Convective and Gravity Wave Momentum Transport during EUREC^{4}A

Bachelor thesis

1. Supervisor	Prof. Dr. Joakim Kjellsson
	Christian-Albrechts-University Kiel
2. Supervisor	Dr. Claudia Stephan
	Max Planck Institut for Meteorology Hamburg
Author	Katharina Schmitt
Matriculation Number	1148594
Submitted at	May 10, 2023 in Kiel

Statutory Declaration

I, Katharina Schmitt, declare on oath towards the Christian-Albrechts-University of Kiel, that I have prepared the present bachelor thesis independently and with the aid of nothing but the resources listed in the bibliography. This thesis has neither as-is nor similarly been submitted to any other university.

Kiel, May 10, 2023

Katharina Schmitt

Acknowledgment

This research would not have been possible without the help of many people. First, I would like to thank my two supervisors, Prof. Joakim Kjellsson and Dr. Claudia Stephan, for their excellent supervision, the regular discussions and their good advice. Many thanks also to Dr. Laura Köhler, for her help with all kinds of problems and to Dr. Lukas Kluft for his support in solving programming problems. I would also like to express my gratitude towards Alessandro Savazzi for the inspiring discussions.

This thesis studies the influence of gravity wave and convective momentum transport on the trade winds during the EUREC^{4}A field campaign and estimates their relative importance, respectively. For this, the ICON Large-Eddy Simulation with a horizontal resolution of 312m is analysed and the pressure flux, the sensible heat flux, the horizontal momentum flux as well as the drag are calculated. For the sensible heat flux, the results fit well with the theory as it is close to zero for gravity waves, positive for convection and negative for dissipating gravity waves. In contrast to theory, we found out that the pressure flux is still positive for clouds and only in the unstable boundary layer negative for the convective case. This might be because we only analyse shallow convection and use a 3D model instead of a 2D model. Therefore, the influence due to the advection of clouds to our sample areas by the meridional and zonal wind component cannot be neglected. Further, we analysed the influence of wind stress. While in the convective case, no tendency could be observed, in the non-convective case an increase in maximum height of positive pressure flux with increasing wind shear is visible. Also, a negative pressure flux with a higher wind shear could be found which can be possibly explained by tropopause-reflected and downward propagating gravity waves. By analysing the drag it can be seen that both, convection and gravity waves, lead to a deceleration of the mean flow. While gravity waves are responsible for the deceleration directly above the unstable boundary layer, convection is accountable for the deceleration close to the mean cloud top between 1000m and 2000m. For further research, a more high-frequency output would be needed so that the Eliassen-Palm-Theorem can be properly applied and the pressure flux in particular can be investigated.

In dieser Arbeit wird der Einfluss des Impulstransport von Schwerewellen und Konvektion auf die Passatwinde während der EUREC^{4}A Kampagne untersucht und ihre relative Bedeutung abgeschätzt. Dazu wird die ICON Large-Eddy Simulation mit einer horizontalen Auflösung von 312m analysiert und der Druckfluss, der fühlbare Wärmefluss, der horizontale Impulsfluss sowie der Impulsübertrag berechnet. Für den fühlbaren Wärmefluss stimmen die Ergebnisse gut mit der Theorie überein, da er für Schwerewellen nahe null, für Konvektion positiv und für dissipierende Schwerewellen negativ ist. Im Gegensatz zur Theorie haben wir festgestellt, dass der Druckfluss für Wolken immer noch positiv und nur in der instabilen Grenzschicht für den konvektiven Fall negativ ist. Dies könnte daran liegen, dass wir nur flache Konvektion analysieren und ein 3D-Modell anstelle eines 2D-Modells verwenden. Daher kann der Einfluss der Advektion von Wolken durch die meridionale Windkomponente in den analysierten Abschnitten nicht vernachlässigt werden. Außerdem haben wir den Einfluss der Windscherung untersucht. Während im konvektiven Fall keine Tendenz zu beobachten war, zeigte sich im nicht-konvektiven Fall eine Zunahme der maximalen Höhe des positiven Druckflusses mit zunehmender Windscherung. Auch konnte ein negativer Druckfluss bei höherer Windscherung beobachtet werden, was möglicherweise durch an der Tropopause reflektierte und sich nach unten ausbreitende Schwerewellen erklärt werden kann. Die Analyse des Impulsübertrags zeigt, dass sowohl Konvektion als auch die Schwerewellen zu einem Abbremsen der mittleren Strömung führen. Während Schwerewellen für die Abbremsung direkt über der instabilen Grenzschicht verantwortlich sind, ist die Konvektion für die Abbremsung in der Nähe der mittleren Wolkenobergrenze zwischen 1000 und 2000m verantwortlich. Für weitere Forschungen wäre eine hochfrequentere Ausgabe erforderlich, damit das Eliassen-Palm-Theorem korrekt angewendet und insbesondere der Druckfluss untersucht werden kann.

