measured. All of the substance parameters are recorded in volume mixing ratios as functions of pressure. [46]

Limb sounding itself is a popular remote sensing technique that uses airborne or spaceborne instruments to look at the edge of the atmosphere, which means measuring parallel to the planet's surface in order to detect emitted or scattered radiation from between the upper troposphere (about 10 km) and the mid-thermosphere (about 450 km). Compared to passive nadir sounding, which views straight down, the limb sounding technique allows for a higher vertical resolution. Other obstacles, such as the planet's surface being able to reflect or emit radiation, have negligible impact on limb sounding, and with the longer path through the atmosphere (around a few 100 km), the signal-to-noise ratio is improved, but in turn, the horizontal resolution suffers. [47]

The so-called tangent height is the distance between the measurement beam and the surface of the Earth at the closest point (see Fig. 2.4). The higher the tangent height, the lower usually the signal since the density of the atmosphere decreases exponentially for higher altitudes. When going further down, the density gets higher, and therefore either the emission or the scattering increases, which enhances the signals. If the tangent height is too low, the atmosphere eventually becomes too dense, and therefore, no signals from further down can be picked up by the instrument because they are absorbed by layers on top. Thus, signals from lower tangent heights stay constant at that point. [47]

There is a spectral line broadening from the molecules through a combination of two processes. On the one hand, there is Doppler broadening because of the molecules' thermal movement, and on the other hand, there is pressure broadening because of molecules colliding with each other. For microwave signals, the second one is dominating up to heights of 60 km. This line broadening holds additional vertical information about the composition since signals that are farther away from central wavelengths can stem from lower altitudes with larger broadening. Additional information can also be gathered from the varying absorption of different spectral regions. [47]

Microwave Limb Sounding observes thermal emissions of wavelengths between 5 mm (or 60 GHz frequency) and 120 μm (or 2.5 THz frequency) to determine molecular rotational transitions of atmospheric substances that have enough dipole moment. Only very dense clouds can interfere with these signals because, usually, the wavelengths are much longer than the sizes of the particles. Due to its high frequency resolution, line shapes can be resolved very well by Microwave Limb Sounding. The broadening of the line shape can give information about the pressure since higher pressures increase the pressure broadening, while the strength of the line correlates with the abundance of the species. Using the hydrostatic balance, the pressure information, together with the tangent height, can then be used to calculate a temperature profile. Finally, the larger the antenna size and the shorter the wavelength of a Microwave Limb Sounder is, the narrower the field of view. The used instrument aboard the Aura satellite has a 1.6 m antenna, and for 200 GHz at 700 km altitude, the visible field is 3.5 km over the tangent point. The lower observable altitude limit is situated at around 8 km altitude, where the signals are fully absorbed by oxygen, water vapor, and nitrogen. [47]

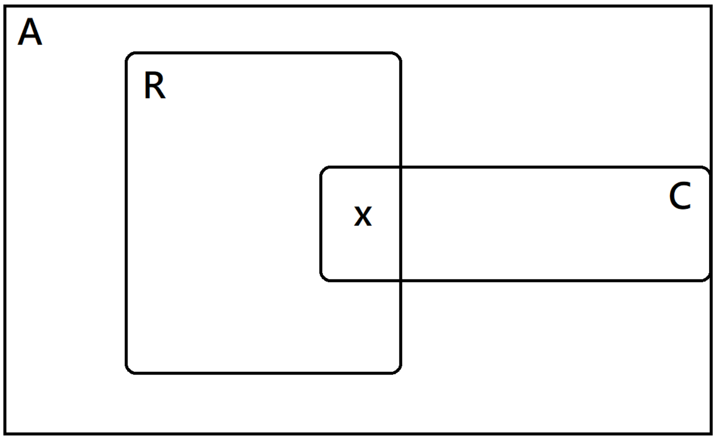

Figure 2.5: Simplified presentation of the nested sets of solutions, where A stands for the entirety of possible solutions, R stands for solutions possible from the measured signals, C stands for solutions possible from the a priori information, and x marks the optimal solution for the case of probability distributions instead of discrete sets. Figure is adapted from [48].

Only considering the radiances recorded by the instrument, though, cannot give absolute information about the species' abundances. They hold an infinite number of solutions and, therefore, a priori information at given locations and times from previous meteorological studies is needed. In Fig. 2.5, A stands for all possible solutions. R represents the set of solutions possible from the measured signals, whereas C represents the set from a priori information that has to hold the solution. As a consequence, the solution must be situated inside the intersection of R and C. When viewing these sets as probability distributions, then there must be an optimum in the intersection, which is marked with an x. However, the relative size of the intersection has to be taken into consideration since if, for example, the whole area of C was included in R, the satellite data would have contributed no additional information. [48]

3

Aims

The reason for writing this thesis is to check if the ozone layer keeps recovering and, if it does not, find what hinders the recovery because, without the ozone layer, most life on Earth would be severely affected and harmed to an unimaginable extent. Therefore, this thesis approaches three main points:

- What could be the mechanism behind the wildfires influencing stratospheric ozone? Although not yet studied in the laboratory and, therefore, not yet known in detail, there are several possible theories about the mechanisms behind it.

- How are these theories applicable to the 2019-20 Australian bushfire season, and to what extent did these mechanisms influence stratospheric ozone and ozone depleting substance abundances? Different studies have already found connections between the Australian wildfire and midlatitude ozone depletion, but do the results produced here match these findings?

- Did wildfires in the northern hemisphere, and in particular the one in 2017 in British Columbia, also influence the midlatitude stratospheric ozone abundance in any way (reasons for why this one was selected as an example are explained in section 5.2.1)? The insights gained from the analysis of the Australian wildfire can then be used to evaluate the results from the Canadian wildfire.

Using satellite data from the Microwave Limb Sounder, the 2019-20 Australian fires and the 2017 fires in British Columbia can be analyzed and compared in terms of trace gas abundances and possible ozone destruction. This should then give indications about how the scale and other factors, like the type of burned materials or injection height of the wildfire smoke, will affect the fire's influence on the stratospheric ozone layer and how well-suited the instrument used is for this type of analysis. The benefit of this gained knowledge will be that predictions of possible consequences can be more precise, and precautions like bans for still allowed man-made ozone depleting substances or more public funding for fire prevention as well as fire fighting can be taken accordingly.

4

Methods

4.1 Data preparation

The data that was used for this thesis was pre-processed in two steps. The first processing step was done by the Jet Propulsion Laboratory at the California Institute of Technology. There are four different versions of the data in different processing states available from them. They are called Level-0, Level-1B, Level-2 and Level-3. The Level-0 data is the raw, time-ordered satellite data that still has the original resolution, and all packets that are duplicates are removed. From this data, the Level-1B gets produced that transforms counts to radiometrically calibrated radiances. The further processed Level-2 data is the one that was used for this thesis. It contains the actual geophysical data products that are produced from the Level-1B calibrated radiances. Level-3 data is even further processed, but it is not used here and, therefore, not explained further. The Level-2 data, of which version 5.0x was used, includes temporal, spatial, and viewing track information. It was even undergoing another processing step by the supervisor of this thesis Johan Friberg, who excluded data from for this study unnecessary species and regridded the data points. [46]

The original daily basis Level-2 data is gridded in 1.5° sections along the longitudes and latitudes of the Earth, and the height profile was split into sections of varying height but is saved in pressure data with measuring points at $10^{n/6}$ hPa [46](except temperature, water, and ozone that were more finely gridded). This was then regridded to 1° sections on the latitudes only (longitudinal resolution was given up for easier analysis) by Johan Friberg.

Further, only parts of the satellite-produced data were used. The parameters from the MLS data that were analyzed for this study, due to their connection to wildfires and ozone destruction, as already mentioned in the theory section, were O_3, HCl, CH_3Cl, ClO, H_2O, SO_2, and temperature. The used height/pressure ranges were limited by different sources of constraint. For one, there were individual useful ranges suggested in the manual for the data from the Jet Propulsion Laboratory [49], and those were set as follows: for temperature 216-0.00046 hPa, for H_2O 316-0.001 hPa, for SO_2 215-10.0 hPa, for O_3 261-0.001 hPa, for HCl 100-0.32 hPa, for CH_3Cl 147-4.6 hPa, and for ClO 147-1.0 hPa [49]. In addition, as this study only focuses on the stratosphere and in order to prevent influences from clouds, tropospheric values had to be removed. This led to the exclusion of the lowest height values for ozone at 11.31 km because they lie below the highest annual altitude of the tropopause for the later examined extratropics. There the maximum altitude of the troposphere varies between about 8 and 12 km. To ensure that only stratospheric values are used in the analysis, only heights above 12 km were considered [50]. Moreover, heights were cut from latitude ranges if they included missing data since this would break the calculation of column values and could have uncontrollable influences if data is averaged. Furthermore, for ClO, the three lowermost heights could only be used with an additional latitude-dependent bias. Hence, they were not used, and the pressure range was limited to 46-1.0 hPa [49]. In addition, ClO has a strong shift in abundance between day- and nighttime and is, therefore, hard to analyze on a daily basis since the values might fluctuate strongly depending on the time of measurement. However, due to the periodicity of the satellite's orbit, the average over longer periods of time can be used for analysis. As for the precision of the data from the MLS in the selected height ranges that can be found in the manual for the data from the Jet Propulsion Laboratory [49], it varies for each height and each recorded parameter but was found to be of sufficient height for the analysis in this study, which focuses on trends rather than absolute values. Finally, this screened data was then processed with programs written in Python that are further explained in the next section.

4.2 Programming

In order to evaluate all of the data with more ease and to extract valuable information regarding the stratospheric ozone layer, different programs were written in Python, which process the data and display it in different comprehensible ways. The codes for all four programs can be found in the Appendix. The rest of this methods section is dedicated to a thorough description of what the scripts do and how exactly they work. However, this is not essential for understanding the other parts of this thesis, but it can be used to comprehend the work process and to possibly recreate results from this thesis together with the listed code.

The first program was written to get a general understanding of the abundance of different species inside the stratosphere and how they vary with time. It created an animation that showed the recorded data directly in volume mixing ratios and displayed it via a color map plotted over height and latitude changing on a daily basis. Added below it was a bar plot for all latitudes of column values in Dobson units, also animated for each day. Column values sum up all molecules over a unit area above the Earth for a certain height range, as already explained in section 2.2. The plots from this first program were exceptionally helpful for gaining a first understanding of the spatial and temporal location of exceptionally high and low substance abundances, but none of them are shown in this report since it was an animation and only snapshots in time could have been included.

Since the programs all work with the same set of data, the first steps were basically the same for all of them. They start by extracting the data from the provided data files, masking the NaN values, and converting all temperature resolutions to the same grid by interpolation. From the pressure values, the approximate height (geo-potential altitude) values were calculated with the formula [51]:

$$\text{height in km} = 44.3076925 \cdot \left[1 - \left(\frac{\text{pressure in hPa}}{1013.25}\right)^{0.190284}\right]. \tag{4.1}$$

The volume mixing ratio values can then be converted into concentrations in molecules per cm^3 by the formula [52]:

$$c = \frac{p}{k \cdot T} \cdot C_x, \tag{4.2}$$

where c is the concentration, p is the pressure, k is the Boltzmann constant, T is the temperature, and C_x is the volume mixing ratio. Subsequently, this can be converted to Dobson units by integrating over the column's height and the conversion [53]:

$$1 \text{ Dobson unit} = 2.6867 \cdot 10^{16} \text{ molecules per square centimeter.} \tag{4.3}$$

In the part only included in the first program, the mixing ratio for a selected substance gets plotted in a color map over height and latitude. Then, the column value in Dobson units is calculated for each latitude and shown as a bar plot. The values of the two plots get updated for every new day after a certain break in a loop over a certain selected time period, and this results in an animation.

The second program is creating a color map plot that displays the deviation from the average column value over latitude and time. First, a height range, a time period, and a substance are chosen. A loop over all selected pressures is followed by a loop over the selected years. In it, all volume mixing ratios in a latitude-time matrix for one pressure are converted to Dobson units, as explained above. Next, all these matrices for each year are put one after another to get one bigger matrix with the whole selected time period. This is then repeated for each pressure in the chosen range, and these matrices are then added up. In the end, this gives the column value in the selected height range for each latitude on each day. Subtracting the average matrix value from each value in the matrix then results in the deviation from the average at each point. Finally, the color map is printed over latitude and time.

For programs number three and four, multiple plots were produced per program run, and therefore, the actual plot creation is framed by loops over the selected substances and years, followed by a section for saving all the produced plots. The first one of those two programs is creating a latitude range averaged and month-averaged substance concentration plotted for a certain height range over the concentration, comparing particular months between years, highlighting one year per plot. Right next to it, another graphic displays the deviation from the average over the same height range for the aforementioned highlighted year. All this is done by first selecting a height range, a time period, a set of substances, and a certain latitude range. In a loop over all the months of the year, another loop over the days of the months is embedded. Afterward, only the selected latitudes are cut out of the data set, and in a loop over these very latitudes, the values are first area weighted by latitude and converted to concentrations as explained earlier and then averaged over the selected latitudes. The daily profiles are then used to calculate the monthly average. For each month, there is one matrix into which all the height profiles of each year are saved. Next, all of them are plotted in a height-concentration diagram, in which one of the years is highlighted in a different color. A second height-concentration diagram is then created next to it, and the difference between the highlighted year's profile and the average of all years gets calculated and displayed in it.

The last of the programs creates a latitude range averaged substance column value for every day plotted over an annual axis comparing the chosen years. Again, a height range, a time period, a set of substances, and a certain latitude range are selected for this first. Similar to program three, loops over the years, the months, and then the days are used to break down the data sets, but here the months are not averaged; only the latitudes are again weighted and then averaged over the selected range. Column values in Dobson units are created for all height slices and then added together to get the total column values for the height section. Finally, the months of one year are put into a bigger matrix that is then plotted for each year in a substance column over date diagram.

All figures created by the programs serve the purpose of understanding the similarities and differences between different years, latitudes, heights, and substance abundances better and comprehending interconnections between them.

5

Data analysis

5.1 Australia

5.1.1 Results

For getting to know how a massive wildfire is affecting the chemical composition of the midlatitude stratosphere, the biggest wildfire in recent history was analyzed with data available from the MLS instrument. With a total area of around 243,000 km^2 burnt, the 2019-20 Australian bushfire season, or Black Summer, was the biggest wildfire recorded in the 21st century [8].

Because of its sheer size and the resulting possibility of the produced smoke reaching the stratosphere, its effects on the stratospheric chemistry should be the most visible in the data. Furthermore, as proven by multiple studies, there was a detectable decrease in midlatitude ozone as a result of these Australian fires [4, 7]. In addition, there were 38 pyroCbs recorded [9], and it was observed that a pyroCb-created smoke patch of about 1000 km in diameter confined by a vortex traveled over 66,000 km, rose up to stratospheric heights of about 35 km [38] until it finally dissolved at the end of February 2020 [36].

The fires started around mid-June 2019, peaked between the end of December and the beginning of January, with smoke particle injections into the stratosphere on 29-31 December 2019 and 4 January 2020 [9], and were finally over in May 2020 [54, 55]. The fires occurred throughout most of the country, but the southeast was affected the strongest, namely the states of Victoria and New South Wales [54].

The ozone values are the first to be analyzed. To begin with, all latitudes are represented in Fig. 5.1, showing the derivation from the average ozone column for all available latitude ranges from 12.49 to 33.27 km over time. It must be mentioned, even though the region is not of particular interest for this study, that this height range includes some missing values around the equator and some tropospheric values close to the equator due to the latitudinal changing height of the tropopause (see black dashed line in Fig. 2.3). In addition, there are missing latitudes due to the regridding mentioned in section 4.1, which are visible as white lines in this kind of plot. According to this figure, there is a clear annual and latitudinal pattern visible. Around the equator at latitude 0°, ignoring the missing values, the ozone layer is about -50 Dobson units from the global average and only fluctuates very slightly throughout each year. Towards the poles on both sides of the equator, the fluctuation from the average ozone column increases to its maximum. The most northern latitudes at 82° thereby have a higher deviation towards positive values of up to 140 Dobson units from the average and only about -40 Dobson units in the negative

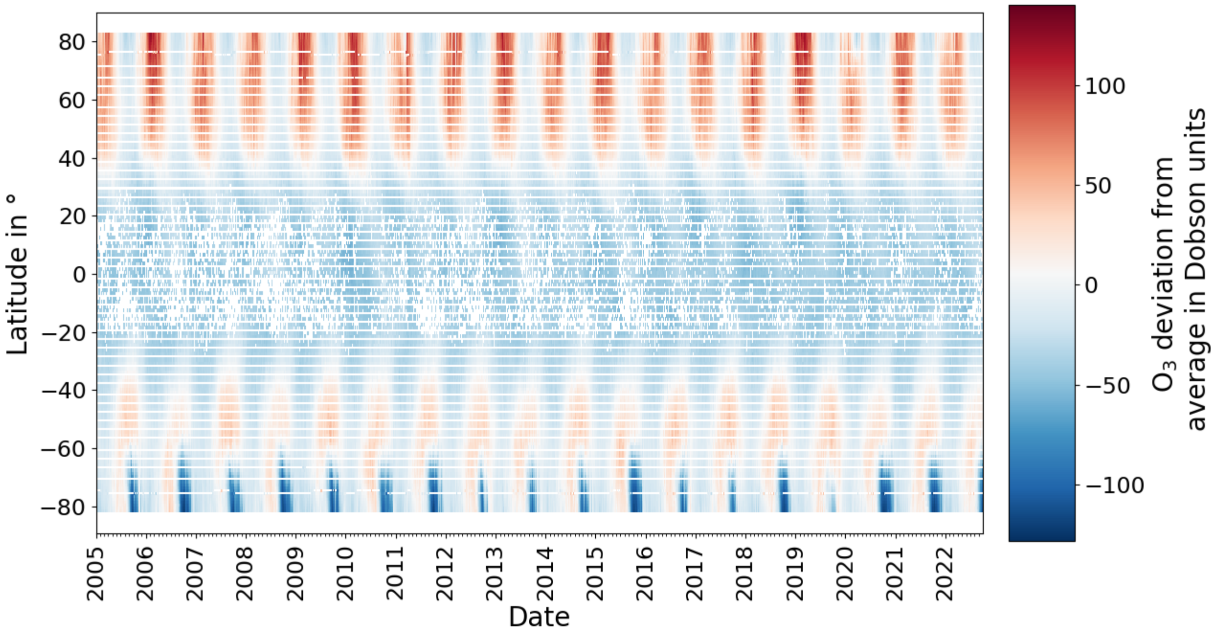

Figure 5.1: Daily plotted deviation from average ozone column color map with data from the MLS for all latitudes between -82° and 82° over time from the beginning of 2005 until end of September 2022 in the height range from 12.49 to 33.27 km.

direction. Towards the south, the latitudinal average first increases before decreasing once more at the most southern latitudes close to -82°. There, the negative fluctuation reaches down to -130 Dobson units, and the positive deviations only approach approximately 40 Dobson units over average. The annual column pattern at the maximum latitude in the north at 82° has a maximum at the beginning of each year and drops towards the end of the year into a minimum. This further applies to the -82° latitude, but the minimum is significantly lower. In addition, an elevation appears in the middle of the year, being even larger than the one in the first half of the year. Between -60° and -30° another pattern becomes visible that contradicts the ones at the latitudes closer to the poles. Each year starts with a minimum about as large as the minimum at 82° and ends with a maximum of about 50 Dobson units above average.

The latitude regime that contains the area of the Australian 2019-20 fires spans from -60° to -45° in the southern hemisphere. There is an annual pattern with a maximum in the second half and a minimum in the first half of each year. In Fig. 5.2, which is a close-up of that very latitude range, it is possible to see that the maximum for all latitudes in 2020 is smaller than the three years before 2020, very similar to the drop from 2014 to 2015. This is better visible in the ozone column representation averaged over the latitudes from -60° to -45° in Fig. 5.3. The 2020 ozone values start off slightly higher than the average of the former years in January and February and then in March continue fluctuating centered around the average. In June and the following months, the column is constantly below the average at the lower end of the previous fluctuations, with a maximum deviation from the average in the month of August with a deviation from the average of about -9 Dobson units. Followed by this slight dip is a prolonged period of low ozone values that continue to be below average until the end of the year. The year 2015 follows along very similarly, just deviating with a small peak in July and a dip in September. In Fig. 5.4, the average ozone concentration of August is displayed in dependence on the altitude. The graph on the right of it shows the deviation of the year 2020 from the average ozone concentration

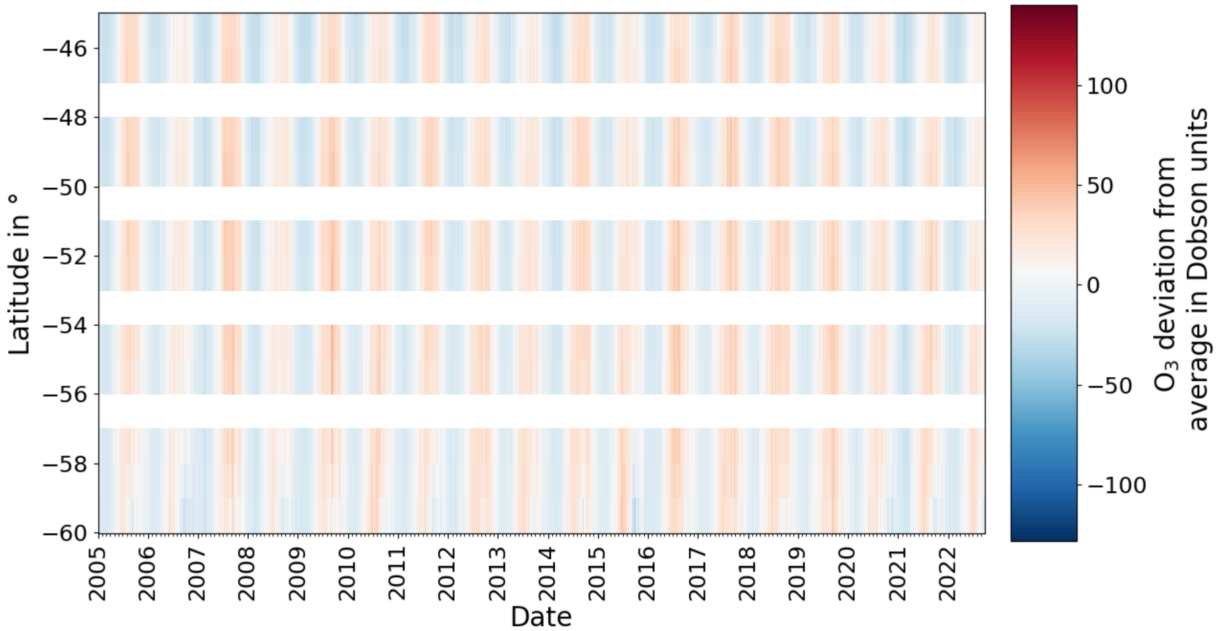

Figure 5.2: Daily plotted deviation from average ozone column color map with data from the MLS for the latitudes between -60° and -45° over time from the beginning of 2005 until end of September 2022 in the height range from 12.49 to 33.27 km.

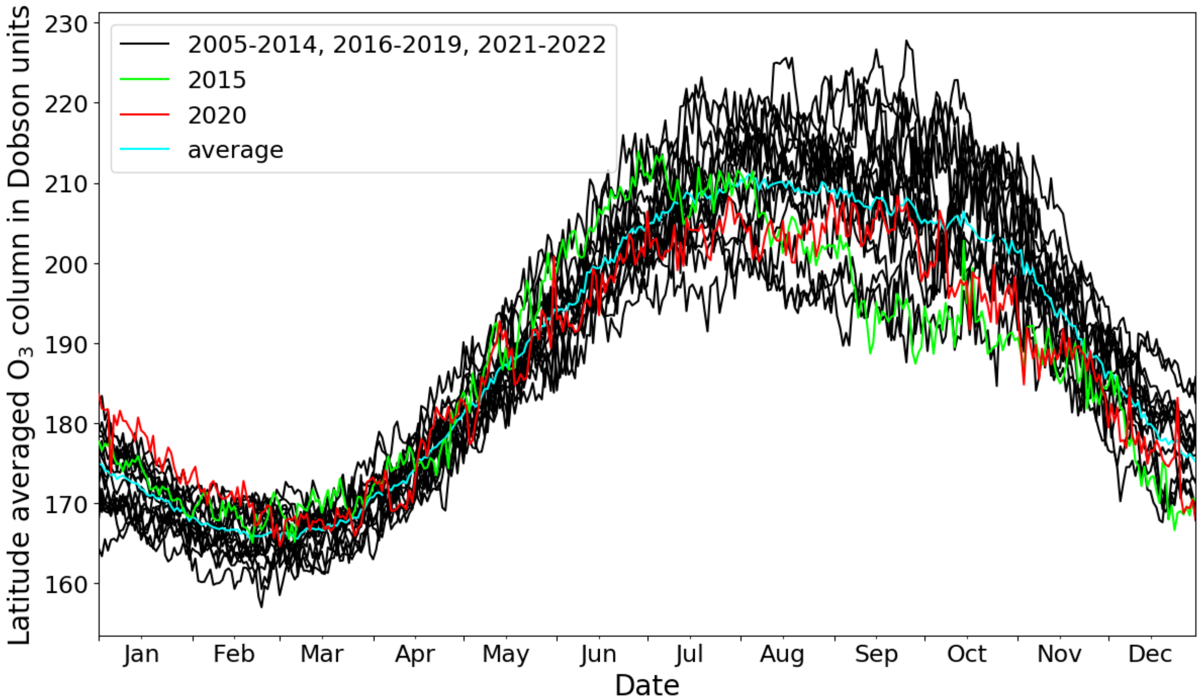

Figure 5.3: Plot of latitude range (from -60° to -45°) averaged substance (O_3) column values between 12.49 and 33.27 km for every day over an annual axis comparing all years. Highlighted in green is 2015, in red 2020, and in blue the average of all years. Created using MLS data.

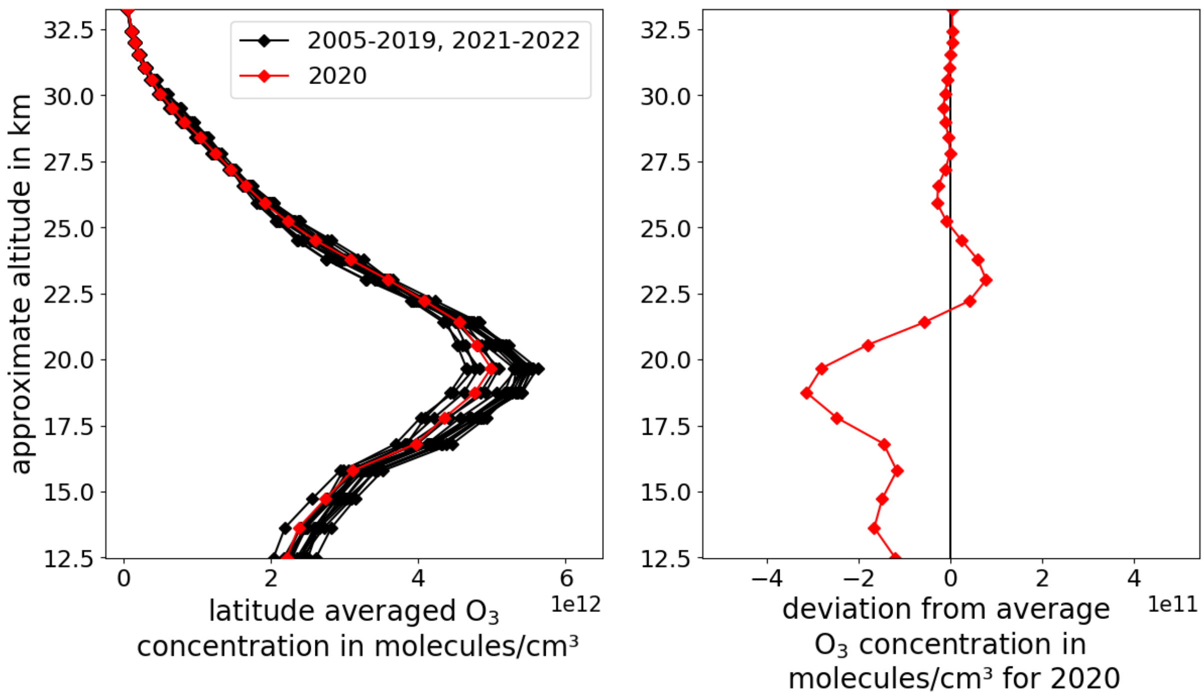

Figure 5.4: Latitude range (from -60° to -45°) averaged and month (August) averaged substance (O_3) height-concentration plot for years 2005 to 2022 highlighting the year 2020 in red for altitudes between 12.49 and 33.27 km (left) and a plot with the deviation of August 2020 from the average of August from all years over the same altitude range (right). Created using MLS data.

from all years for each altitude. It is shown that, in 2020, at lower altitudes, the ozone concentrations have strongly fallen below the average and have a maximum deviation from the average at about -3.5 $\times 10^{11}$ molecules/cm^3 at the height of 18.75 km. This negative deviation trend continues upwards until 22.22 km, where the values start to exceed the average and reach a maximum positive deviation of about 1 $\times 10^{11}$ molecules/cm^3 at 23.01 km. Above 25.22 km, little to no deviation exists. In total, the negative outweighs the positive deviations, and this is in agreement with the total column value in 5.2. However, most of the deviation is found at lower altitudes. Similar behavior for the height profile, but flipped in sign can be seen for the initial high ozone column in January as seen in Fig. 5.3. Its maximum positive deviation from the average from the same years of about 3 $\times 10^{11}$ molecules/cm^3 lies at 13.63 km.

Checking the patterns for the other gases recorded by the MLS instrument, SO_2 can clearly be excluded from further analysis since there is no annual pattern visible, and the daily column, as well as the monthly average plot, reveal a steadily decreasing SO_2 concentration for the available height range in any latitude from 2005 to 2022. There are no anomalies visible except the explained steady year-by-year decrease, and therefore, no additional information about wildfires could be extracted.

The daily column plot of CH_3Cl for the years from 2009 to 2022 for altitudes between 15.79 and 28.41 km in Fig. 5.5 shows values fluctuating strongly day by day but centering mostly around the average and, hence, making it hard to spot any monthly trends. There is an elevation in the average concentration from January until March compared to the other years with a maximum of about 0.5 $\times 10^8$ molecules/cm^3 at the height of 15.79 km. Furthermore, there is a demotion of similar size and height from May (see Fig. 5.6) until

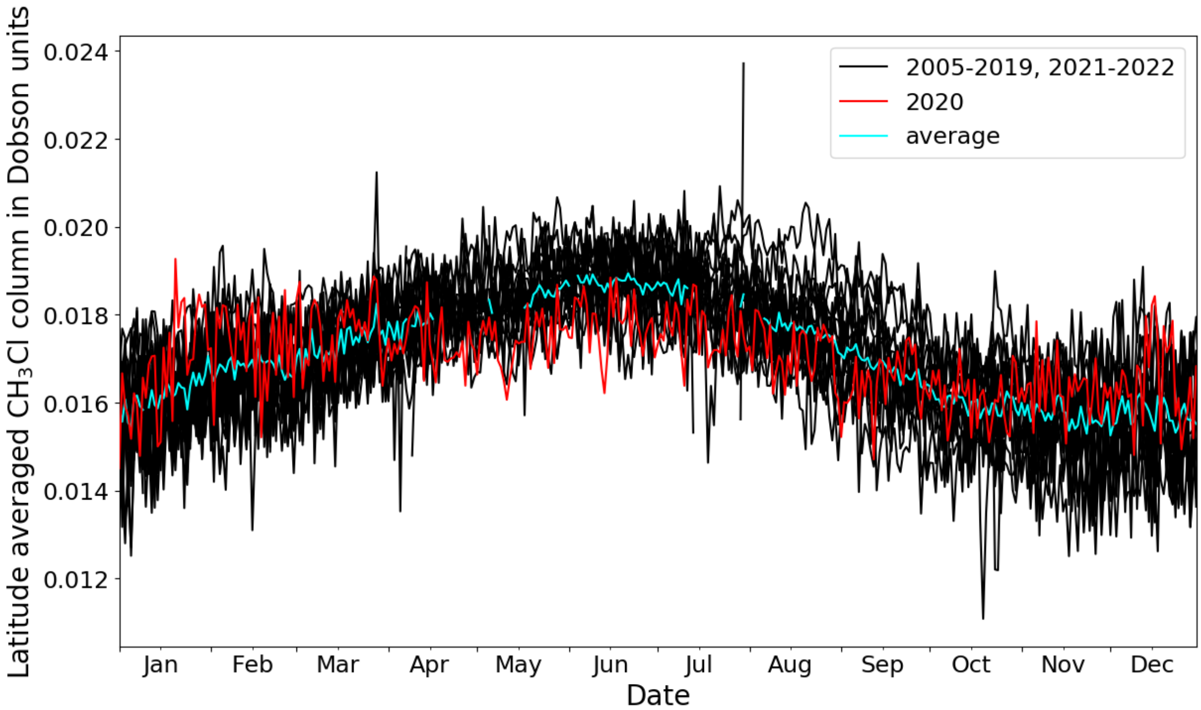

Figure 5.5: Plot of latitude range (from -60° to -45°) averaged substance (CH_3Cl) column values between 15.79 and 28.41 km for every day over an annual axis comparing all years. Highlighted in red is the year 2020, and in blue the average of all years. Created using MLS data.

September, and in December, there is about the same elevation as at the beginning of the year 2020.

In nearly the same time frame, a strong negative deviation from the average in the HCl column from 17.8 to 33.27 km altitude appears. The considerable deviation starts slightly earlier in February but also peaks in May (see Fig. 5.7). The height for the maximum deviation in May is 17.8 km (see Fig. 5.8), and compared to the other years, it is much more significant with -1.35 $\times 10^9$ molecules/cm^3 deviation from the average, competing with 0.35 $\times 10^9$ molecules/cm^3 in 2015 as the second highest one at this height.

The ClO in the height profile of May from 21.51 to 30.56 km cannot be evaluated solely by its comparison to the average, since very much as in SO_2, there is a steady decrease in ClO almost every year and the strongest overall ClO concentration can be found in 2005. Therefore, the concentrations in May 2020 were compared to the years before and after, namely, 2019 and 2021 in Fig. 5.9. In this depiction, a slight elevation of ClO in May 2020 compared to the immediate neighboring years can be seen. Compared to the neighboring months, there is also an elevation in May 2020 visible.

Regarding the H_2O, just a very slight elevation at the bottom of the height profile from 16.81 to 33.27 km in 2020 can be seen (see Fig. 5.10), but the daily column plot in Fig. 5.11 shows a relatively constant level of 5 Dobson units above the average for the year 2020. This is an elevation from the average H_2O values at the end of 2019 and the beginning of January 2020. The general trend of H_2O over the years, though slightly varying, is an increased H_2O concentration culminating in 2022, with the column reaching values of around 275 Dobson units in August, which is most likely, as studies found out, caused by an eruption of the Hunga Tonga–Hunga Ha'apai submarine volcano in the middle of January 2022 [56].

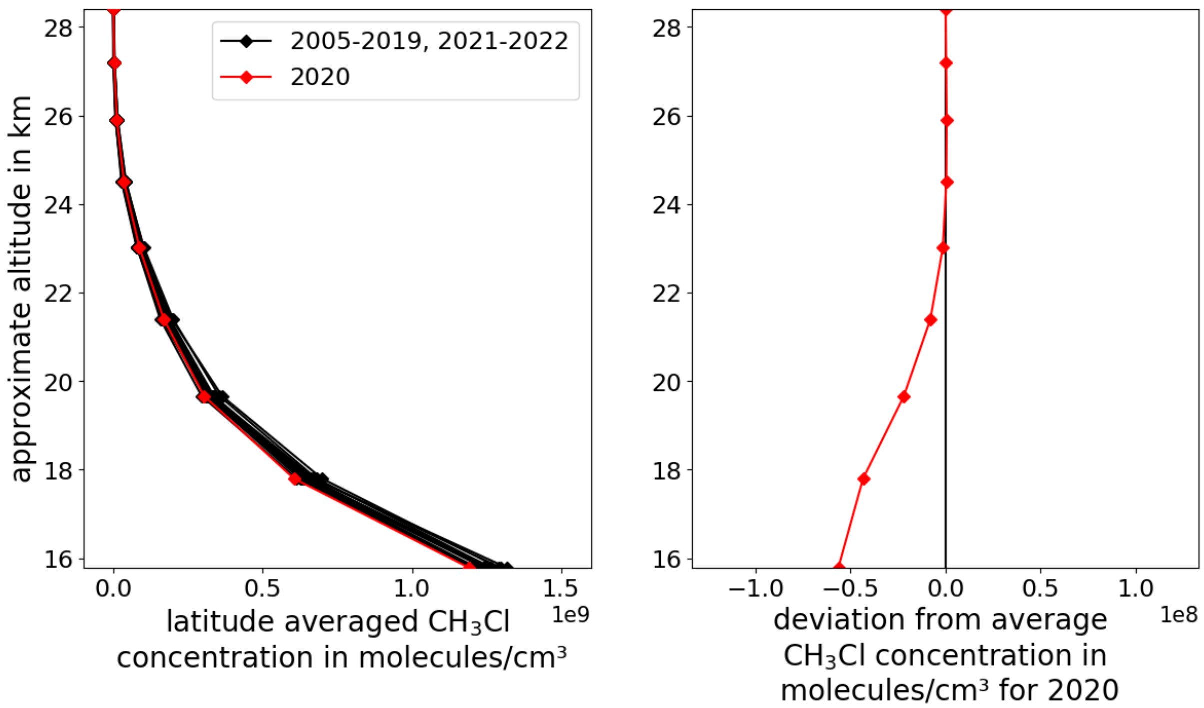

Figure 5.6: Latitude range (from -60° to -45°) averaged and month (May) averaged substance (CH_3Cl) height-concentration plot for years 2005 to 2022 highlighting the year 2020 in red for altitudes between 15.79 and 28.41 km (left) and a plot with the deviation of May 2020 from the average of May from all years over the same altitude range (right). Created using MLS data.

5.1.2 Discussion

Due to studies having found a slight increase in midlatitude stratospheric ozone since the late 1990s [6], it should be expected that the ozone in 2020 should be higher than the average. However, there is the in Fig. 5.2 found and in Fig. 5.3 and 5.4 confirmed dip in the ozone column in August 2020, starting with a decrease in March and peaking at the beginning of August in relation to the average, which follows about seven months after the peak in Australian wildfires around the turn of the year 2019-20. In the period after the fire and before the decline, the slightly elevated ozone values were most likely caused by the large number of organic molecules that were injected into the lower stratosphere by pyroCbs [4]. The tropospheric reaction of nitrogen oxides, together with volatile organic compounds and sunlight, could therefore be the driving mechanism in this ozone elevation [33], as already suggested by an earlier study [4]. Because of the hypothesized sulfate coating of these organic molecules in the stratosphere, the change does not continue after March [7]. Several studies found the depletion of ozone in the summer following the wildfire event to be existing [4, 7, 55], but it is considered to be only a minor change [7] as verified by the MLS data.

One component that must be factored in is the difference in height ranges that were used because of the different capabilities of the MLS for each substance. Ozone was recorded with the highest height range, and most substances were recorded with an upper limit of over 30 km (except CH_3Cl only up to 28.41 km). The highest lower limit was ClO with 21.4 km. Cutting the examined ranges to the smallest common section would, thus, have resulted in very little data, and vertical air motions are still always possible. There

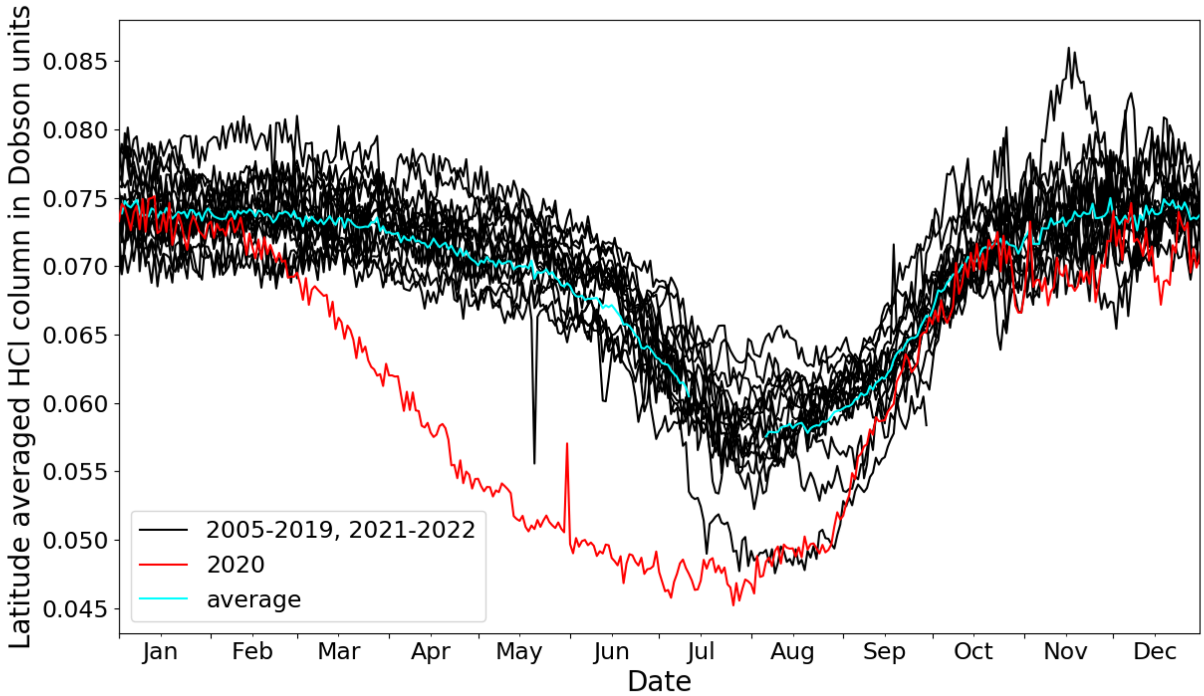

Figure 5.7: Plot of latitude range (from -60° to -45°) averaged substance (HCl) column values between 17.8 and 33.27 km for every day over an annual axis comparing all years. Highlighted in red is the year 2020, and in blue the average of all years. Created using MLS data.

could be anomalies outside of these limits that could explain or contradict the findings, but that is not possible to say in the scope of this study and with the instrument at hand. Hence, different height ranges were used but always counted into the evaluation of the data.

One serious advantage of the selected latitude range is that polar stratospheric clouds do not form in that range and usually only occur from -60° polewards [34]. Therefore, they do not directly influence the stratospheric chemistry in these latitudes, and only possible transportation must be considered. Another aspect that has to be considered when evaluating the data is that the deviation from the average concentration height profiles show absolute values. While this is helpful in evaluating the total loss or gain in the ozone column that is important for the functioning of the ozone layer, the profiles do not directly show the efficiency of the loss or gain in relation to the total abundance at each height.