Table of Contents

List of Figures

List of Tables

Table of Acronyms

GWMT gravity wave momentum transport

CMT convective momentum transport

PBL planetary boundary layer

DALES Dutch Atmospheric Large-Eddy Simulation

ICON Icosahedral Nonhydrostatic Model

1. Introduction

The tropical trade winds are a high-priority system to understand as they cause the convergence patterns in the tropics which lead to the majority of tropical rainfall and they influence ocean currents, upwelling and sea-surface temperature (Nuijens et al. (2022)). Especially the surface wind-speed has a big impact on cloud amount, precipitation and organization on different timescales (Nuijens et al. (2022), Brueck, Nuijens, and Stevens (2015), Klein (1997)). Not only large-scales influence the trade winds, but also turbulence, convection as well as gravity waves modify the wind profile through vertical transport of horizontal momentum and hence indirectly influence the transport of heat and moisture, which has a great impact on the general circulation (Nuijens et al. (2022)). Wave stress and especially the propagation of wave momentum and energy away from the planetary boundary layer (PBL) plays an important role in linking small scale phenomena to meso- and large-scale ones and is therefore an essential component of the global momentum balance of the atmosphere and the general circulation (Nappo (2014), Lalas and Einaudi (1976)). Since systematic forecast errors in models cause big wind biases in the lower atmosphere of the trades, there is a potential for model improvement. According to Savazzi et al. (2022), models predict lower wind speed in the trades than observed (Fig 1.1). It is still an open question which processes cause these biases. Additionally, Shaw and Lane (2013) as well as Lane and Moncrieff (2010) describe the need for further model improvement in this aspect as the current parametrizations of convective momentum transport (CMT) and gravity wave momentum transport (GWMT) in global circulation models are independent in contrast to the theoretically known connection between the processes.
The EUREC^{4}A field campaign, a campaign which was designed to study the interplay between clouds, convection and circulation and their role in climate change, took place between 20th January and 20th February 2020 (Stevens et al. (2021)) near Barbados (see Fig. 1.1) and offers a

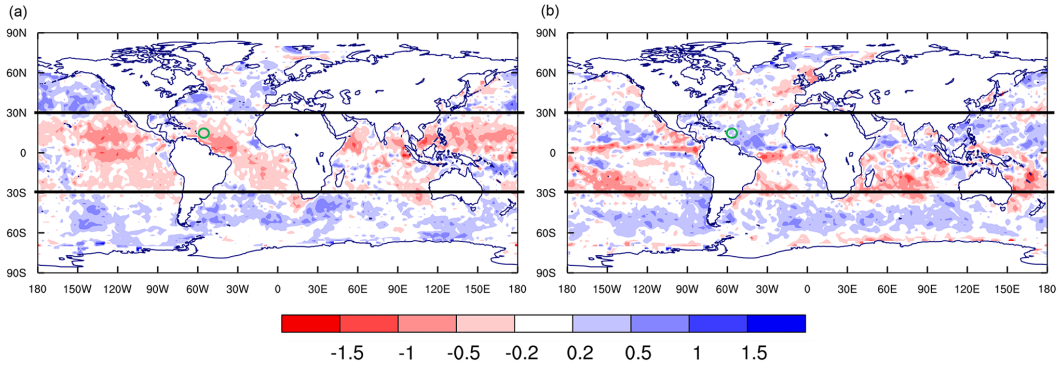

Figure 1.1.: Surface wind biases in operational deterministic forecasts produced with the Integrated Forecasting System of the European Center for Medium-Range Weather Forecasts in January and February. Panel a) refers to the zonal and panel b) to the meridional wind components (Savazzi et al. (2022)). Units are in ms^{-1}. The green circle marks the study area of EUREC^{4}A.

great amount of observation and model data in the area to analyse which processes matter most for the momentum transport. The observed or total momentum budget (Fig. 1.2) can be described by

$$\left(\frac{\partial \bar{u}}{\partial t}\right) = \left(\frac{\partial \bar{u}}{\partial t}\right)_{large\ scale} + \left(\frac{\partial \bar{u}}{\partial t}\right)_{small\ scale} \tag{1.1}$$

where the large-scale momentum tendency is due to advection, coriolis and pressure gradient force while the small-scale momentum tendency refers to CMT, GWMT and turbulence. As Nuijens et al. (2022) could not rule out that GWMT is playing a role in the observed momentum budget, this thesis tries to disentangle the signals of GWMT and CMT and to estimate the relative importance of GWMT versus CMT for the small-scale momentum budget during the EUREC^{4}A field campaign.

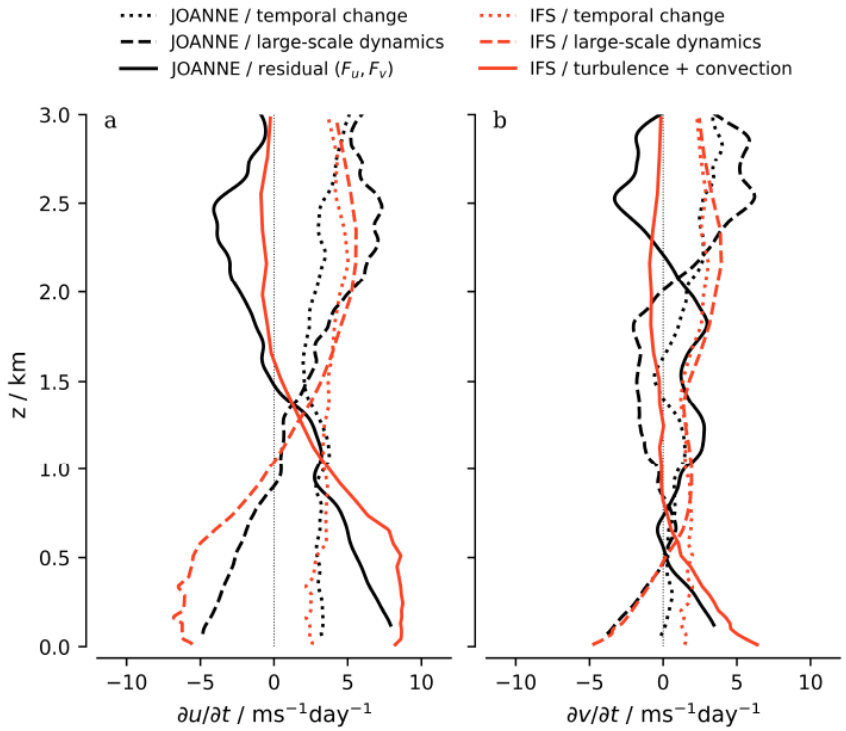

Figure 1.2.: Overview over observed and simulated by the IFS momentum fluxes during the EUREC^{4}A campaign (Nuijens et al. (2022)).