The decrease in stratospheric ozone in August followed after a slight decrease in CH_3Cl as well as a severe decrease in HCl in May, both of which mainly took place in the lower stratosphere. The beginning of the decrease of both species in February coincides with the breaking of the smoke vortex created by the fire [36]. Therefore, the possible hydrated smoke particle induced dissolving of HCl, instead of being confined to the vortex, could affect a larger part of the stratosphere. On the other hand, the biomass-burning-released CH_3Cl in the vortex would spread out and, therefore, possibly be hit by more UVC light to photochemically separate. Another possible explanation would be a kind of saturation in the confined vortex, which would slow down the reactions and, hence, would explain a drop in the abundance of the two substances after the vortex breaks. Both species contribute to the stratospheric chlorine budget. CH_3Cl, as the largest reservoir of chlorine

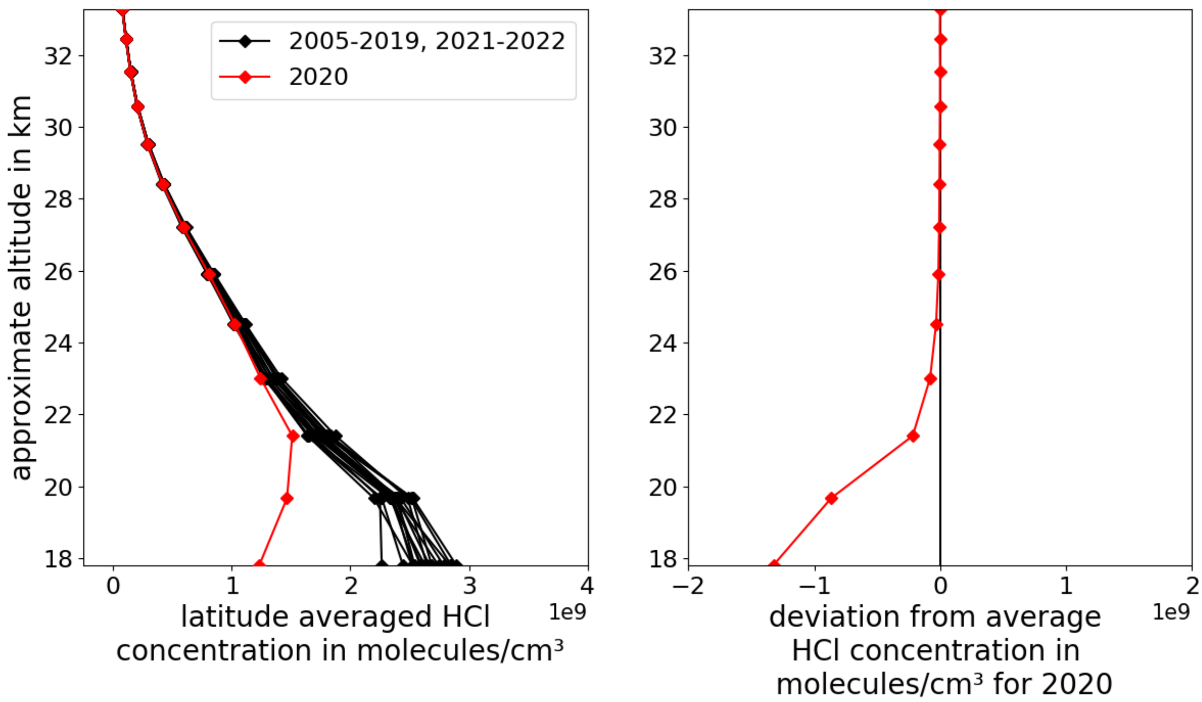

Figure 5.8: Latitude range (from -60° to -45°) averaged and month (May) averaged substance (HCl) height-concentration plot for years 2005 to 2022 highlighting the year 2020 in red for altitudes between 17.8 and 33.27 km (left) and a plot with the deviation of May 2020 from the average of May from all years over the same altitude range (right). Created using MLS data.

in the stratosphere [57], photodissociates into ClO and HCl, on the other hand, reacts with hypochlorous acid on hydrated smoke particles to water and chlorine, which then photodissociates to chlorine radicals that then can form ClO [4]. This suggests that a drop in said two species (CH_3Cl and HCl) could initiate a formation of ClO, which indeed is indicated by elevated ClO concentrations in May 2020, shown in Fig. 5.9. The increased values of ClO could therefore have led to ozone destruction (see cycles 2 and 3 from section 2.2.2) and caused the low ozone values in comparison to the average that started around June 2020. Compared to polar chlorine enhancements in winter (HCl abundances approach zero [7]) and the subsequent ozone destruction, the midlatitude chlorine activation from wildfire smoke is far weaker [7]. Even though a transport of low ozone tropospheric air by the uplift from the fires was observed in a study [39], the time delay between wildfire events and the observed low ozone values reinforces the theory of a chemical mechanism behind it [3]. Additionally, the occurrence of the ozone reduction prior to the springtime (September-November) Antarctic ozone hole formation points towards a chemical cause instead of a transport-related origin [40]. Another possible reason for the change in ozone abundance mentioned in a study is the enhancement of polar stratospheric clouds by wildfire smoke and the transport of that air to the midlatitudes [40].

The constantly slightly elevated H_2O values in 2020, following the average value at the end of 2019, on the other hand, indicate the injection of water. Water can get transported to the very dry stratosphere from the water-rich troposphere by rising air from volcanic eruptions or wildfire events [58]. A study showed a 3% increase in the total water mass in the southern extratropics, which can be confirmed by the close to 3% increase from 2019 to

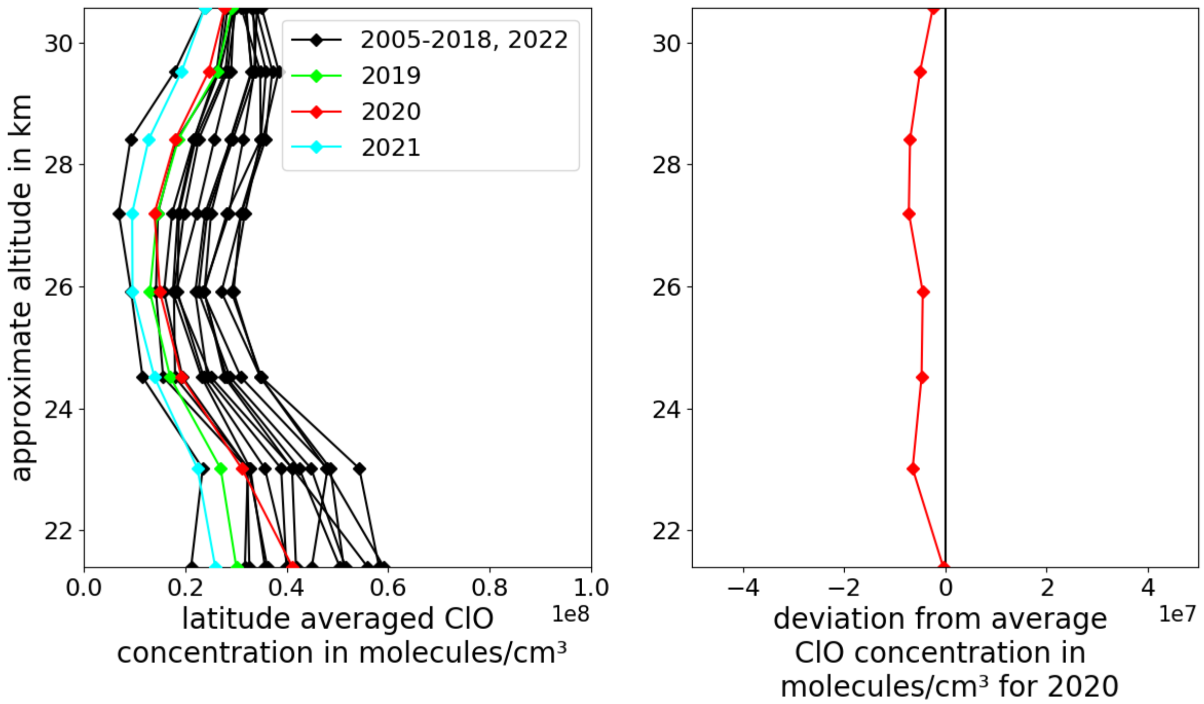

Figure 5.9: Latitude range (from -60° to -45°) averaged and month (May) averaged substance (ClO) height-concentration plot for years 2005 to 2022, highlighting the years 2019 in green, 2020 in red and 2021 in blue for altitudes between 21.51 and 30.56 km (left) and a plot with the deviation of May 2020 from the average of May from all years over the same altitude range (right). Created using MLS data.

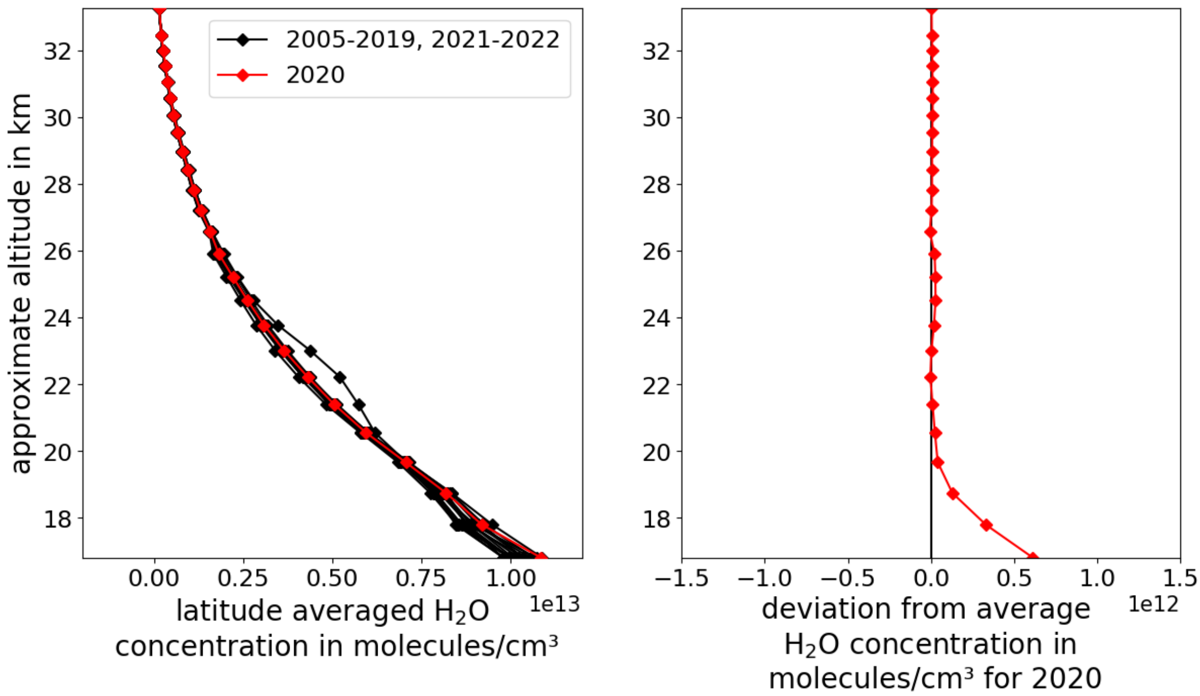

Figure 5.10: Latitude range (from -60° to -45°) averaged and month (May) averaged substance (H_2O) height-concentration plot for years 2005 to 2022 highlighting the year 2020 in red for altitudes between 16.81 and 33.27 km (left) and a plot with the deviation of May 2020 from the average of May from all years over the same altitude range (right). Created using MLS data.

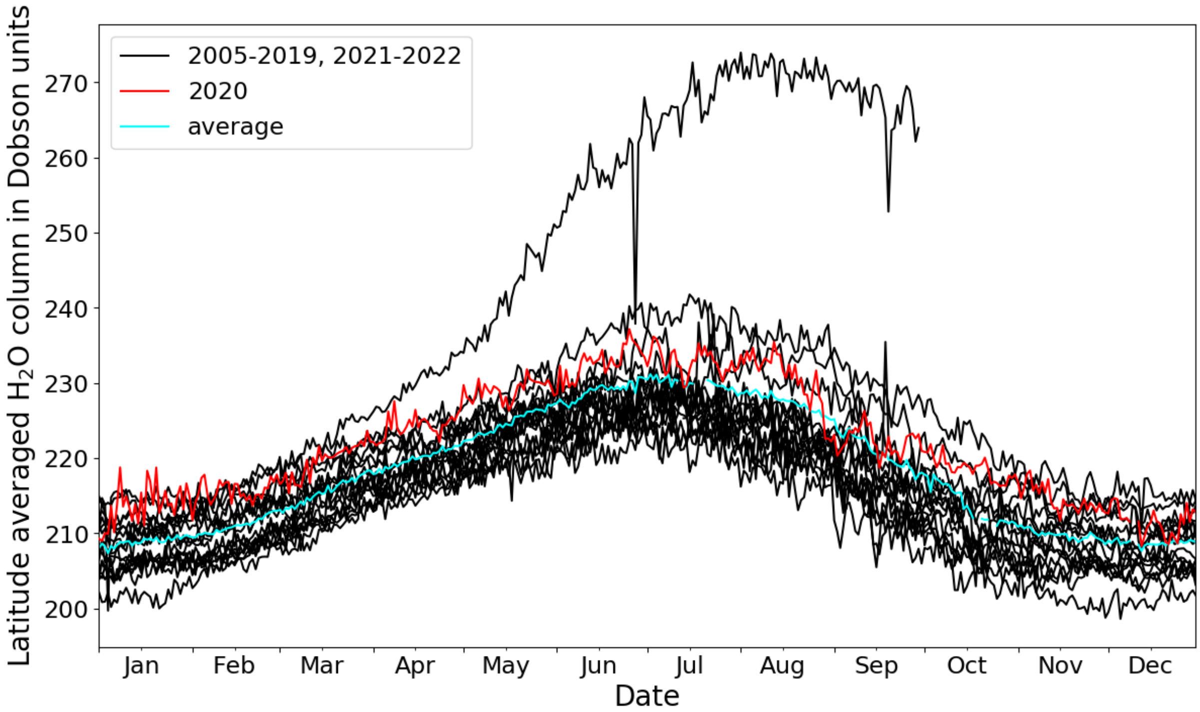

Figure 5.11: Plot of latitude range (from -60° to -45°) averaged substance (H_2O) column values between 16.81 and 33.27 km for every day over an annual axis comparing all years. Highlighted in red is the year 2020, and in blue the average of all years. Created using MLS data.

2020 at the peak of the wildfires and the injection of smoke into the stratosphere (see Fig. 5.11). In addition to the fact that the SO_2 values did not add valuable information to the analysis, the manual for the MLS data suggests poor precision for lower altitudes, which were already excluded from the analysis, and states that too low concentrations cannot be resolved correctly [49]. Therefore, possible slight variations could not be detected.

The similarities in ozone abundance between 2020 and 2015 are highlighted (see Fig. 5.2 and Fig. 5.3) because of the volcanic eruption of Calbuco in Chile in April 2015 [3]. The volcano is located at the -41° latitude in the southern hemisphere [3] and, thus, close to the examined region. Both wildfire and volcanic eruption can destroy ozone [3] and, as suggested in earlier studies, invoke similar chemistry that could be caused by the creation of particles covered with sulfuric acid [7, 39].

5.2 Canada

5.2.1 Results

For the northern hemisphere, the 2017 Canadian wildfire in British Columbia was chosen as the example since it was referenced by multiple papers as the second biggest wildfire in recent history after the Australian Black Summer in terms of pyroconvection and particle mass injection into the stratosphere [9, 36]. It could, therefore, have the most visible influence on midlatitude stratospheric ozone. A comparison of different aerosol particle injection events in particle mass that shows the severity of the Canadian fires is shown in Fig. 5.12.

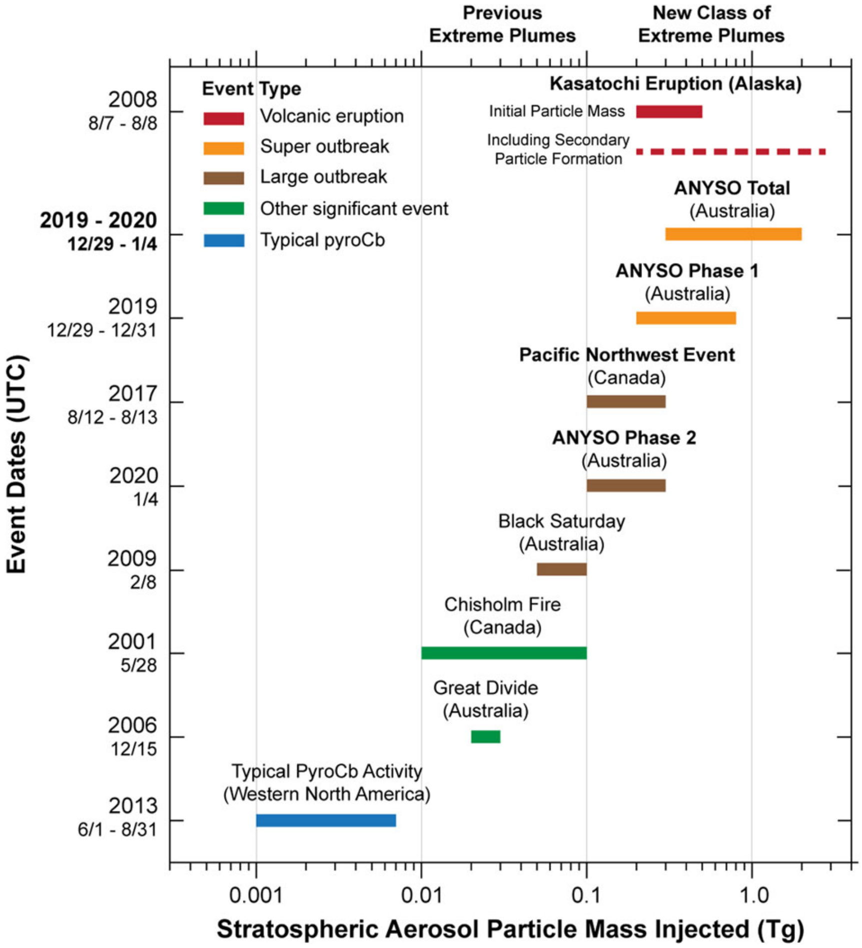

Figure 5.12: Diagram of the size of stratospheric aerosol particle mass injections for different events in Tg, color-coded for event types, where the bar size represents the uncertainty and the x-axis uses a logarithmic scale. Figure is acquired from [9].

The fires started around early July and peaked in the middle of August [11]. On 12 August 2017, pyroCbs resulting from the wildfires injected aerosol particles into the stratosphere [9]. By the end of the fires, around 12,000 km^2 were burned in the province of British Columbia [11]. In addition, the fire created one big vortex that split into three smaller vortices; they rose up to heights between 21 and 23 km and traveled eastwards, mostly around the same latitudes as the location of the fire between 45° and 60° in the northern hemisphere [36].

Again, the first substance to be investigated is ozone. In Fig. 5.1, there are no obvious changes in the ozone column visible after the fires in August 2017. The close-up in Fig. 5.13 depicts a slight increase in the ozone column at the maximum in 2018 compared to 2017 for the 45° to 60° range of the actual fire, but a closer inspection is needed. Fig. 5.14 highlights the ozone column development throughout each year. Beginning in September, right after the fire in August 2017, the ozone column values that were very close in accordance to the average in August start to fluctuate mainly slightly above it and culminate in a large ozone increase with a maximum deviation in March of about 20 Dobson units compared to the average value of about 245 Dobson units, but returning to an average value in May. For the month of March, the ozone concentration by height is displayed in Fig. 5.15. It can be seen that especially between 17.8 and 24.51 km, the concentration is especially high and, moreover, always 2 $\times 10^{11}$ molecules/cm^3 higher

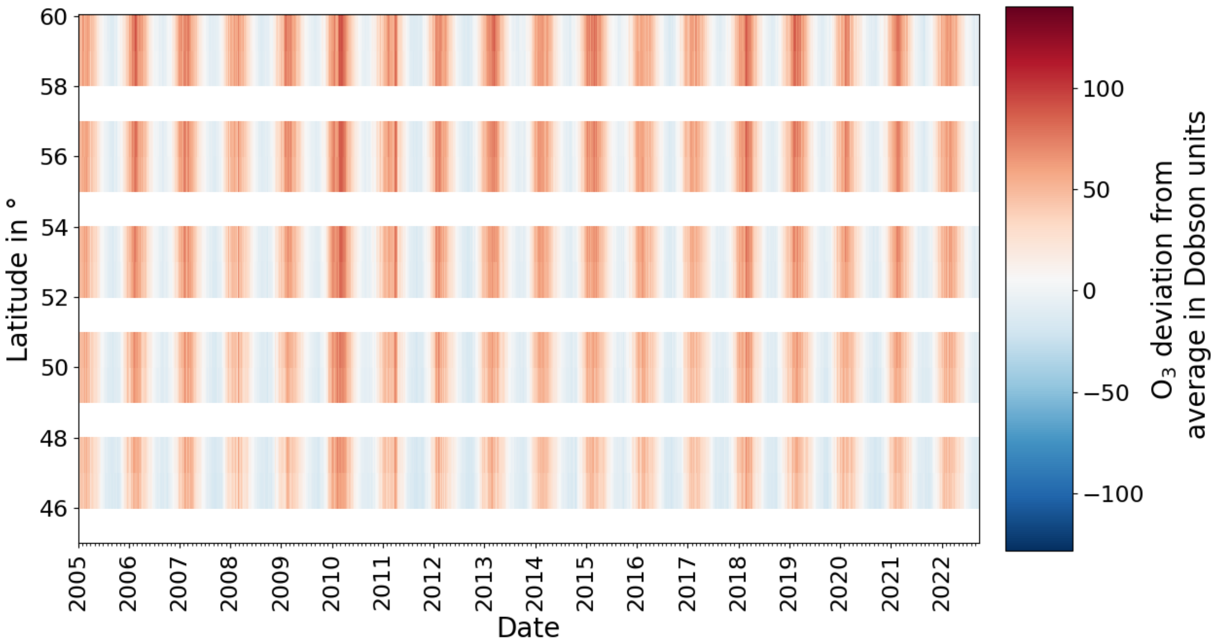

Figure 5.13: Daily plotted deviation from average ozone column color map with data from the MLS for the latitudes between 60° and 45° over time from the beginning of 2005 until end of September 2022 in the height range from 12.49 to 33.27 km.

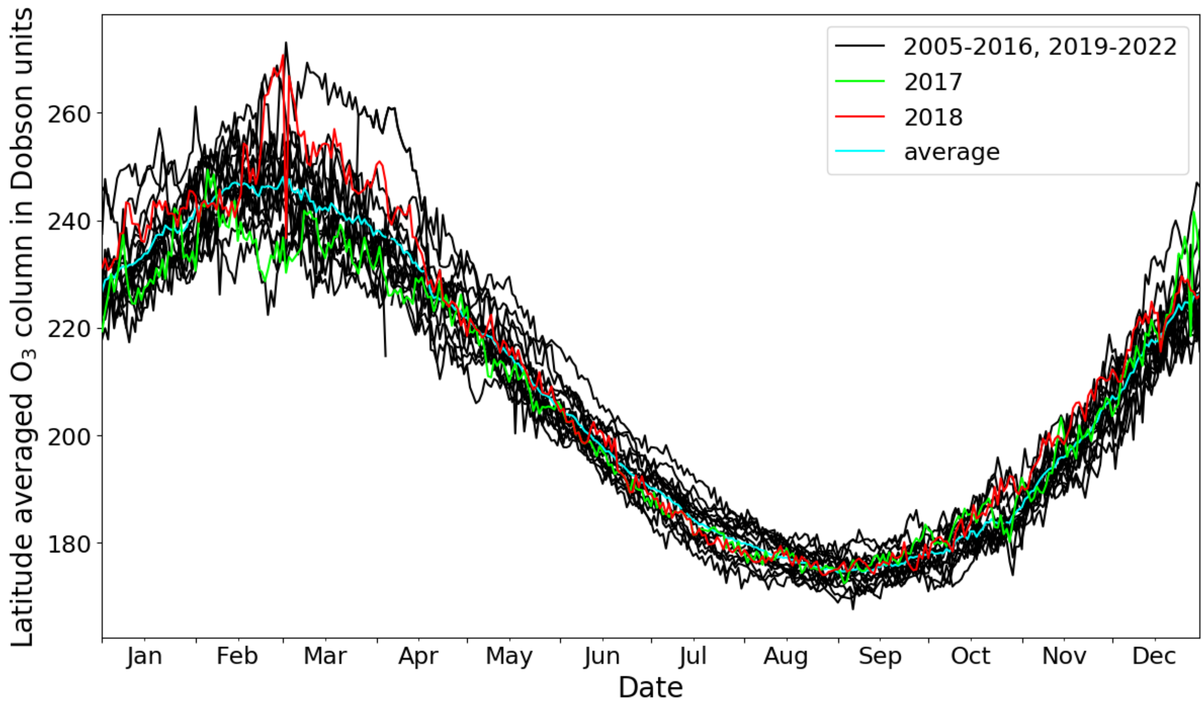

Figure 5.14: Plot of latitude range (from 45° to 60°) averaged substance (O_3) column values between 12.49 and 33.27 km for every day over an annual axis comparing all years. Highlighted in green is 2017, in red 2018, and in blue the average of all years. Created using MLS data.

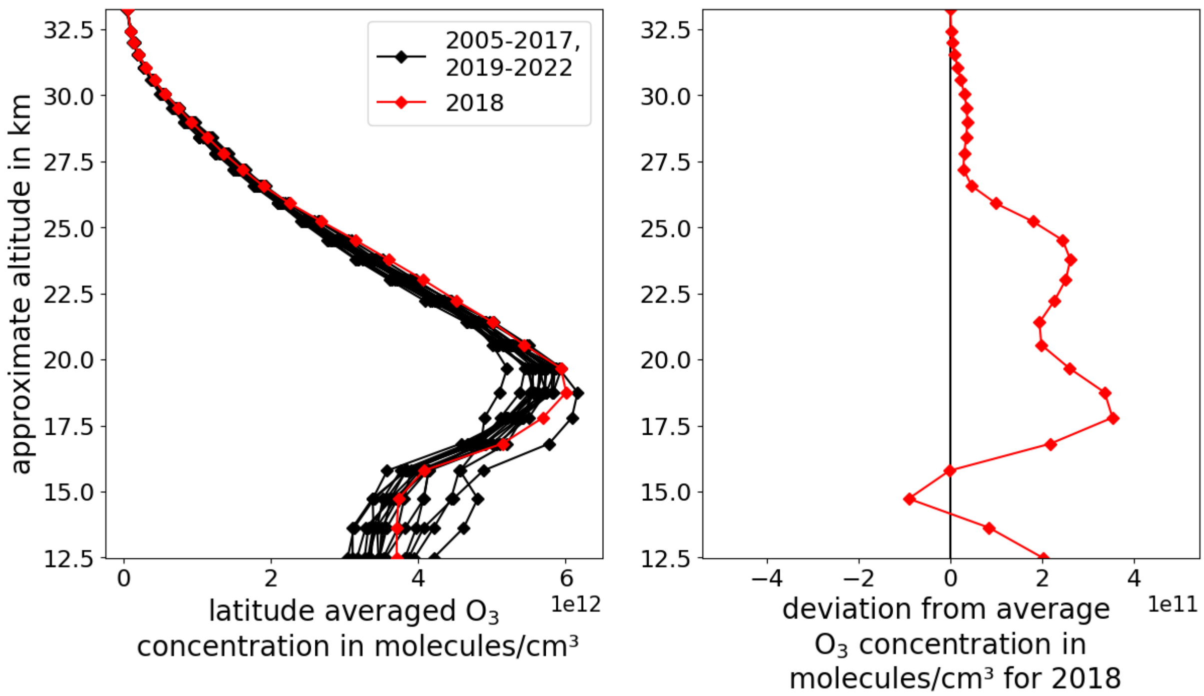

Figure 5.15: Latitude range (from 45° to 60°) averaged and month (March) averaged substance (O_3) height-concentration plot for years 2005 to 2022 highlighting the year 2018 in red for altitudes between 12.49 and 33.27 km (left) and a plot with the deviation of August 2018 from the average of August from all years over the same altitude range (right). Created using MLS data.

than the average of all years. At 14.73 km, there is a dip to -1 $\times 10^{11}$ molecules/cm^3 below average, but in total, the concentration is much higher than the average, as already suggested by Fig. 5.14.

In accordance with the results found for the Australian fire, SO_2 is excluded from further analysis since there was no additional information extractable from the data.

Looking at the daily values for the CH_3Cl column between 15.79 and 28.41 km after the fire in August 2017 in Fig. 5.16, the values stay slightly below the average until the end of January 2018, fall down to about double the deviation from before in February and eventually return to these values before the dip in the following month of March. The dip in February can be seen in Fig. 5.17, where the concentration for the heights from 15.79 to 28.41 km can be seen. The concentration low can be found in the lower altitudes, with the minimum at 15.79 km. Going up, the concentrations slowly increase and reach average at the height of 21.4 km.

In the same fashion, the column values for HCl in Fig. 5.18 show a strong dip in February 2018, with a starting decrease in the middle of December 2017. The directly following months after the fires in August do not differ from right before the fire but constantly stay well above average until the change in December 2017. Similar to the height profile in CH_3Cl, the negative deviation from the average concentration is mostly at lower altitudes. The lowest altitude here is 17.8 km, but the minimum concentration is found at 21.4 km height. Higher up, the concentrations slowly approach the average and reach it at the height of 28.41 km. Checking the same month as for CH_3Cl and HCl, the concentration value for ClO is high (about 3.5 $\times 10^7$ molecules/cm^3) for the lowest altitude of 21.4 km but is slightly below average between 24.51 to 30.57 km. In addition,

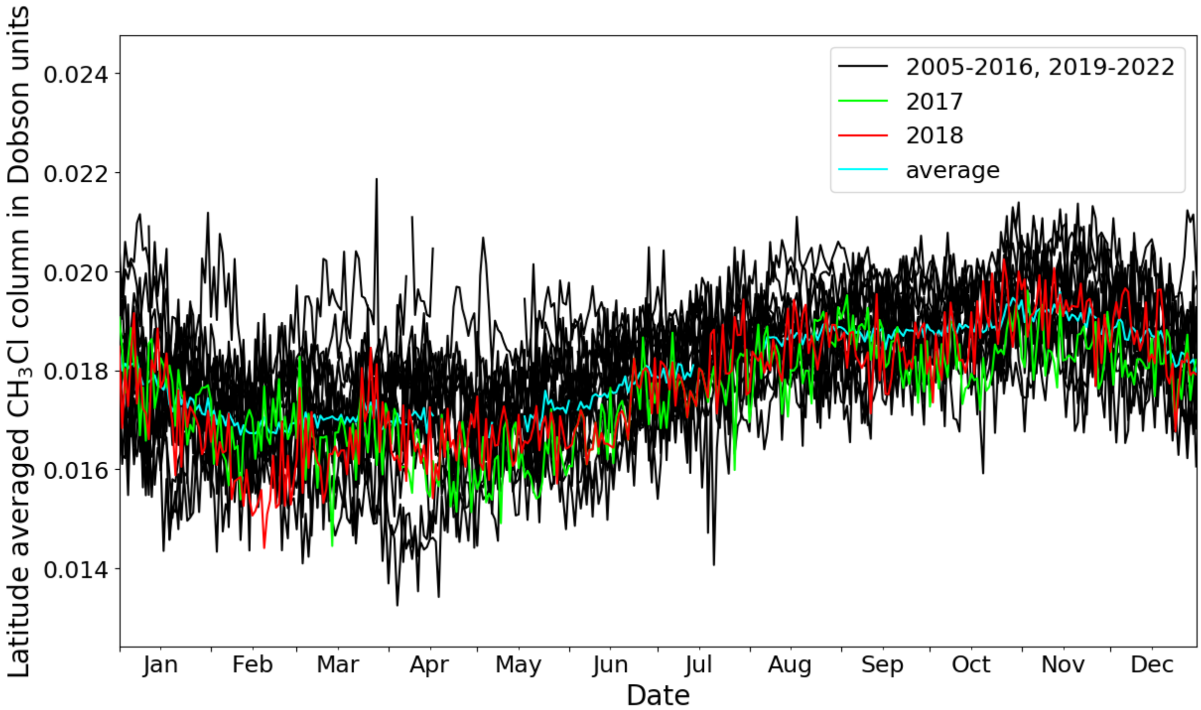

Figure 5.16: Plot of latitude range (from 45° to 60°) averaged substance (CH_3Cl) column values between 15.79 and 28.41 km for every day over an annual axis comparing all years. Highlighted in green is 2017, in red 2018, and in blue the average of all years. Created using MLS data.

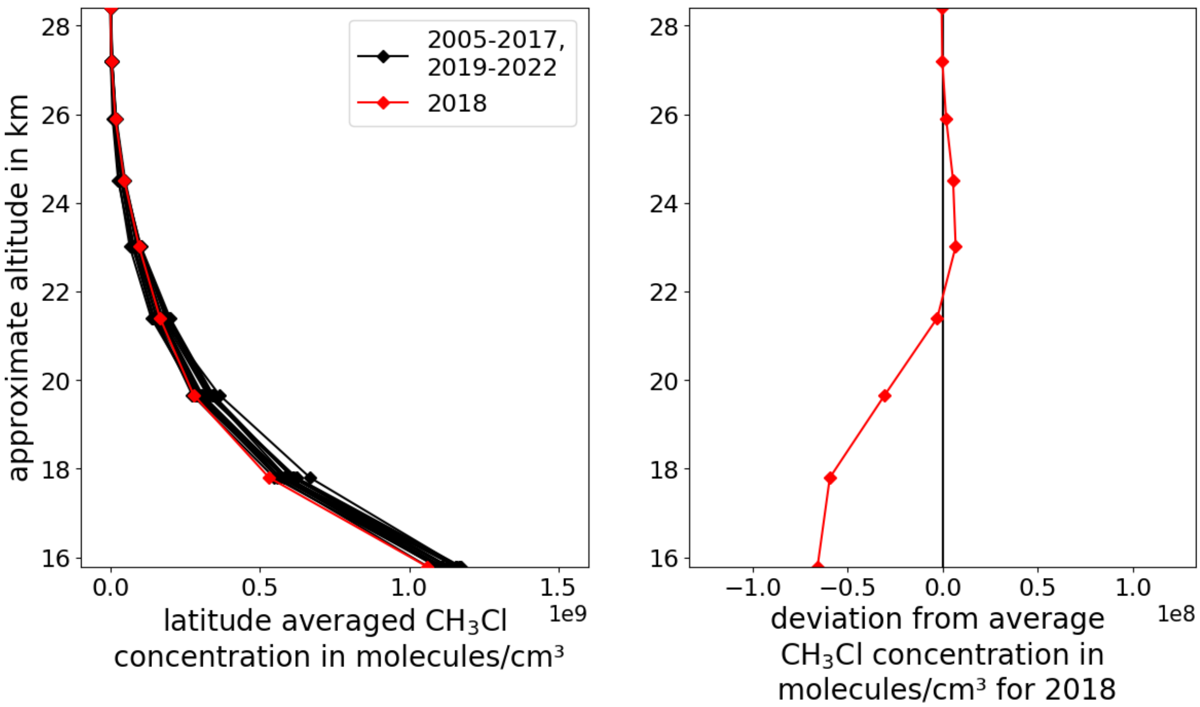

Figure 5.17: Latitude range (from 45° to 60°) averaged and month (February) averaged substance (CH_3Cl) height-concentration plot for years 2005 to 2022 highlighting the year 2018 in red for altitudes between 15.79 and 28.41 km (left) and a plot with the deviation of February 2018 from the average of February from all years over the same altitude range (right). Created using MLS data.

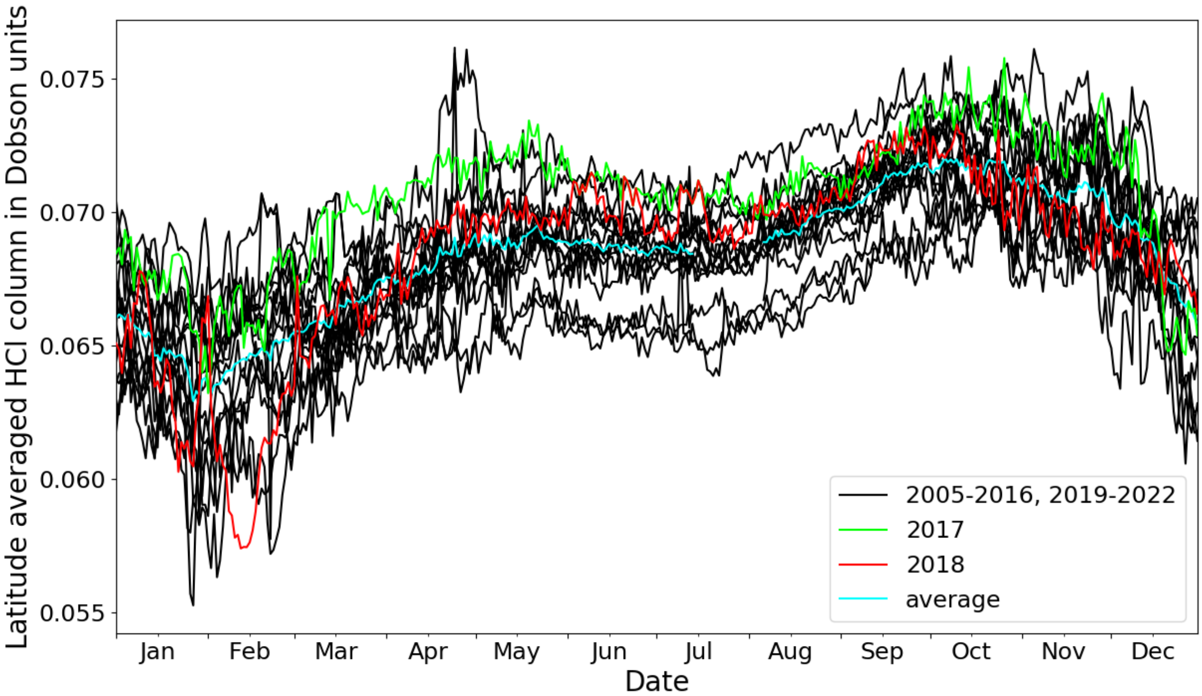

Figure 5.18: Plot of latitude range (from 45° to 60°) averaged substance (HCl) column values between 17.8 and 33.27 km for every day over an annual axis comparing all years. Highlighted in green is 2017, in red 2018, and in blue the average of all years. Created using MLS data.

when compared to the same month in the year before and the year after, it can be seen that the values are substantially higher in all altitudes from 28.41 km and below.

Finally, the H_2O height profile for February in Fig. 5.21 looks very much like the one for the Australian fire, and no substantial difference can be found. The lowest height at 16.81 km has a concentration slightly over average, and the rest of the heights above that are very close to average. In the daily plot in Fig. 5.22, the water column for the selected height segment in the middle of August 2017 is about 5 Dobson units above the average value of 215 Dobson units over all years. Continuing for the months after the wildfire, this increased value over the average remains until the end of November and spikes to about 10 Dobson units in the middle of December but falls back to average values around the end of the month. Rising again to 10 Dobson units positive deviation from the average in January 2018, the column decreases to about 5 Dobson units below average at the end of February at about the same time as the low in HCl and CH_3Cl. After that, March has a short increase to about 10 Dobson units, and finally, in April and the following months, the values settle in at around 5 Dobson units above average. The black line at the top that starts already at about 240 Dobson units is the year 2022, and the one below all other years from May until October is the year 2006; therefore, they are of no particular interest for this investigation.

5.2.2 Discussion

The height ranges for all substances were the same as for the Australian wildfire and were always factored into the analysis. Because of the same maximum poleward latitude, polar stratospheric clouds do not have to be factored in directly. After the Australian fire, the

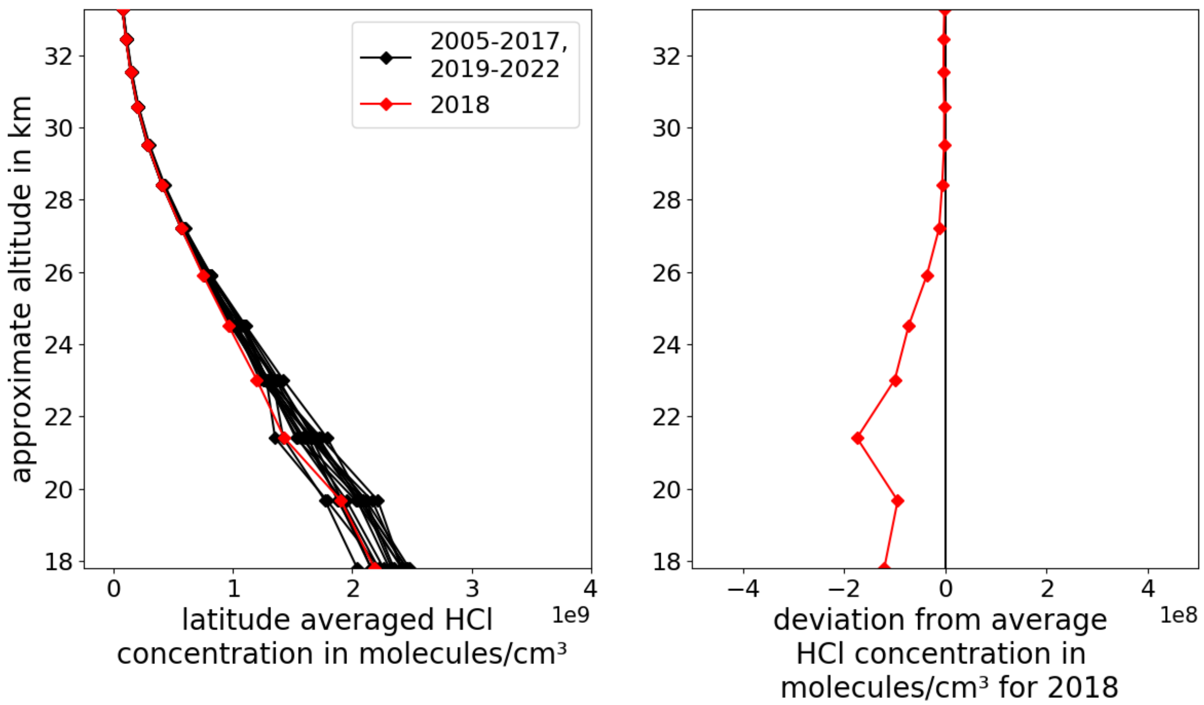

Figure 5.19: Latitude range (from 45° to 60°) averaged and month (February) averaged substance (HCl) height-concentration plot for years 2005 to 2022 highlighting the year 2018 in red for altitudes between 17.8 and 33.27 km (left) and a plot with the deviation of February 2018 from the average of February from all years over the same altitude range (right). Created using MLS data.

pyroCb-created smoke vortex moved through the stratosphere for about two months until finally breaking at the end of February [36], directly followed by the start of decreasing HCl and CH_3Cl values in relation to the average. When expecting that same behavior for the wildfire in British Columbia, the drop in both of those species would be expected to occur starting in the middle of October, right after the vortices broke at the beginning of the month. No behavior like that can be found in the results, and the accompanying ClO increase did also not appear in the data.

In February 2018, a strong negative deviation from the average appears in both HCl and CH_3Cl, which is about five months after the breaking of the vortices. One point worth mentioning is that the minimum in both species is found at different heights. The HCl minimum in the examined range lies at 21.4 km, and the minimum for CH_3Cl lies lower at 15.79 km. While the height profile for CH_3Cl shows the maximum absolute negative deviation at the height with the highest concentration, the HCl shows its maximum absolute negative deviation at a height with not the largest concentration (maximum is found at the lowest altitude). This indicates more efficient destruction at the altitude of 21.4 km and could indicate a larger abundance of hydrated, acidic soot there. CH_3Cl, on the other hand, is dependent on UVC radiation that gets absorbed by ozone, which in turn should make the destruction less efficient at heights with higher ozone column above, which means lower altitudes. That is shown in the data when regarding the steep increase in total concentration and the low change in deviation from the average at the lowest altitudes. These facts, therefore, support a chemical instead of a transport-related cause.

When the HCl values for the Australian wildfire hit a maximum negative deviation

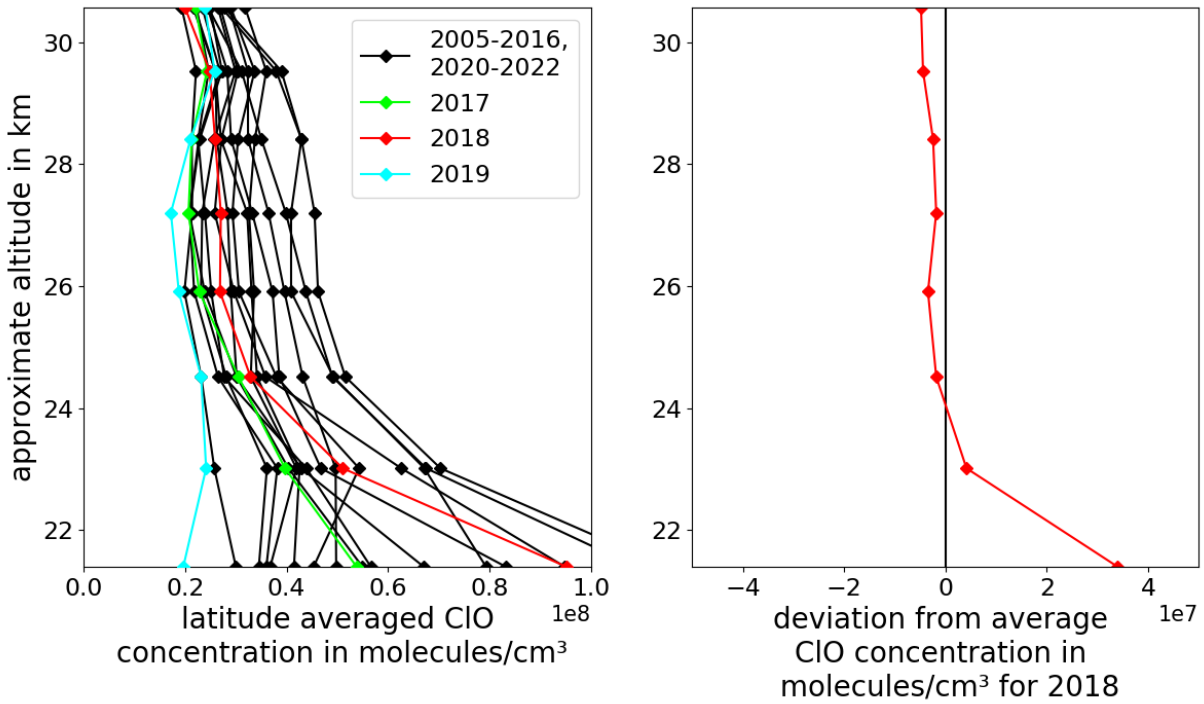

Figure 5.20: Latitude range (from 45° to 60°) averaged and month (February) averaged substance (ClO) height-concentration plot for years 2005 to 2022 highlighting the years 2017 in green, 2018 in red and 2019 in blue for altitudes between 21.51 and 30.56 km (left) and a plot with the deviation of February 2018 from the average of February from all years over the same altitude range (right). Created using MLS data.

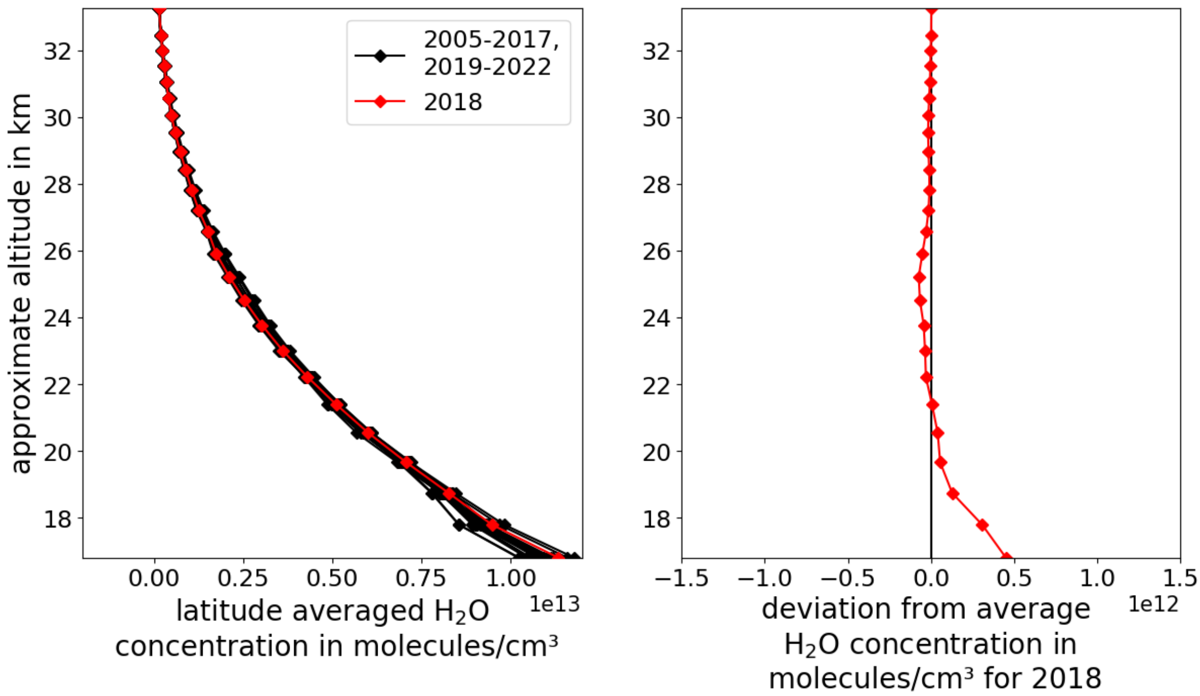

Figure 5.21: Latitude range (from 45° to 60°) averaged and month (February) averaged substance (H_2O) height-concentration plot for years 2005 to 2022 highlighting the year 2018 in red for altitudes between 16.81 and 33.27 km (left) and a plot with the deviation of February 2018 from the average of February from all years over the same altitude range (right). Created using MLS data.

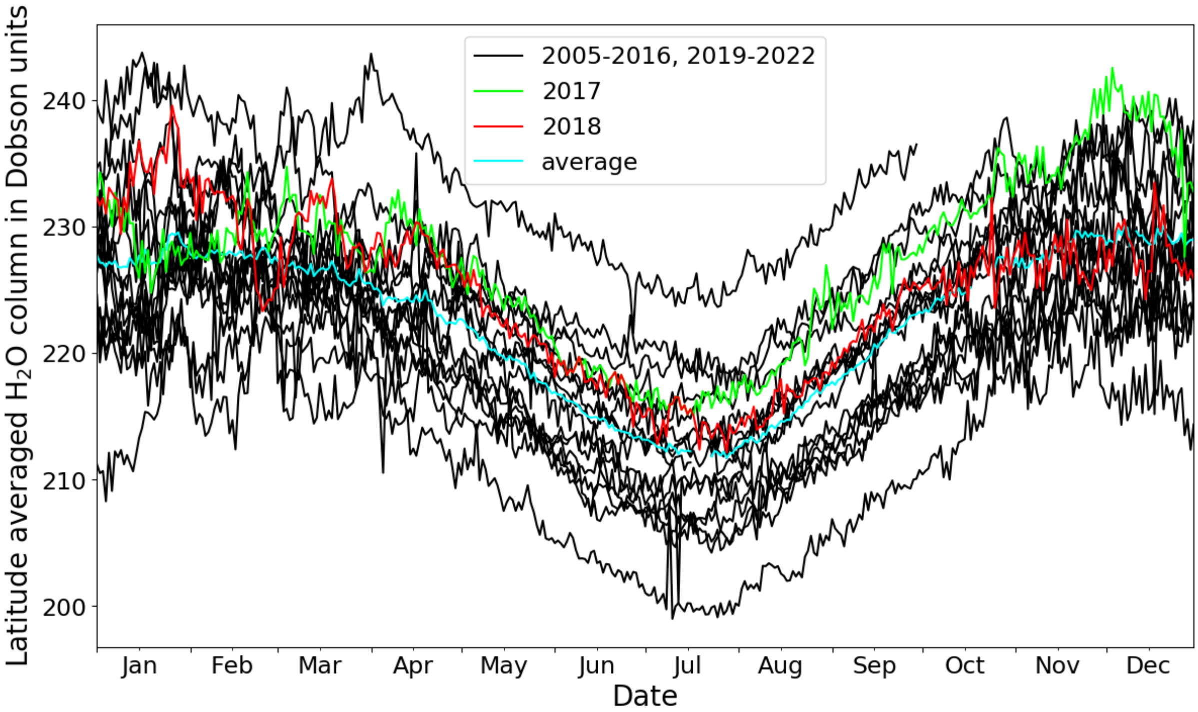

Figure 5.22: Plot of latitude range (from 45° to 60°) averaged substance (H_2O) column values between 16.81 and 33.27 km for every day over an annual axis comparing all years. Highlighted in green is 2017, in red 2018, and in blue the average of all years. Created using MLS data.

in May 2020, it had been two and a half months since the vortex broke. Therefore, it would be too much time between those two events for the Canadian fire to expect any connection between them when assuming the same process as in Australia 2019-20; in addition, there is the lack of a steady decrease towards this low, which is in fact only half as big as the Australian one.

For the ozone values, the results from the Australian wildfires showed a delay in the ozone depletion that was potentially produced by the fires. This was seven months after the smoke particle injection into the stratosphere or about five months after the breaking of the smoke vortex. Taking this kind of delay for the Canadian wildfire event, a drop in the ozone column should be expected either around the mid of March (seven months after the injection) or the beginning of March (five months after the vortices broke). But on the contrary, there appears an unexpected increase in the ozone column (see Fig. 5.14) in the month of March. Even though the values normalize to an average level in the succeeding months, such an increase after the decrease in both HCl and CH_3Cl and the subsequent increase of the ozone depleting ClO because of dissolving on hydrated smoke particles and photodissociation (as discussed earlier in section 5.1.2) in the prior month do not match the findings of the Australian wildfires. It would be expected from the ozone destruction cycles (cycles 2 and 3 from section 2.2.2) that an elevation in the ozone depleting substance ClO leads to a decrease in the ozone column, but the results display the opposite. The reason for that could be that the increase in ozone depleting ClO over the average in February is just not large and prolonged enough, and the increase of ozone is caused by another process. For a wildfire particle induced tropospheric reaction of nitrogen oxides, together with volatile organic compounds and sunlight [33], it would be very late since for the Australian fires three months after pyroCbs injected the smoke into the stratosphere, the ozone enhancing effect was over, possibly because the organics got coated with sulfuric acid [7]. A possibility would be that the sulfur content of the atmosphere was not high and, therefore, it took longer for the coating process, but this cannot be evaluated since the SO_2 values did not give any insights and other sulfuric species were not investigated.

Unlike in the values for the 2019-20 Australian fires, the amount of H_2O in the atmosphere was not constantly elevated after the Canadian fires in August 2017. A slight increase in September could be spotted, but it quickly returned to its original value and, therefore, an increase due to the fire, as in Australia, seems unlikely. The very low overall H_2O concentration of the year 2006 is very odd since the rapid addition of water to the stratosphere, like in 2022, is possible, but a quick removal, like in the years after 2006, is highly unlikely because of the missing precipitation. Therefore, the reason for these values might be a bad calibration in the early stages of the satellite's usage.

A reason for the possible difference in impact on the stratospheric chemistry between Australia 2019-20 and Canada 2017, besides the size and, thus, injected substance mass of the wildfires, could be the difference in fuels, as tropospheric studies showed [40]. While most of the burning trees in Australia were eucalyptus, the Canadian trees were mostly conifers, which could result in a different composition of the wildfire particles [40]. The exact consequences must still be studied in the laboratory, but a possible effect cannot be excluded. Another difference was the injected smoke mass that was more than three times as high for the Australian wildfire compared to the Canadian one [9]. Finally, the maximum reached altitude of the vortex from the Australian fire of 35 km was much higher than that of the Canadian one that only reached up to 23 km [36]. This could decrease the longevity of the particles in the stratosphere significantly due to faster possible reintroduction into the troposphere [59].

6

Conclusion and outlook

Findings of previous studies about the 2019-20 Australian wildfire were verified with the MLS instrument, which proves its suitability for this kind of research. Right after the peak in the fires in December and January, when smoke particles were injected into the stratosphere by pyroCbs, the initial midlatitude stratospheric ozone increased over the average, which was potentially caused by nitrogen oxides reacting with organics and sunlight and stopped by the hypothesized sulfate coating of just these organics, was followed by comparatively low ozone in August and the surrounding months. Below-average CH_3Cl and HCl starting with the breaking of the smoke vortex and subsequent enhanced ClO, most probably caused by reactions on the surface of hydrated smoke particles, could have caused the minor ozone decrease in the summer months. Considering the initial goal of gaining knowledge about whether wildfires in the northern hemisphere influenced midlatitude stratospheric ozone or not resulted in partial success since it has to be considered that even though there was no evidence found in accordance with the findings from the Australian fires that the wildfire event in Canada 2017 did influence the ozone in any substantial way, it does not mean that any other recent wildfire in the northern hemisphere could not have had a measurable impact on midlatitude ozone. The only substantial anomaly detected in the year after the wildfire was an increase in ozone seven months after the smoke injection into the stratosphere, for which no connection to the wildfire could be found. Therefore, the aerosol particle mass injected into the stratosphere might be less important than the duration of the injection, its reached altitudes, or its chemical composition because of the type of fuel burned. Concerning the mechanisms behind the chemical changes in the stratosphere, previously observed connections between wildfire smoke injections could be witnessed with data from the MLS. However, the negative feedback from the 2017 Canadian wildfire points towards more complex conditions for the occurrence of stratospheric ozone depletion induced by wildfire aerosol particles. There might be a certain threshold that must be exceeded so that the ozone depletion is effective against the initial increase, or the decrease in ozone was too small to observe with the MLS.

The bottom line of this analysis is very positive for nature's health and the recovery of the ozone layer. The biggest and most intense wildfire in recent history only had minor negative effects on the midlatitude stratospheric ozone, and the second biggest wildfire in terms of stratospheric aerosol particle mass injection seems to have no measurable effect at all.

Now referring back to the introduction and giving a short outlook on possible future research, the aforementioned nuclear detonations would still culminate in severe consequences for stratospheric ozone since a model study predicted the injected soot mass to be between one and five teragrams, which would be about one to five times the mass from the 2019-20 Australian wildfire and the soot is expected to reach up to 80 km altitude which is more than double of what the vortex for that fire reached [14]. Therefore, ozone depletion would most probably still be severe. Unfortunately, no statement can be made about the possible impacts of geoengineering with SO_2 because no usable information about that species was gathered in this study. Thus, maybe another instrument should be used to conduct research for that species. Finally, future investigations should be carried out on the 2021 wildfire in Russia because the total burnt area was higher than for the fires in British Columbia, and they lasted about two months. Those parameters might have a bigger influence on the ozone depletion ability of wildfires. Nevertheless, more research has to be conducted on the effects of wildfires in general since, due to climate change, it is most certain that more of them will occur in the future [4].

Bibliography

[1] S. E. Strahan and A. R. Douglass, “Decline in Antarctic Ozone Depletion and Lower Stratospheric Chlorine Determined From Aura Microwave Limb Sounder Observations,” *Geophysical Research Letters*, vol. 45, no. 1, pp. 382–390, 2018.

[2] K. Helfenstein. “Healing the Ozone Layer Through Diplomacy.” Accessed: 2023-26-02. (2021), [Online]. Available: https://policycommons.net/artifacts/1815079/healing-the-ozone-layer-through-diplomacy/2551411/.

[3] L. Rieger *et al.*, “Stratospheric temperature and ozone anomalies associated with the 2020 Australian New Year fires,” *Geophysical Research Letters*, vol. 48, no. 24, 2021.

[4] P. Bernath *et al.*, “Wildfire smoke destroys stratospheric ozone,” *Science*, vol. 375, no. 6586, pp. 1292–1295, 2022.

[5] P. Fabian and M. Dameris, “The Ozone Layer,” in *Ozone in the Atmosphere: Basic Principles, Natural and Human Impacts*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 13–47.

[6] M. P. Chipperfield *et al.*, “Detecting recovery of the stratospheric ozone layer,” *Nature*, vol. 549, no. 7671, pp. 211–218, 2017.

[7] M. L. Santee *et al.*, “Prolonged and Pervasive Perturbations in the Composition of the Southern Hemisphere Midlatitude Lower Stratosphere From the Australian New Year’s Fires,” *Geophysical Research Letters*, vol. 49, no. 4, 2022.

[8] M. Binskin *et al.*, “Royal Commission into National Natural Disaster Arrangements Report,” *Canberra: Commonwealth of Australia*, 2020.

[9] D. A. Peterson *et al.*, “Australia’s Black Summer pyrocumulonimbus super outbreak reveals potential for increasingly extreme stratospheric smoke events,” *NPJ climate and atmospheric science*, vol. 4, no. 1, p. 38, 2021.

[10] O. Voronova *et al.*, “Strong Wildfires in the Russian Federation in 2021 Detected Using Satellite Data,” *Izvestiya, Atmospheric and Oceanic Physics*, vol. 58, no. 9, pp. 1065–1076, 2022.

[11] J. Mao *et al.*, “Measuring atmospheric CO2 enhancements from the 2017 British Columbia wildfires using a lidar,” *Geophysical Research Letters*, vol. 48, no. 16, 2021.

[12] M. Kummu and O. Varis, “The world by latitudes: A global analysis of human population, development level and environment across the north–south axis over the past half century,” *Applied Geography*, vol. 31, no. 2, pp. 495–507, 2011.

[13] R.-S. Gao *et al.*, “Toward practical stratospheric aerosol albedo modification: Solar-powered lofting,” *Science Advances*, vol. 7, no. 20, 2021.

[14] M. J. Mills *et al.*, "Massive global ozone loss predicted following regional nuclear conflict," *Proceedings of the National Academy of Sciences*, vol. 105, no. 14, pp. 5307–5312, 2008.

[15] M. J. Molina, "Polar Ozone Depletion (Nobel Lecture)," *Angewandte Chemie International Edition in English*, vol. 35, no. 16, pp. 1778–1785, 1996.

[16] R. G. Barry and R. J. Chorley, *Atmosphere, weather and climate*. Routledge, 2009.

[17] M. Baldwin *et al.*, "100 Years of Progress in Understanding the Stratosphere and Mesosphere," *Meteorological Monographs*, vol. 59, Oct. 2019.

[18] P. Fabian and M. Dameris, "Introduction," in *Ozone in the Atmosphere: Basic Principles, Natural and Human Impacts*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 1–3.

[19] V. Bocci, "Physical-Chemical Properties of Ozone – Natural Production of Ozone: The Toxicology of Ozone," in *OZONE: A new medical drug*. Dordrecht: Springer Netherlands, 2011, pp. 1–4.

[20] C. V. Rekhate and J. Srivastava, "Recent advances in ozone-based advanced oxidation processes for treatment of wastewater- A review," *Chemical Engineering Journal Advances*, vol. 3, p. 100 031, 2020.

[21] J. Zhang *et al.*, "Ozone pollution: a major health hazard worldwide," *Frontiers in immunology*, vol. 10, p. 2518, 2019.

[22] S. Avnery *et al.*, "Global crop yield reductions due to surface ozone exposure: 1. Year 2000 crop production losses and economic damage," *Atmospheric Environment*, vol. 45, no. 13, pp. 2284–2296, 2011.

[23] Z. Feng *et al.*, "Ozone pollution threatens the production of major staple crops in East Asia," *Nature Food*, vol. 3, no. 1, pp. 47–56, 2022.

[24] WHO. "WHO global air quality guidelines: particulate matter (PM2.5 and PM10), ozone, nitrogen dioxide, sulfur dioxide and carbon monoxide." Accessed: 2023-26-02. (2021), [Online]. Available: https://apps.who.int/iris/handle/10665/345329.

[25] K. Mohanakumar, *Stratosphere troposphere interactions: an introduction*. Springer Science & Business Media, 2008.

[26] D. H. Sliney *et al.*, "Infrared, Visible, and Ultraviolet Radiation," in *Patty's Toxicology*. John Wiley Sons, Ltd, 2012, ch. 102, pp. 169–208.

[27] M. Widel *et al.*, "Induction of bystander effects by UVA, UVB, and UVC radiation in human fibroblasts and the implication of reactive oxygen species," *Free Radical Biology and Medicine*, vol. 68, pp. 278–287, 2014.

[28] A. Q. Khan *et al.*, "Roles of UVA radiation and DNA damage responses in melanoma pathogenesis," *Environmental and Molecular Mutagenesis*, vol. 59, no. 5, pp. 438–460, 2018.

[29] E. Kovács and Á. Keresztes, "Effect of gamma and UV-B/C radiation on plant cells," *Micron*, vol. 33, no. 2, pp. 199–210, 2002.

[30] G. Horneck, "Ozone Layer," in *Encyclopedia of Astrobiology*, M. Gargaud *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1209–1210.

[31] B. Agarwal *et al.*, "Ozone and environment," *Radiation protection and environment*, vol. 34, no. 3, p. 164, 2011.

[32] U. Langematz, "Stratospheric ozone: down and up through the anthropocene," *ChemTexts*, vol. 5, pp. 1–12, 2019.

[33] B. J. Finlayson-Pitts and J. N. Pitts Jr, *Chemistry of the upper and lower atmosphere: theory, experiments, and applications*. Elsevier, 1999.

[34] A. Klekociuk, "Beautiful, mysterious polar stratospheric clouds," *Australian Antarctic Magazine*, vol. Winter 2003, no. 5, Mar. 2002.

[35] M. Wehner *et al.*, "Weather and climate extreme events in a changing climate," in *AGU Fall Meeting Abstracts*, vol. 2021, 2021, pp. 1513–1765.

[36] H. Lestrelin *et al.*, "Smoke-charged vortices in the stratosphere generated by wildfires and their behaviour in both hemispheres: comparing Australia 2020 to Canada 2017," *Atmospheric Chemistry and Physics*, vol. 21, no. 9, pp. 7113–7134, 2021.

[37] K. J. Tory and W. Thurston, *Pyrocumulonimbus: A Literature Review*. Bushfire and Natural Hazards CRC, 2015.

[38] S. Khaykin *et al.*, *Australian wildfires cause major perturbation of the stratosphere and generate a self-maintained smoke-charged vortex rising up to 35 km*, Jun. 2020.

[39] P. Yu *et al.*, "Persistent stratospheric warming due to 2019–2020 Australian wildfire smoke," *Geophysical Research Letters*, vol. 48, no. 7, 2021.

[40] S. Solomon *et al.*, "On the stratospheric chemistry of midlatitude wildfire smoke," *Proceedings of the National Academy of Sciences*, vol. 119, no. 10, 2022.

[41] J.-N. Weber *et al.*, "Impact of wildfires on SO2 detoxification mechanisms in leaves of oak and beech trees," *Environmental Pollution*, vol. 272, p. 116 389, 2021.

[42] T. E. Reinhardt and D. E. Ward, "Factors Affecting Methyl Chloride Emissions from Forest Biomass Combustion," *Environmental Science & Technology*, vol. 29, no. 3, pp. 825–832, 1995.

[43] E. Sher, "Chapter 2 - Environmental Aspects of Air Pollution," in *Handbook of Air Pollution From Internal Combustion Engines*, E. Sher, Ed., San Diego: Academic Press, 1998, pp. 27–41.

[44] S. Solomon *et al.*, *Chemical impacts of wildfire smoke on stratospheric chlorine and ozone depletion*, 2022.

[45] A. Lambert *et al.*, "Validation of the Aura Microwave Limb Sounder middle atmosphere water vapor and nitrous oxide measurements," *Journal of Geophysical Research: Atmospheres*, vol. 112, no. D24, 2007.

[46] J. W. Waters *et al.*, "The earth observing system microwave limb sounder (EOS MLS) on the Aura satellite," *IEEE transactions on geoscience and remote sensing*, vol. 44, no. 5, pp. 1075–1092, 2006.

[47] E. G. Njoku, *Encyclopedia of remote sensing*. Springer New York, 2014, pp. 344–348.

[48] R. A. Vaughan and A. P. Cracknell, *Remote sensing and global climate change*. Springer Science & Business Media, 2013, vol. 24.

[49] N. J. Livesey *et al.*, *Earth Observing System (EOS) Aura Microwave Limb Sounder (MLS) Version 5.0x Level 2 and 3 data quality and description document*. Jan. 2022.

[50] T. Rieckh *et al.*, "Characteristics of tropopause parameters as observed with GPS radio occultation," *Atmospheric Measurement Techniques*, vol. 7, no. 11, pp. 3947–3958, 2014.

[51] J. Wen *et al.*, "Quantifying Global Variation of Sonic Boom Carpet for Commercial Supersonic Operations Due to Flight Condition and Meteorological Effects," in *AIAA AVIATION 2022 Forum*, 2022, p. 4109.

[52] W. H. Brune. "4.1 atmospheric composition." Accessed: 2023-26-02. (2020), [Online]. Available: https://www.e-education.psu.edu/meteo300/node/534.

[53] M. Newchurch *et al.*, "On the accuracy of Total Ozone Mapping Spectrometer retrievals over tropical cloudy regions," *Journal of Geophysical Research: Atmospheres*, vol. 106, no. D23, pp. 32 315–32 326, 2001.

[54] M. Haque *et al.*, "Wildfire in Australia during 2019-2020, Its Impact on Health, Biodiversity and Environment with Some Proposals for Risk Management: A Review," *Journal of Environmental Protection*, vol. 12, pp. 391–414, Jan. 2021.

[55] A. Ansmann *et al.*, "Ozone depletion in the Arctic and Antarctic stratosphere induced by wildfire smoke," *Atmospheric Chemistry and Physics*, vol. 22, no. 17, pp. 11 701–11 726, 2022.

[56] M. R. Schoeberl *et al.*, "Analysis and Impact of the Hunga Tonga-Hunga Ha'apai Stratospheric Water Vapor Plume," *Geophysical Research Letters*, vol. 49, no. 20, 2022.

[57] M. Aydin and V. V. Petrenko, "History of Carbon Monoxide and Other Ultra-Trace Level Ice Core Gas Measurements," *Reference Module in Earth Systems and Environmental Sciences*, 2018.

[58] S. Khaykin *et al.*, "The 2019/20 Australian wildfires generated a persistent smoke-charged vortex rising up to 35 km altitude," *Communications Earth & Environment*, vol. 1, no. 1, p. 22, 2020.

[59] M. Abalos *et al.*, "Future trends in stratosphere-to-troposphere transport in CCMI models," *Atmospheric chemistry and physics*, vol. 20, no. 11, pp. 6883–6901, 2020.

Appendix

```python
import numpy as np
import matplotlib.pyplot as plt
import mat73
from matplotlib.pyplot import figure, draw, pause
from mpl_toolkits.axes_grid1 import make_axes_locatable
import functools
import datetime
from scipy.interpolate import interp1d

"""Creating an animation for volume mixing ratio colormap plotted over height and latitude with bar plot of latitudes
column values animated per day"""

def mouseclick(event):                              # creating mouse click function
    mode = event.canvas.toolbar.mode                # setting mouse click
    if event.button == 1 and event.inaxes == ax and mode == '':          # checking for mouse click in axes
        pause(10)                                                        # pausing animation for set time


fig = plt.figure(figsize=(12, 8), constrained_layout=False)     # create plot window
ax = fig.add_subplot(2, 1, 1)                                   # and plot area 1
ay = fig.add_subplot(2, 1, 2)                                   # and 2
plt.gcf().subplots_adjust(right=0.8)                            # create room on the right

year_data = 2020                        # selecting a year to play

height_lower = 3                  # selecting lowest included pressure
height_higher = 37                # selecting highest included pressure

substance = 'O3'   # here you can select the substance out of (print(mat.keys())) 'CH3Cl', 'ClO', 'H2O', 'HCl',
                                  # 'O3', 'SO2', 'Temperature'

if substance == 'ClO': # pressure values for ClO that are in data but are not usable are cut (see MLS instruction 3.6.5)
    if height_lower <= 9:   # checking if lower pressure value is below 9
        height_lower = 9    # setting lower pressure value to 9
    if height_higher <= 9:  # checking if higher pressure value is below 9
        height_higher = 9   # setting higher pressure value to 9



```

```
data_file = 'Data/MLS_gridded_' + str(year_data) + '-01-01_' + str(year_data) + '-12-31.mat'   # defining data file name

mat = mat73.loadmat(data_file, use_attrdict=True)['MLS_gridded_tmp']    # importing mat file and changing from one
                                                                         # dictionary key to keys in it

date = 180                                                               # selecting starting date
pause_time = 0.0000000001                                                # selecting pause time between frames

y = mat.get(substance, 'Not Found').get('Pressure', 'Not Found')  # extracting pressure vec of one substance out of dict
z = mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat', 'Not Found')[:,:,date-1]
                                                                         # extracting one substance 2d array out of dict
x = np.arange(-90, 91, 1)                         # creating latitude vector
temp = mat.get('Temperature', 'Not Found').get('Temperature' + '_flagCleaned_strat', 'Not Found')[:,:,:]
                                                                         # extracting the temperature array

if substance == 'CH3Cl':    # setting the substance units
    substance_unit = 'ppmv'
if substance == 'ClO':
    substance_unit = 'ppmv'
if substance == 'H2O':
    substance_unit = 'ppmv'
if substance == 'HCl':
    substance_unit = 'ppmv'
if substance == 'O3':
    substance_unit = 'ppmv'
if substance == 'SO2':
    substance_unit = 'ppmv'
if substance == 'Temperature':
    substance_unit = 'ppmv'

if substance_unit == 'ppmv':  # setting the substance unit factors for conversion to ppmv
    substance_unit_factor = 1
if substance_unit == 'ppbv':
    substance_unit_factor = 1000
if substance_unit == 'pptv':
    substance_unit_factor = 1000000

if year_data == 2004 or year_data == 2008 or year_data == 2012 or year_data == 2016 or year_data == 2020:
    # checking if year is leap year
    endtime = len(mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat', 'Not Found')[0,0,:])
    # extracting end time
if year_data == 2005 or year_data == 2006 or year_data == 2007 or year_data == 2009 or year_data == 2010 or \
```

```
        year_data == 2011 or year_data == 2013 or year_data == 2014 or year_data == 2015 or year_data == 2017 or \
        year_data == 2018 or year_data == 2019 or year_data == 2021 or year_data == 2022:
                                                                            # checking if year is no leap year
    endtime = len(mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat', 'Not Found')[0, 0,:]) - 1
                                                                            # extracting end time

y = np.ma.array(y, mask=np.isnan(y))            # masking NaN values from pressure array
z = np.ma.array(z, mask=np.isnan(z))            # masking NaN values from substance 2d array
temp = np.ma.array(temp, mask=np.isnan(temp))   # masking NaN values from temperature array

pos = np.arange(-90, 90+1, 1)                           # creating latitude array (0 = equator)

vmin = np.min(z)                                # setting the minimum for the colorbar
vmax = np.max(z)                                # setting the maximum for the colorbar

height = 145366.45*(1-(y/1013.25)**0.190284) / 3.28084 /1000    # calculating the heights from the pressure array in km
height2 = 145366.45*(1-(y[height_lower:height_higher]/1013.25)**0.190284) / 3.28084 /1000    # calculating same only for
                                                                # selected heights

height_slice = ((np.append(height[1:], 0) - height) * 0.5) + ((height - np.append(0, height[:-1])) * 0.5)
                                                                # calculating the height slice to each height
new = np.arange(0, 55, 55/z.shape[0])           # making new temperature grid
old = np.arange(0, 55, 1)                       # making old temperature grid
temp_lat = temp[:, :, 0][:, :]                  # just taking zero value for getting shape in next line
temp_lat_new = np.zeros((z.shape[0], (temp_lat[:, :]).shape[1]))        # creating 2d zero matrix
for l in range(z.shape[1]):  # loop over all before selected latitudes
    interp_func = interp1d(old, temp_lat[:, l])  # interpolate temperature array
    temp_lat_new[:, l] = interp_func(new)  # putting on new "grid"
    temp_lat_new = np.ma.array(temp_lat_new, mask=np.isnan(temp_lat_new))  # masking the NaN values

h = ax.pcolormesh(x, height2, z[height_lower:height_higher,:], vmin=vmin, vmax=vmax, cmap='viridis')
                                            # creating the first heatmap of z array over latitude vector and pressure vector

divider = make_axes_locatable(ax)                               # move colorbar
```

```
cax = plt.gcf().add_axes([0.85, 0.15, 0.05, 0.7])           # new axes for color bar
fig.colorbar(h, cax=cax, orientation='vertical', label=substance + ' concentration in ' + substance_unit)
                 # create colorbar

z = z / temp_lat_new                                         # division of substance values by temperature
print(z.shape[1])
for t in range(z.shape[1]):                                  # loop over latitudes
    z[:,t] = z[:,t] * height_slice * 100000 * 10000 * y * 100 / (1.38064852 * 10**(-23)) * (1/10**6)
                                                             # converting whole z to Dobson units

height_slice_matrix = np.zeros(z.shape)              # creating zero array for height slice
for n in range(z.shape[1]):                          # loop over latitudes
    height_slice_matrix[:, n] = height_slice         # creating height slice 2d array


j = ay.bar(pos, (z[height_lower:height_higher,:].sum(axis=0) / (2.6867 * 10 ** 20))/substance_unit_factor)
                 # first latitude bar plot

ax.set_ylim(height[height_lower-1], height[height_higher+1])  # cutting off NaN points of array
ax.set_xlabel('latitude in   ', fontsize=15)                  # labeling x axis
ax.set_ylabel('approximate altitude in km', fontsize=15)     # labeling y axis
ax.set_title(substance + ' on day number:' + str(date))      # labeling the plot for the starting date


ay.set_xlabel('latitude in   ', fontsize=15)                        # labeling x axis (2nd plot)
ay.set_ylabel(substance +' column in Dobson units', fontsize=15)     # labeling y axis (2nd plot)
ay.set_ylim(-10, 1.2 * np.max((z[height_lower:height_higher,:].sum(axis=0) / (2.6867 * 10 ** 20))))
                                                     # setting the limits for the bar plot (for the scale not to change)
draw(), pause(pause_time)                            # drawing the new plot and waiting

click_funktion = functools.partial(mouseclick)                # implement click function
fig.canvas.mpl_connect('button_press_event', click_funktion)    # connecting click function to click event

for i in range(endtime - date):                             # plot animation iteration loop (over endtime minus start time)
    z = mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat
```

```
', 'Not Found')[:, :, i + date]
                                                                # acquiring subst data for time i
    z = np.ma.array(z, mask=np.isnan(z))                        # masking NaN values from substance 2d array
    h.set_array(z[height_lower:height_higher,:][:-1, :-1].ravel())              # update substance data
    ax.set_title(substance + ' on date: ' + str(datetime.date(year_data, 1, 1) + datetime.timedelta(days=i+date)))
                                                                # updating title for 2d array plot

    temp_lat = temp[:, :, i + date][:, :]                   # selecting the day temp_lat array
    temp_lat_new = np.zeros((z.shape[0], (temp_lat[:, :]).shape[1]))  # creating 2d zero matrix
    for l in range(z.shape[1]):  # loop over all afore selected latitudes
        interp_func = interp1d(old, temp_lat[:, l])  # interpolate temperature array
        temp_lat_new[:, l] = interp_func(new)        # putting on new "grid"
        temp_lat_new = np.ma.array(temp_lat_new, mask=np.isnan(temp_lat_new))  # masking the NaN values
    z = z / temp_lat_new              # divide z by temperatures
    for g in range(z.shape[1]):              # loop over latitudes
        z[:, g] = z[:, g] * height_slice * 100000 * 10000 * y * 100 / (1.38064852 * 10**(-23)) * (1/10**6)
                                                                               # conversion to cm and molec per cm3
    z_new = (z[height_lower:height_higher,:].sum(axis=0) / (2.6867 * 10 ** 20)) /substance_unit_factor
                                                                               # converting to Dobson units
    for o in range(len(pos)):                                                  # loop for updating bar blot
        j[o].set_height(z_new[o])                                              # updating each bar in bar plot
    draw(), pause(pause_time)                                                  # drawing the new plot and waiting

plt.show()                          # show plot window
```

Listing 1: Program 1

```
import numpy as np
import matplotlib.pyplot as plt
import mat73
import datetime
from mpl_toolkits.axes_grid1 import make_axes_locatable
import matplotlib.dates as mdates


"""Creating a colormap plot that displays the deviation from the average column value over latitude and time"""


fig = plt.figure(figsize=(12, 8), constrained_layout=False)     # create plot window
```

```python
ax = fig.add_subplot(1, 1, 1)                                          # and plot area 1
plt.gcf().subplots_adjust(right=0.8)                                   # create room on the right


year_data_start = 2015  # start year (min 2005)
year_data_end = 2020    # end year (max 2021)
year_array = np.arange(year_data_start, year_data_end + 1, 1)  # create year array

pressure_low = 10          # selecting lowest included pressure value out of p vector: (O3 8 - 37 (from 13 no holes at
                           # equator)) (ClO 9 - 16) (HCl 8-19) (CH3Cl 8 - 14) (SO2 8 - 10) (H2O 8 -37)
pressure_high = 18         # selecting highest included pressure value out of p vector

substance = 'O3'        # select the substance out of (print(mat.keys())) 'CH3Cl', 'ClO', 'H2O', 'HCl', 'O3', 'SO2',
                        # 'Temperature' (should not be used for ClO, HCl, CH3Cl and SO2, since they have a different
                        # sized temperature array, therefore wrong values)



if substance == 'ClO': # pressure values for ClO that are in data but are not usable are cut (see MLS instruction 3.6.5)
    if pressure_low <= 9:   # checking if lower pressure value is below 9
        pressure_low = 9    # setting lower pressure value to 9
    if pressure_high <= 9:  # checking if higher pressure value is below 9
        pressure_high = 9   # setting higher pressure value to 9



for pressure in np.arange(pressure_low, pressure_high + 1, 1):  # loop over selected pressure area
    print(pressure)     # printing the pressure, to see the progress

    for year_data in year_array:    # loop over selected years

        data_file = 'Data/MLS_gridded_' + str(year_data) + '-01-01_' + str(year_data) + '-12-31.mat'    # defining data
                                                                                                     # file name

        mat = mat73.loadmat(data_file, use_attrdict=True)['MLS_gridded_tmp']   # importing mat file and changing from
                                                                               # one dictionary key to keys in it


        if substance == 'CH3Cl':     # setting the substance units
            substance_unit = 'ppmv'
        if substance == 'ClO':
            substance_unit = 'ppmv'
        if substance == 'H2O':
            substance_unit = 'ppmv'
```

```python
        if substance == 'HCl':
            substance_unit = 'ppmv'
        if substance == 'O3':
            substance_unit = 'ppmv'
        if substance == 'SO2':
            substance_unit = 'ppmv'
        if substance == 'Temperature':
            substance_unit = 'ppmv'

        if substance_unit == 'ppmv':    # setting the substance unit factors for conversion to ppmv
            substance_unit_factor = 1
        if substance_unit == 'ppbv':
            substance_unit_factor = 1000
        if substance_unit == 'pptv':
            substance_unit_factor = 1000000


        y = mat.get(substance, 'Not Found').get('Pressure', 'Not Found')  # extracting pressure vec of one substance out
                                                                          # of dict



        z = mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat', 'Not Found')[pressure,:,:]
                                                                          # extracting one substance 2d array out of dict
        x = np.arange(-90, 91, 1)                         # creating latitude vector
        temp = mat.get('Temperature', 'Not Found').get('Temperature' + '_flagCleaned_strat', 'Not Found')[:,:,:]
                                                                          # extracting the temperature array

        y = np.ma.array(y, mask=np.isnan(y))              # masking NaN values from pressure array
        z = np.ma.array(z, mask=np.isnan(z))              # masking NaN values from substance 2d array
        temp = np.ma.array(temp, mask=np.isnan(temp))     # masking NaN values from temperature array


        if year_data == 2005 or year_data == 2006 or year_data == 2007 or year_data == 2009 or year_data == 2010 or \
                year_data == 2011 or year_data == 2013 or year_data == 2014 or year_data == 2015 or year_data == 2017 \
                or year_data == 2018 or year_data == 2019 or year_data == 2021 or year_data == 2022:
                                                                          # checking if the year is not a leap year
            z = z[:,0:-1]     # cutting of last substance value (366) left from other years

        height = 145366.45 * (1 - (y / 1013.25) ** 0.190284) / 3.28084 / 1000  # calculating the heights from the

```

```python
    # pressure array in km

    height_slice = ((np.append(height[1:], 0) - height) * 0.5) + ((height - np.append(0, height[:-1])) * 0.5)
                                                                        # calculating the height slice to each height


    pos = np.arange(-90,90+1,1)                         # creating latitude array (0 = equator)

    if year_data == 2005 or year_data == 2006 or year_data == 2007 or year_data == 2009 or year_data == 2010 or \
            year_data == 2011 or year_data == 2013 or year_data == 2014 or year_data == 2015 or year_data == 2017 \
            or year_data == 2018 or year_data == 2019 or year_data == 2021 or year_data == 2022:
                                                                        # checking if the year is not a leap year
        temp = temp[:,:,0:-1]       # cutting of last temperature value (366) left from other years

    if not substance == 'Temperature':          # checking if the substance is not Temperature
        z = np.divide(z,temp[pressure])         # dividing the mixing ratio by Temperature



    if year_data == year_data_start:        # checking if it is the start year in the loop
        all_z = z                           # filling all_z with z
    else:                                   # for all other years
        all_z = np.append(all_z, z, axis=1)         # appending all_z with z
    all_z = np.ma.array(all_z, mask=np.isnan(all_z))            # masking the NaN values for all_z substance values

if not substance == 'Temperature':          # checking if the substance is not Temperature
    all_z = all_z * y[pressure] * 100 / (1.38064852 * 10**(-23)) * (1/10**6)  # transforming ppmv/T to molec per cm3

if not substance == 'Temperature':          # checking if the substance is not Temperature
    all_z = all_z * height_slice[pressure] * 100000 * 10000         # multiplying all_z by height and unit correction
                                                                    # for just on area

if not substance == 'Temperature':          # checking if the substance is not Temperature
    if pressure == pressure_low:            # checking if it is the lowest pressure
        all_p_z = all_z / (2.6867 * 10 ** 20)       # converting to Dobson units
    else:                                   # for all other pressures
        all_p_z = np.add(all_p_z, (all_z / (2.6867 * 10 ** 20)))
```

```
    # converting to Dobson units
    else:            # excecption for substance Temperature
        if pressure == pressure_low:    # checking if it is the lowest pressure
            all_p_z = all_z             # leaving the values just transferring them to all_p_z
        else:                           # for all other pressures
            all_p_z = np.add(all_p_z, (all_z))      # leaving the values just transferring them to all_p_z

if substance == 'Temperature':                       # check if substance is Temperature
    all_p_z = all_p_z / (pressure_high - pressure_low + 1)          # averaging temperature over heights


start_date = datetime.date(year_data_start,1,1)    # extracting the start date from date string
end_date = datetime.date(year_data_end,12,31)      # extracting the end date from date string
date = np.arange(start_date, end_date + datetime.timedelta(days=1), dtype='datetime64[D]')
                                                   # creating a date array from start to end

all_p_z = np.subtract(all_p_z, np.average(all_p_z))     # converting values to differences from the average


h = ax.pcolormesh(date, x, all_p_z/substance_unit_factor, cmap='RdBu_r')  # creating the first heatmap of z array over
                                                                          # latitude vector and pressure vector


ax.xaxis.set_major_locator(mdates.YearLocator())               # setting the x axis major locators years
ax.xaxis.set_major_formatter(mdates.DateFormatter('%Y'))        # setting years for labeling on the x axis
ax.xaxis.set_minor_locator(mdates.MonthLocator())              # setting the x axis minor locators to month



divider = make_axes_locatable(ax)                            # move colorbar
cax = plt.gcf().add_axes([0.85, 0.15, 0.05, 0.7])            # new axes for color bar
cb = fig.colorbar(h, cax=cax, orientation='vertical')            # creating colorbar
cb.set_label('O3 deviation from average in Dobson units', rotation = 90)
    # labeling and rotating it



ax.set_xlabel('Time', fontsize=15)                         # labeling x axis
ax.set_ylabel('Latitude in   ', fontsize=15)                  # labeling y axis
ax.set_title(substance + ' column from height: ' + str(np.round(height[
```

```
    pressure_low], 2)) + ' km to height: '
            + str(np.round(height[pressure_high], 2)) + ' km')      # labeling plot including substance and height range

plt.show()                          # show plot window
```

Listing 2: Program 2

```
import numpy as np
import matplotlib.pyplot as plt
import mat73
import datetime
from calendar import monthrange
import os
from scipy.interpolate import interp1d


"""Creating a latitude range averaged and month averaged substance concentration plotted for each height over the
concentration comparing particular months between years highlighting one year with plot on the right that displays the
concentration deviation from the average for each height for aforementioned highlighted year"""


substance_list = ['HCl', 'ClO', 'H2O', 'O3', 'CH3Cl', 'SO2']     # making array with all substance names

for substance in substance_list:              # loop through all substances
    print(substance)                          # print substance name to know progress

    year_start = 2005                         # setting starting year
    year_end = 2022                           # setting ending year
    northern_latitude = 60  # selecting more northern latitude barrier (only works, if both are either southern or
                            # northern hemisphere)
    southern_latitude = 45  # selecting more southern latitude barrier
    height_l = -1  # selecting lower limit for selected height range (-1 selects whole range, lowest pressure)
    height_h = -1  # selecting higher limit for selected height range (-1 selects whole range, highest pressure)
    mon_list = np.arange(1, 13, 1)                # making array with all months of the year
    height_lower = height_l  # for resetting these values after new substance and just for putting the setting up there
    height_higher = height_h  # for resetting these values after new substance and just for putting the setting up there

    for mon in mon_list:         # loop over month array
        print(mon)               # print month to know progress
        if mon >= 10 and year_end == 2022:      # checking for October, November and December of 2022
            year_end = 2021                     # cutting out 2022 October, November and December

        fig = plt.figure(figsize=(12, 8), constrained_layout=False)  # creating plot window
        ax = fig.add_subplot(1, 2, 1)                # creating plot area 1
```

```
    ay = fig.add_subplot(1, 2, 2)                # creating plot area 2

    if substance == 'CH3Cl':  # setting the x axis limits, cutting off heights with missing data, selecting
                              # maximum/minimum data if selected by -1 and setting unit for all substances
        substance_unit = 'ppmv'
        xlim_low = -0.1*10**9
        xlim_high = 1.6*10**9
        if height_lower == -1:
            height_lower = 6
        if height_higher == -1:
            height_higher = 14
        if not((northern_latitude >= 25 and southern_latitude >= 25) or (
                northern_latitude <= -25 and southern_latitude <= -25)):
            print('Watch out! Changed height!')
            if height_lower <= 8:
                height_lower = 8
            if height_higher <= 8:
                height_higher = 8
    if substance == 'ClO':
        substance_unit = 'ppmv'
        xlim_low = 0 * 10 ** 8
        xlim_high = 1 * 10 ** 8
        if height_lower == -1:
            height_lower = 9
        if height_higher == -1:
            height_higher = 16
        if not ((northern_latitude >= 90 and southern_latitude >= 90) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 9:
                height_lower = 9
            if height_higher <= 9:
                height_higher = 9
    if substance == 'H2O':
        substance_unit = 'ppmv'
        xlim_low = -0.20*10**13
        xlim_high = 1.2*10**13
        if height_lower == -1:
            height_lower = 7
        if height_higher == -1:
            height_higher = 37
        if not((northern_latitude >= 90 and southern_latitude >= 90) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 13:
                height_lower = 13
            if height_higher <= 13:
                height_higher = 13
    if substance == 'HCl':
        substance_unit = 'ppmv'
```

```
            xlim_low = -0.25*10**9
            xlim_high = 4*10**9
            if height_lower == -1:
                height_lower = 7
            if height_higher == -1:
                height_higher = 19
        if substance == 'O3':
            substance_unit = 'ppmv'
            xlim_low = -0.25 * 10**12
            xlim_high = 6.5 * 10**12
            if height_lower == -1:
                height_lower = 8
            if height_higher == -1:
                height_higher = 37
            if not((northern_latitude >= -70 and southern_latitude >= -70) or (
                    northern_latitude <= -90 and southern_latitude <= -90)):
                print('Watch out! Changed height!')
                if height_lower <= 10:
                    height_lower = 10
                if height_higher <= 10:
                    height_higher = 10
            if not((northern_latitude >= 45 and southern_latitude >= 45) or (
                    northern_latitude <= -45 and southern_latitude <= -45)):
                print('Watch out! Changed height!')
                if height_lower <= 13:
                    height_lower = 13
                if height_higher <= 13:
                    height_higher = 13
        if substance == 'SO2':
            substance_unit = 'ppmv'
            xlim_low = -4*10**9
            xlim_high = 6*10**9
            if height_lower == -1:
                height_lower = 5
            if height_higher == -1:
                height_higher = 11
            if not((northern_latitude >= 90 and southern_latitude >= 90) or (
                    northern_latitude <= -90 and southern_latitude <= -90)):
                print('Watch out! Changed height!')
                if height_lower <= 8:
                    height_lower = 8
                if height_higher <= 8:
                    height_higher = 8
            if not((northern_latitude >= 90 and southern_latitude >= 90) or (
                    northern_latitude <= -90 and southern_latitude <= -90)):
                print('Watch out! Changed height!')
                if height_lower >= 10:
                    height_lower = 10
                if height_higher >= 10:
                    height_higher = 10
```

```
    if substance == 'Temperature':
        substance_unit = 'K'
        xlim_low = -0.1*10**21
        xlim_high = 1.4*10**21
        if height_lower == -1:
            height_lower = 8
        if height_higher == -1:
            height_higher = 37
        if not((northern_latitude >= -70 and southern_latitude >= -70) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 10:
                height_lower = 10
            if height_higher <= 10:
                height_higher = 10
        if not((northern_latitude >= 45 and southern_latitude >= 45) or (
                northern_latitude <= -45 and southern_latitude <= -45)):
            print('Watch out! Changed height!')
            if height_lower <= 14:
                height_lower = 14
            if height_higher <= 14:
                height_higher = 14

    if substance_unit == 'ppmv':        # setting the substance unit factors for conversion to ppmv
        substance_unit_factor = 1
    if substance_unit == 'ppbv':
        substance_unit_factor = 1000
    if substance_unit == 'pptv':
        substance_unit_factor = 1000000



    for year_data in np.arange(year_start, year_end+1, 1):                    # looping over selected years
        print(year_data)                                                       # printing the year, to see the progress

        data_file = 'Data/MLS_gridded_' + str(year_data) + '-01-01_' + str(year_data) + '-12-31.mat'  # defining
                                                                              # file name
        mat = mat73.loadmat(data_file, use_attrdict=True)['MLS_gridded_tmp']  # importing mat file and changing from
                                                                              # one dictionary key to keys in it


        start_date = datetime.date(1,1,1) + \
                     datetime.timedelta(days=mat.get(substance, 'Not Found').get('Date', 'Not Found')[0]-365-2)
                                                                              # extrating the start date from date string

```

```
            if year_data == 2004 or year_data == 2008 or year_data == 2012 or year_data == 2016 or year_data == 2020:
 # checking if it's a leap year
                end_date = datetime.date(1,1,1) + \
                           datetime.timedelta(days=mat.get(substance, 'Not Found').get('Date', 'Not Found')[-1]-365-2)
 # extrating the end date from date string

            if year_data == 2005 or year_data == 2006 or year_data == 2007 or year_data == 2009 or year_data == 2010 \
                    or year_data == 2011 or year_data == 2013 or year_data == 2014 or year_data == 2015 or year_data \
                    == 2017 or year_data == 2018 or year_data == 2019 or year_data == 2021 or year_data == 2022:
 # checking if it's not a leap year
                end_date = datetime.date(1, 1, 1) + \
                           datetime.timedelta(days=mat.get(substance, 'Not Found').get('Date', 'Not Found')[-2] - 365-2)
 # extrating the end date from date string

            p = mat.get(substance, 'Not Found').get('Pressure', 'Not Found')  # extracting pressure vec of one substance
   # out of dict

            z = mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat', 'Not Found')[:,:,:]
                                                                        # extracting substance 2d arrays for all dates out of dict

            x = np.arange(-90, 91, 1)                                   # creating latitude vector
            temp = mat.get('Temperature', 'Not Found').get('Temperature' + '_flagCleaned_strat', 'Not Found')[:,:,:]
                                                                        # extracting the temperature array

            new = np.arange(0, 55, 55/z.shape[0])                       # making new temperature grid
            old = np.arange(0, 55, 1)                                   # making old temperature grid

            p = np.ma.array(p, mask=np.isnan(p))                        # masking NaN values from pressure array
            z = np.ma.array(z, mask=np.isnan(z))                        # masking NaN values from substance 2d array
            temp = np.ma.array(temp, mask=np.isnan(temp))               # masking NaN values from temperature array

            pos = np.arange(-90, 90+1, 1)                               # creating latitude array (0 = equator)



```

```
        if northern_latitude > 0 and southern_latitude > 0:    #
checking for northern hemisphere
                                                                #  ->
leaving a and be "as they are"
            a_latitude = abs(abs(northern_latitude) - 90)      #
converting it to angle to work with later formula
            b_latitude = abs(abs(southern_latitude) - 90)      #
converting second angle
        if northern_latitude < 0 and southern_latitude < 0:    #
checking for southern hemisphere
                                                                #  ->
therefore changing roles of a and b
            b_latitude = abs(abs(northern_latitude) - 90)      #
converting it to angle to work with later formula
            a_latitude = abs(abs(southern_latitude) - 90)      #
converting second angle


        years = np.arange(start_date.year, end_date.year + 1, 1)    #
making array of all years from start to end
        k = 0                                                        #
set start value for k
        mean_month = np.empty(shape=len(years))                      #
create empty mean_month vector

        for y in years:              # loop over years
            for t in np.arange((datetime.date(y, mon, 1) - start_date).
days,
                               (datetime.date(y, mon, monthrange(y, mon
)[1]) - start_date).days + 1, 1):
 # loop over all days of the selected month
                print(t)             # print day in moth to check
progress
                z_lat = z[:, :, t][:, southern_latitude + 90:
northern_latitude + 90 + 1]
                                                                  #
slices the latitudes we selected out of the array
                temp_lat = temp[:, :, t][:, southern_latitude + 90:
northern_latitude + 90 + 1]
                                                                  #
slices the latitudes we selected out of the array
                total_A = 0                               # resetting
total area
                temp_lat_new = np.zeros((z.shape[0], (temp_lat[:, :]).
shape[1]))         # creating 2d zero matrix
                for l in range(z_lat.shape[1]):           # loop over all
 afore selected latitudes
                    if substance == 'HCl' or substance == 'ClO' or
substance == 'SO2' or substance == 'CH3Cl':
                                                                  #
checking if substance is HCl, ClO, SO2 or CH3Cl
                        interp_func = interp1d(old, temp_lat[:, l])
  # interpolate temperature array

                    if substance == 'HCl' or substance == 'ClO' or
substance == 'SO2' or substance == 'CH3Cl':
                                                                  #
```

```
checking if substance is HCl, ClO, SO2 or CH3Cl
                        temp_lat_new[:, l] = interp_func(new)  # putting on new "grid"

                    else:                          # checking if substance is not HCl, ClO, SO2 or CH3Cl
                        temp_lat_new[:, l] = temp_lat [:, l]     # putting old temp array as new one

                    temp_lat_new = np.ma.array(temp_lat_new, mask=np.isnan(temp_lat_new))   # masking the NaN values

                    z_lat[:, l] = z_lat[:, l] * (np.cos(np.radians(a_latitude+l-0.5)) -
                                                 np.cos(np.radians(a_latitude+l+0.5)))
                                    # multiplying each latitude vector with "area" (without radius and circumference)

                    z_lat[:, l] = z_lat[:, l] / temp_lat_new[:, l]  # divide mixing ratio by temperature
                    if any(z_lat[:, l].mask == False):                  # checking if any array holds any data
                        total_A = total_A + (np.cos(np.radians(a_latitude+l - 0.5)) -
                                             np.cos(np.radians(a_latitude+l + 0.5)))
                                    # calculation of the whole selected area without the masked latitudes
                z_lat_sum = np.sum(z_lat, axis=1)                      # summing 2d array over all latitudes -> vector
                z_lat_mean = z_lat_sum / total_A                       # division by the whole selected area

                if k == 0:                                             # entering in first try
                    c = z_lat_mean                                     # filling the c matrix with vectors of days
                    d = z_lat_mean.mask == True                        # filling the d matrix with mask vectors of days

                    k = 1                                              # setting the k to 1 for entering else branch
                else:                                                  # else branch
                    c = np.c_[c, z_lat_mean]                           # filling the c matrix with vectors of days
                    d = np.c_[d, z_lat_mean.mask == True]              # filling the d matrix with mask vectors of days

            k = 0                                         # resetting the k
            c = np.ma.array(c, mask=d)                    # putting the mask back on (because it got lost with np.c_)
            c_mean = c.mean(axis=1)                       # creating the mean over the days of the certain month
            c = np.empty(shape=len(p))                    # empty c

            c_mean = c_mean * p * 100 / (1.38064852 * 10**(-23)) *
```

```
(1/10**6)  # transition to molec per cm3



            height = 145366.45*(1-(p/1013.25)**0.190284) / 3.28084 /1000    # calculating height array in km

            y_axis = 'height'                       # choose height or pressure
            if y_axis == 'p':                       # checking if pressure was selected
                if year_data == year_start:         # checking if the year is the start year
                    avg = c_mean                    # setting the calculated values into avg array
                    avg_mask = c_mean.mask == True       # creating mask again
                    avg = np.ma.array(avg, mask=avg_mask)        # putting the mask back on
                else:                               # checking if the year is not the start year
                    avg = np.c_[avg, c_mean]    # expanding the avg array with calculated values
                    avg_mask = np.c_[avg_mask, c_mean.mask == True]  # creating mask again for 2d
                    avg = np.ma.array(avg, mask=avg_mask)        # filling avg matrix with masked vectors of days

                ax.plot(c_mean/substance_unit_factor, p, marker='D', color='black', label=str(y))

    # plot function of mean over pressure
            if y_axis == 'height':                  # checking if height was selected
                if year_data == year_start:         # checking if the year is the start year
                    avg = c_mean                    # setting the calculated values into avg array
                    avg_mask = c_mean.mask == True       # creating mask again
                    avg = np.ma.array(avg, mask=avg_mask)        # putting the mask back on
                else:                               # checking if the year is not the start year
                    avg = np.c_[avg, c_mean]    # expanding the avg array with calculated values
                    avg_mask = np.c_[avg_mask, c_mean.mask == True]  # creating mask again for 2d
                    avg = np.ma.array(avg, mask=avg_mask)        # filling avg matrix with masked vectors of days
                ax.plot(c_mean/substance_unit_factor, height, marker='D', color='black', label=str(y), zorder=1)

    # plot function of mean over height


        if mon == 1:                    # putting numbers in connection to the month names
```

```
            month = 'January'
        if mon == 2:
            month = 'February'
        if mon == 3:
            month = 'March'
        if mon == 4:
            month = 'April'
        if mon == 5:
            month = 'May'
        if mon == 6:
            month = 'June'
        if mon == 7:
            month = 'July'
        if mon == 8:
            month = 'August'
        if mon == 9:
            month = 'September'
        if mon == 10:
            month = 'October'
        if mon == 11:
            month = 'November'
        if mon == 12:
            month = 'December'


        if y_axis == 'height':       # checking if y axis is height
            ax.set_ylim(height[height_lower], height[height_higher])  # setting the limits (everything else 0)
            ay.set_ylim(height[height_lower], height[height_higher])  # setting the limits (everything else 0)
            ax.set_ylabel('approximate altitude in km', fontsize=15)  # labeling y axis
        if y_axis == 'p':            # checking if y axis is pressure
            plt.ylim(p[np.argwhere(~np.isnan(z))[0][0]], p[np.argwhere(~np.isnan(z))[-1][0]])
                                                                      # setting the limits (everything else 0)
            plt.yscale('log')                                         # making y scale log
            ax.set_ylabel('pressure in hPa', fontsize=15)             # labeling y axis

        ax.set_xlim(xlim_low/substance_unit_factor, xlim_high/substance_unit_factor)
                                                                      # setting the limits for x (manual adjustment)
        if substance == 'O3' or substance == 'CH3Cl':            # setting y limits for all substances separately
            if abs(xlim_low) > abs(xlim_high):
                ay.set_xlim(-abs(xlim_low)/substance_unit_factor/12, abs(xlim_low)/substance_unit_factor/12)
            else:
                ay.set_xlim(-abs(xlim_high)/substance_unit_factor/12, abs(xlim_high)/substance_unit_factor/12)
        elif substance == 'HCl' or substance == 'H2O':
            if abs(xlim_low) > abs(xlim_high):
                ay.set_xlim(-abs(xlim_low)/substance_unit_factor /8, abs(xlim_low)/substance_unit_factor/8)
```

```
            else:
                ay.set_xlim(-abs(xlim_high)/substance_unit_factor /8, abs(xlim_high)/substance_unit_factor/8)
        else:
            if abs(xlim_low) > abs(xlim_high):
                ay.set_xlim(-abs(xlim_low)/substance_unit_factor/2, abs(xlim_low)/substance_unit_factor/2)
            else:
                ay.set_xlim(-abs(xlim_high)/substance_unit_factor/2, abs(xlim_high)/substance_unit_factor/2)
        ax.set_title('Average ' + substance + ' concentration in stratosphere \n between latitudes of '
                            + str(southern_latitude) + '  and '+ str(northern_latitude) + '  in ' + month, fontsize=10)
                                    # setting title
        ax.set_xlabel('average concentration of ' + substance + ' in molecules/cm ', fontsize=10)  # labeling x axis
    if os.path.exists(str(substance) + '_from_' + str(southern_latitude) + '_to_ '
                        + str(northern_latitude) + ' _ +average') is False:            # checking if path exists not
        os.mkdir(str(substance) + '_from_' + str(southern_latitude) + ' _to_ '
                    + str(northern_latitude) + ' _ +average')
                    # creating path
    for year_select in range(year_end - year_start + 1):
# loop over years
        ay.set_xlabel('deviation from average concentration \n of ' + substance + ' in molecules/cm  for '
                        + str(year_start + year_select), fontsize=10)
# labeling x axis
        if year_select > 0:
# checking if first year
            ax.get_lines()[year_select - 1].set_color("black")
# putting last selected year back to black
        #ax.get_lines()[2017-year_start].set_color("lime")
# can be turned on to highlight special years
        #ax.get_lines()[2017-year_start].set_zorder(year_select + 2)
# can be turned on to highlight special years
        #ax.get_lines()[2019-year_start].set_color("cyan")
# can be turned on to highlight special years
        #ax.get_lines()[2019-year_start].set_zorder(year_select + 2)
# can be turned on to highlight special years
        ax.get_lines()[year_select].set_color("red")
# making selected year red in plot
        ax.get_lines()[year_select].set_zorder(year_select + 2)
# putting selected year in foreground
        ax.legend()
# creating legend

        ay.axvline(x=0, color='black', zorder=1)
#
        ay.plot((avg[:,year_select] - np.array(avg.mean(axis=1)))
                /substance_unit_factor, height, marker='D', color='red', label=str(y), zorder=2)
        # ax.plot(np.array(avg.mean(axis=1))
        # /substance_unit_factor, height, marker='D', color='cyan',
```

```
label=str(y), zorder=3) # can be activated to

                        # put average in plot
            print('producing images...')
# print hint when creating pictures
            if os.path.exists(str(substance) + '_from_' + str(
southern_latitude) + '_to_' + str(northern_latitude)
                                + '_+average' + '/' + str(substance) + '
_from_' + str(southern_latitude) + '_to_'
                                + str(northern_latitude) + '_' + str(
year_select + year_start)) is False:

                # checking if path exists not
                os.mkdir(str(substance) + '_from_' + str(southern_latitude)
 + '_to_' + str(northern_latitude)
                         + '_+average' + '/' + str(substance) + '_from_'
+ str(southern_latitude) + '_to_'
                         + str(northern_latitude) + '_' + str(year_select
 + year_start))       # creating path
            plt.savefig(str(substance) + '_from_' + str(southern_latitude)
+ '_to_' + str(northern_latitude)
                        + '_+average' + '/' + str(substance) + '_from_' +
 str(southern_latitude) + '_to_'
                        + str(northern_latitude) + '_' + str(year_select
+ year_start) + '/' + str(mon) + '_' + month
                        + '_' + str(year_select + year_start) + '_' +
substance + '_' + str(southern_latitude)
                        + '-' + str(northern_latitude) + '.png',
bbox_inches='tight')        # safe picture to directory
            if os.path.exists(str(substance) + '_from_' + str(
southern_latitude) + '_to_' + str(northern_latitude)
                                + '_+average' + '/' + str(substance) + '
_from_' + str(southern_latitude)
                                + '_to_' + str(northern_latitude) + '_all
') is False:     # checking if path exists not
                os.mkdir(str(substance) + '_from_' + str(southern_latitude)
 + '_to_' + str(northern_latitude)
                         + '_+average' + '/' + str(substance) + '_from_'
+ str(southern_latitude) + '_to_'
                         + str(northern_latitude) + '_all')
                        # creating path
            plt.savefig(str(substance) + '_from_' + str(southern_latitude)
+ '_to_' + str(northern_latitude)
                        + '_+average' + '/' + str(substance) + '_from_' +
 str(southern_latitude) + '_to_'
                        + str(northern_latitude) + '_all' + '/' + str(mon
) + '_' + month + '_'
                        + str(year_select + year_start) + '_' + substance +
 '_' + str(southern_latitude) + '-'
                        + str(northern_latitude) + '.png', bbox_inches='
tight')              # safe picture to directory
            ay.cla()                              # clear axis
            if substance == 'O3' or substance == 'CH3Cl':           #
setting y limits for all substances separately
                if abs(xlim_low) > abs(xlim_high):
                    ay.set_xlim(-abs(xlim_low) / substance_unit_factor /
12, abs(xlim_low) / substance_unit_factor / 12)
                else:
```

```
                    ay.set_xlim(-abs(xlim_high) / substance_unit_factor / 12,
                                abs(xlim_high) / substance_unit_factor / 12)
            elif substance == 'HCl' or substance == 'H2O':
                if abs(xlim_low) > abs(xlim_high):
                    ay.set_xlim(-abs(xlim_low) / substance_unit_factor /8, abs(xlim_low) / substance_unit_factor /8)
                else:
                    ay.set_xlim(-abs(xlim_high) / substance_unit_factor /8, abs(xlim_high) / substance_unit_factor /8)
            else:
                if abs(xlim_low) > abs(xlim_high):
                    ay.set_xlim(-abs(xlim_low) / substance_unit_factor / 2, abs(xlim_low) / substance_unit_factor / 2)
                else:
                    ay.set_xlim(-abs(xlim_high) / substance_unit_factor / 2, abs(xlim_high) / substance_unit_factor / 2)
            ay.set_ylim(height[height_lower], height[height_higher])  # setting the limits (everything else 0)
            ay.set_xlabel('deviation from average concentration \n of ' + substance + ' in molecules/cm  for '
                          + str(year_select), fontsize=10)            # labeling x axis
        ax.cla()                                      # clear axis
```

Listing 3: Program 3

```
import numpy as np
import matplotlib.pyplot as plt
import mat73
import datetime
from calendar import monthrange
import os
from scipy.interpolate import interp1d
from matplotlib.ticker import NullFormatter
from matplotlib.dates import MonthLocator, DateFormatter


"""Creating a latitude range averaged substance column value for every day plotted over an annual axis comparing
years highlighting one year and the average in color"""


year_start = 2005                # selecting starting year
year_end = 2022                  # selecting ending year
height_l = -1   # selecting lower limit for selected height range (-1 selects whole range, lowest pressure)
height_h = -1   # selecting higher limit for selected height range (-1 selects whole range, highest pressure)

northern_latitude = 60  # selecting more northern latitude barrier (only works, if both are either southern or northern
                        # hemisphere)
southern_latitude = 45  # selecting more southern latitude barrier

mon_list = np.arange(1, 13, 1)      # making array with all months of the year

```

```python

substance_list = ['CH3Cl', 'H2O', 'ClO', 'SO2', 'HCl', 'O3']  # making array with all substance names

for substance in substance_list:         # loop through all substances
    print(substance)                     # print substance name to know progress

    height_lower = height_l  # for resetting these values after new substance and just for putting the setting up there
    height_higher = height_h # for resetting these values after new substance and just for putting the setting up there

    fig = plt.figure(figsize=(12, 8), constrained_layout=False)  # create plot window
    ax = fig.add_subplot(1, 1, 1)  # and plot area 1


    if substance == 'CH3Cl':  # setting the limits for all substances
        substance_unit = 'ppmv'
        xlim_low = -0.1 * 10 ** 9
        xlim_high = 1.6 * 10 ** 9
        if height_lower == -1:
            height_lower = 6
        if height_higher == -1:
            height_higher = 14
        if not ((northern_latitude >= 25 and southern_latitude >= 25) or (
                northern_latitude <= -25 and southern_latitude <= -25)):
            print('Watch out! Changed height!')
            if height_lower <= 8:
                height_lower = 8
            if height_higher <= 8:
                height_higher = 8
    if substance == 'ClO':
        substance_unit = 'ppmv'
        xlim_low = -6 * 10 ** 8
        xlim_high = 1 * 10 ** 8
        if height_lower == -1:
            height_lower = 9
        if height_higher == -1:
            height_higher = 16
        if not ((northern_latitude >= 90 and southern_latitude >= 90) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 9:
                height_lower = 9
            if height_higher <= 9:
                height_higher = 9
    if substance == 'H2O':
        substance_unit = 'ppmv'
        xlim_low = -0.25 * 10 ** 13
        xlim_high = 4.5 * 10 ** 13
        if height_lower == -1:
            height_lower = 7
        if height_higher == -1:
            height_higher = 37
        if not ((northern_latitude >= 90 and southern_latitude >= 90) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
```

```
            print('Watch out! Changed height!')
            if height_lower <= 13:
                height_lower = 13
            if height_higher <= 13:
                height_higher = 13
    if substance == 'HCl':
        substance_unit = 'ppmv'
        xlim_low = -0.25 * 10 ** 9
        xlim_high = 4 * 10 ** 9
        if height_lower == -1:
            height_lower = 7
        if height_higher == -1:
            height_higher = 19
    if substance == 'O3':
        substance_unit = 'ppmv'
        xlim_low = -0.25 * 10 ** 12
        xlim_high = 6.5 * 10 ** 12
        if height_lower == -1:
            height_lower = 8
        if height_higher == -1:
            height_higher = 37
        if not ((northern_latitude >= -70 and southern_latitude >= -70) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 10:
                height_lower = 10
            if height_higher <= 10:
                height_higher = 10
        if not ((northern_latitude >= 45 and southern_latitude >= 45) or (
                northern_latitude <= -45 and southern_latitude <= -45)):
            print('Watch out! Changed height!')
            if height_lower <= 13:
                height_lower = 13
            if height_higher <= 13:
                height_higher = 13
    if substance == 'SO2':
        substance_unit = 'ppmv'
        xlim_low = -4 * 10 ** 9
        xlim_high = 6 * 10 ** 9
        if height_lower == -1:
            height_lower = 5
        if height_higher == -1:
            height_higher = 11
        if not ((northern_latitude >= 90 and southern_latitude >= 90) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 8:
                height_lower = 8
            if height_higher <= 8:
                height_higher = 8
        if not ((northern_latitude >= 90 and southern_latitude >= 90) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower >= 10:
                height_lower = 10
            if height_higher >= 10:
                height_higher = 10
```

```
    if substance == 'Temperature':
        substance_unit = 'K'
        xlim_low = -0.1 * 10 ** 21
        xlim_high = 1.4 * 10 ** 21
        if height_lower == -1:
            height_lower = 8
        if height_higher == -1:
            height_higher = 37
        if not ((northern_latitude >= -70 and southern_latitude >= -70) or (
                northern_latitude <= -90 and southern_latitude <= -90)):
            print('Watch out! Changed height!')
            if height_lower <= 10:
                height_lower = 10
            if height_higher <= 10:
                height_higher = 10
        if not ((northern_latitude >= 45 and southern_latitude >= 45) or (
                northern_latitude <= -45 and southern_latitude <= -45)):
            print('Watch out! Changed height!')
            if height_lower <= 14:
                height_lower = 14
            if height_higher <= 14:
                height_higher = 14

    if substance_unit == 'ppmv':                # setting the substance unit factors for conversion to ppmv
        substance_unit_factor = 1
    if substance_unit == 'ppbv':
        substance_unit_factor = 1000
    if substance_unit == 'pptv':
        substance_unit_factor = 1000000

    for year_data in np.arange(year_start, year_end+1, 1):       # looping over selected years
        print(year_data)                                          # printing the year, to see the progress

        data_file = 'Data/MLS_gridded_' + str(year_data) + '-01-01_' + str(year_data) + '-12-31.mat'  # defining
                                            # file name
        mat = mat73.loadmat(data_file, use_attrdict=True)['MLS_gridded_tmp']   # importing mat file and changing from
            # one dictionary key to keys in it



        start_date = datetime.date(1,1,1) \
                     + datetime.timedelta(days=mat.get(substance, 'Not Found').get('Date', 'Not Found')[0]-365-2)
                                                                                                       # extrating the start date from date string

        if year_data == 2004 or year_data == 2008 or year_data == 2012 or year_data == 2016 or year_data == 2020:
                                                                                                       # checking if it's a leap year
```

```
            end_date = datetime.date(1,1,1) \
                       + datetime.timedelta(days=mat.get(substance, 'Not Found').get('Date', 'Not Found')[-1]-365-2)
                                                                                        # extrating the end date from date string

        if year_data == 2005 or year_data == 2006 or year_data == 2007 or year_data == 2009 or year_data == 2010 \
                or year_data == 2011 or year_data == 2013 or year_data == 2014 or year_data == 2015 or year_data \
                == 2017 or year_data == 2018 or year_data == 2019 or year_data == 2021 or year_data == 2022:
                                                                                        # checking if it's not a leap year
            end_date = datetime.date(1, 1, 1) \
                       + datetime.timedelta(days=mat.get(substance, 'Not Found').get('Date', 'Not Found')[-2] - 365 - 2)
                                                                                        # extrating the end date from date string


        p = mat.get(substance, 'Not Found').get('Pressure', 'Not Found')
                                                                        # extracting pressure vec of one substance out of dict

        z = mat.get(substance, 'Not Found').get(substance + '_flagCleaned_strat', 'Not Found')[:,:,:]
                                                                        # extracting substance 2d arrays for all dates out of dict


        x = np.arange(-90, 91, 1)                               # creating latitude vector
        temp = mat.get('Temperature', 'Not Found').get('Temperature' + '_flagCleaned_strat', 'Not Found')[:,:,:]
                                                                        # extracting the temperature array

        new = np.arange(0, 55, 55/z.shape[0])                   # new temperature grid
        old = np.arange(0, 55, 1)                               # old temperature grid




        p = np.ma.array(p, mask=np.isnan(p))                    # masking NaN values from pressure array
        z = np.ma.array(z, mask=np.isnan(z))                    # masking NaN values from substance 2d array
        temp = np.ma.array(temp, mask=np.isnan(temp))           # masking NaN values from temperature array

        pos = np.arange(-90, 90+1, 1)                                   # creating latitude array (0 = equator)

        a_height = 145366.45*(1-(p[height_lower]/1013.25)**0.190284) / 3.28084 /1000          # calculating lower height
```

```
    b_height = 145366.45*(1-(p[height_higher]/1013.25)**0.190284) / 3.28084 /1000        # calculating upper height



    if northern_latitude > 0 and southern_latitude > 0:       # checking for northern hemisphere
                                                              # -> leaving a and be "as they are"
        a_latitude = abs(abs(northern_latitude) - 90)         # converting it to angle to work with later formula
        b_latitude = abs(abs(southern_latitude) - 90)         # converting second angle
    if northern_latitude < 0 and southern_latitude < 0:       # checking for southern hemisphere
                                                              # -> therefore changing roles of a and b
        b_latitude = abs(abs(northern_latitude) - 90)         # converting it to angle to work with later formula
        a_latitude = abs(abs(southern_latitude) - 90)         # converting second angle


    years = np.arange(start_date.year, end_date.year + 1, 1)  # making a vector of all years included in the dataset
    k = 0                                                      # set start value for k
    mean_month = np.empty(shape=len(years))                    # create empty mean_month vector

    for y in years:               # loop over years
        print(y)                  # print year for checking progress
        if y == 2004 or y == 2008 or y == 2012 or y == 2016 or y == 2020:        # checking if it's a leap year
            year_length = 366
        # setting leap year length


            year_array = np.arange(datetime.date(2004, 1, 1), datetime.date(2004, 12, 31)
                                   + datetime.timedelta(days=1), dtype='datetime64[D]')  # creating date array


        if y == 2005 or y == 2006 or y == 2007 or y == 2009 or y == 2010 or y == 2011 or y == 2013 or y == 2014 or \
                y == 2015 or y == 2017 or y == 2018 or y == 2019 or y == 2021 or y == 2022:

        # checking if it's not a leap year
            year_length = 365
        # setting not leap year length

            year_array = np.arange(datetime.date(2004,1,1), datetime.date(2004,12,30)
                                   + datetime.timedelta(days=1), dtype='datetime64[D]')  # creating date array

```

```python

        c_overtime = np.zeros(year_length)          # creating zero array with year length

        j = 0                                       # setting starting value for j
        for mon in mon_list:
            for t in np.arange((datetime.date(y, mon, 1) - start_date).days,
                               (datetime.date(y, mon, monthrange(y, mon)[1]) - start_date).days + 1, 1):
                                                                            # loop over all days of the selected month
                z_lat = z[:, :, t][:, southern_latitude + 90:northern_latitude + 90 + 1]
                                                                            # slices the latitudes we selected out of the array
                temp_lat = temp[:, :, t][:, southern_latitude + 90:northern_latitude + 90 + 1]
                                                                            # slices the latitudes we selected out of the array
                total_A = 0                                                 # resetting total area
                temp_lat_new = np.zeros((z.shape[0], (temp_lat[:, :]).shape[1]))         # creating 2d zero matrix
                for l in range(z_lat.shape[1]):                             # loop over all afore selected latitudes
                    if substance == 'HCl' or substance == 'ClO' or substance == 'SO2' or substance == 'CH3Cl':
                                                                            # checking if substance is HCl, ClO, SO2 or CH3Cl
                        interp_func = interp1d(old, temp_lat[:, l])
            # interpolate temperature array

                    if substance == 'HCl' or substance == 'ClO' or substance == 'SO2' or substance == 'CH3Cl':
                                                                            # checking if substance is HCl, ClO, SO2 or CH3Cl
                        temp_lat_new[:, l] = interp_func(new)
            # putting on new "grid"
                    else:                                   # checking if substance is not HCl, ClO, SO2 or CH3Cl
                        temp_lat_new[:, l] = temp_lat [:, l]    # putting old temp array as new one
                    temp_lat_new = np.ma.array(temp_lat_new, mask=np.isnan(temp_lat_new))  # masking the NaN values

                    if substance == 'HCl' or substance == 'ClO' or substance == 'SO2' or substance == 'CH3Cl':
                                                                            # checking if substance is HCl, ClO, SO2 or CH3Cl
                        if height_lower >= 5:
                            height_lower_psc = height_lower
                        else:
                            height_lower_psc = 5
                        if height_higher <= 12:
                            height_higher_psc = height_higher
                        else:
```

```
                            height_higher_psc = 12
                    else:                                   # checking if substance is not HCl, ClO, SO2 or CH3Cl
                        if height_lower >= 16:
                            height_lower_psc = height_lower
                        else:
                            height_lower_psc = 16
                        if height_higher <= 23:
                            height_higher_psc = height_higher
                        else:
                            height_higher_psc = 23

                    #if np.min(temp_lat_new[height_lower_psc:height_higher_psc,:])
                    #    <= 195.15 and np.min(temp_lat_new[height_lower_psc:height_higher_psc,:]) > 0:
                    #     print('Error: PSC.')
                    #     print(np.min(temp_lat_new[height_lower_psc:height_higher_psc,:]))
                    #     sys.exit(1)
                                                                            # can be activated to check for conditions for PSC
                    z_lat[:, l] = z_lat[:, l] * (np.cos(np.radians(a_latitude+l-0.5))
                                                - np.cos(np.radians(a_latitude+l+0.5)))
                                    # multiplying each latitude vector with "area" (without radius and circumference)
                    z_lat[:, l] = z_lat[:, l] / temp_lat_new[:, l]  # divide mixing ratio by temperature
                    if any(z_lat[:, l].mask == False):
                        total_A = total_A + (np.cos(np.radians(a_latitude+l - 0.5))
                                            - np.cos(np.radians(a_latitude+l + 0.5)))
                                    # calculation of the whole selected area without the masked latitudes
                z_lat_sum = np.sum(z_lat, axis=1)                   # summing 2d array over all latitudes -> vector
                z_lat_mean = z_lat_sum / total_A                    # division by the whole selected area
                if k == 0:                                          # entering in first try
                    c = z_lat_mean                                  # filling the c matrix with vectors of days
                    d = z_lat_mean.mask == True                     # filling the d matrix with mask vectors of days
                    k = 1                                           # setting the k to 1 for entering else branch
                else:                                               # else branch
                    c = np.c_[c, z_lat_mean]                        # filling the c matrix with vectors of days
                    d = np.c_[d, z_lat_mean.mask == True]           # filling the d matrix with mask vectors of days
            k = 0                                       # resetting the k
            c = np.ma.array(c, mask=d)                  # putting on the mask back on (because it got lost with np.c_)
```

```
                c_mean = c                                  # creating the mean over the days of the certain month
                c = np.empty(shape=len(p))                  # empty c

                for i in range(c_mean.shape[1]):            # looping over length of month

                    c_mean[:,i] = c_mean[:,i] * p * 100 / (1.38064852 * 10**(-23)) * (1/10**6)
                                                            # transition to molec per cm3


                    height = 145366.45*(1-(p/1013.25)**0.190284) / 3.28084 /1000   # calculating height array in km

                    height_slice = ((np.append(height[1:], 0) - height) * 0.5) \
                                   + ((height - np.append(0, height[:-1])) * 0.5)  # calculating height slices for each
      # height

                    c_mean[:,i] = c_mean[:,i] * height_slice * 100000 * 10000    # multiplying by height slice and unit
     # correction for just on area

                    c_overtime[j] = np.sum(c_mean[:, i][height_lower:height_higher+1] / (2.6867 * 10 ** 20))
     # conversion to Dobson units

                    j = j + 1
     # increasing j by 1

            if substance == 'O3':
     # excluding measurement errors
                if y == 2005:
                    c_overtime[193] = np.nan
                    c_overtime[95] = np.nan
                if y == 2009:
                    c_overtime[299] = np.nan
            if substance == 'ClO':
                if y == 2007:
                    c_overtime[200] = np.nan
                    c_overtime[210] = np.nan
                    c_overtime[211] = np.nan
            if substance == 'HCl':
                if y == 2007:
                    c_overtime[200] = np.nan
                    c_overtime[210] = np.nan
                    c_overtime[211] = np.nan


        y_axis = 'height'                                   # choose height or pressure
        if y_axis == 'p':                                   # checking if
```

```
pressure was selected
                plt.plot(year_array, c_overtime/substance_unit_factor,
color='black', label=str(y))
                                                            # plot function
 of mean
            if y_axis == 'height':                          # checking if
height was selected
                if year_data == 2022:                       # checking if
year is 2022
                    plt.plot(year_array[0:273], c_overtime[0:273]
                        / substance_unit_factor, color='black', label=
str(y), zorder=1)
                                                            # plot function
 of mean only for available values for 2022
                else:                                       # checking if
year is not 2022
                    plt.plot(year_array, c_overtime/substance_unit_factor,
color='black', label=str(y), zorder=1)
                                                            # plot function
 of mean

                if year_data == year_start:                 # checking if
year is start year
                    avg = c_overtime[0:365]/substance_unit_factor      #
dividing avg value by substance unit factor
                elif year_data == year_end:                 # checking if
year is end year
                    avg = avg + c_overtime[0:365] / substance_unit_factor
                                                        # dividing avg and
new years value sum by substance unit factor
                    plt.plot(year_array, np.array(avg)
                        / (year_end - year_start + 1), color='cyan',
label='average', zorder=2)
                                                            # plot function
 of mean
                else:                                   # checking if year
is not start or end year
                    avg = avg + c_overtime[0:365]/substance_unit_factor
                                                        # dividing avg and
new years value sum by substance unit factor
            del c_overtime                              # delete c_overtime
 variable


 if y_axis == 'height':                # checking if y axis is height
     ax.set_ylabel('Average ' + substance + ' in Dobson units', fontsize
=15)  # labeling y axis
 if y_axis == 'p':                     # checking if y axis is pressure
     plt.ylim(p[np.argwhere(~np.isnan(z))[0][0]], p[np.argwhere(~np.
isnan(z))[-1][0]])
                                       # setting the limits (everything
else 0)
     plt.yscale('log')                 # making y scale log
     ax.set_ylabel('pressure in hPa', fontsize=15)  # labeling y axis

 ax.xaxis.set_major_locator(MonthLocator())                 # setting
the x axis major locators to month
 ax.xaxis.set_minor_locator(MonthLocator(bymonthday=15))    # pushing
```

```
the x axis major locators to middle of month
    ax.xaxis.set_major_formatter(NullFormatter())            # setting the major formatter to nothing
    ax.xaxis.set_minor_formatter(DateFormatter('%b'))         # setting the minor formatter to month shorts

    ax.set_title('Average ' + substance + ' amount in atmosphere from ' + str(np.round(a_height, 2)) + ' km to '
                 + str(np.round(b_height, 2)) + ' km between latitudes of ' + str(southern_latitude) + ' and '
                 + str(northern_latitude) + ' ')               # setting title
    ax.set_xlabel('Day in respective year', fontsize=15)       # labeling x axis


    if os.path.exists('years_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' ') is False:

            # checking if path exists not
        os.mkdir('years_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' ')  # creating path

    for year_select in range(year_end - year_start + 1):       # looping over years
        print(year_select)                                     # print year to know progress
        if year_select > 0:                                    # check if year not start year
            plt.gca().get_lines()[year_select - 1].set_color("black")   # putting last selected year back to black
            plt.gca().get_lines()[year_select - 1].set_zorder(1)        # putting last selected year in background
        #if substance == 'O3':                                          # can be turned on to highlight special years
        #    plt.gca().lines[2017-year_start].set_color("lime")          # can be turned on to highlight special years
        #    plt.gca().lines[2017-year_start].set_zorder(3)              # can be turned on to highlight special years
        plt.gca().lines[2017-year_start].set_color("lime")              # making 2017 as selected year lime in plot
        plt.gca().lines[2017-year_start].set_zorder(3)                  # putting it into foreground
        plt.gca().lines[year_select].set_color("red")                   # making selected year red in plot
        plt.gca().lines[year_select].set_zorder(4)                      # putting it into topmost foreground
        plt.legend()                                                    # creating legend

        if os.path.exists('years_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' ' + '/'
                          + str(substance) + '_from_' + str(southern_latitude) + '_to_' + str(northern_latitude)
                          + ' ') is False:                               # checking if path exists not
            os.mkdir('years_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' ' + '/'
```

```
                        + str(substance) + '_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' ')
                                                                                   # creating path
    plt.savefig('years_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' ' + '/'
                        + str(substance) + '_from_' + str(southern_latitude) + '_to_' + str(northern_latitude) + ' '
                        + '/' + str(year_select + year_start) + '_' + substance + '_' + str(southern_latitude) + '-'
                        + str(northern_latitude) + '.png', bbox_inches='tight')  # safe picture to directory

    plt.clf()            # clear plot
```

Listing 4: Program 4