1.1. Introduction to the Linear Theory of Gravity Waves and Gravity Wave Momentum Transport

Gravity waves are waves generated in a stably stratified fluid or at the interface between two fluids of different density when the buoyancy or gravity force, respectively, tries to restore equilibrium. Gravity waves can be generated by vertical displacement of air parcels for example due to terrain or convection (Nappo (2014)). For shallow convection, the most common mechanism of gravity wave generation is the so-called 'obstacle effect', where penetrating thermals in a shear layer act as obstacles to the flow and generate waves with an apparent upstream dominance (Nappo (2014), Lane and Clark (2002)). The following derivation of the most important equations and relationships for gravity waves are mostly based on Nappo (2014). The Euler equations under the assumption of

two-dimensional flow without rotation, heat conduction and friction and with use of the linearization

$$q(x,z,t) = q_0(z) + q_1(x,z,t) \tag{1.2}$$

can be denoted as

$$\frac{\partial u_1}{\partial t} + u_0\frac{\partial u_1}{\partial x} + w_1\frac{du_0}{dz} = -\frac{1}{\rho_0}\frac{\partial p_1}{\partial x} \tag{1.3}$$

$$\frac{\partial w_1}{\partial t} + u_0\frac{\partial w_1}{\partial x} = -\frac{1}{\rho_0}\frac{\partial p_1}{\partial z} - \frac{\rho_1}{\rho_0}g \tag{1.4}$$

$$\frac{\partial u_1}{\partial x} + \frac{\partial w_1}{\partial z} = 0 \tag{1.5}$$

$$\frac{\partial \rho_1}{\partial t} + u_0\frac{\partial \rho_1}{\partial x} + w_1\frac{d\rho_0}{dz} = 0 \tag{1.6}$$

where u, w are the wind components in x- and z-direction, p the pressure, ρ the density and g the gravitational acceleration. The intrinsic frequency which is the frequency of a wave in the reference frame of the mean flow is

$$\Omega = \omega - u_0 k \tag{1.7}$$

with ground-based wave frequency ω, mean flow velocity u_0 and wavenumber k. Using equation (1.7), the equation for the Brunt-Väisälä-Frequency

$$N^2 = -\frac{g}{\rho_0}\frac{\partial \rho_0}{\partial z} \tag{1.8}$$

and wave-like solutions of the form

$$q_1(x,z,t) = \tilde{q}(z)\exp(i(kx - \omega t)) \tag{1.9}$$

the equations 1.3 - 1.6 become the polarization equations

$$i\Omega\tilde{u} - \tilde{\omega}\frac{du_0}{dz} = \frac{i}{\rho_0}k\tilde{p} \tag{1.10}$$

$$i\Omega\tilde{w} = \frac{1}{\rho_0}\frac{d\tilde{p}}{dz} + \frac{\tilde{\rho}}{\rho_0}g \tag{1.11}$$

$$ik\tilde{u} + \frac{d\tilde{w}}{dz} = 0 \tag{1.12}$$

$$i\Omega\tilde{\rho} - \tilde{w}\frac{\rho_0}{g}N^2 = 0. \tag{1.13}$$

We can use these equations to derive how the phases of the other variables relate to that of $\tilde{w}$. First, we assume a solution of

$$\tilde{w}(z) = Ae^{imz} = A(\cos(mz) + i\sin(mz)) \tag{1.14}$$

with the vertical wave number m. Inserting equation (1.14) in equation (1.12), the real part of the u can be determined as

$$\tilde{u}_R = -A\frac{m}{k}\cos(mz). \tag{1.15}$$

One can see that $\tilde{u_R}$ is either in phase or 180° out of phase with $\tilde{w}$, depending on the sign of m. Reshaping equation 1.10 as above, it is visible that $\tilde{p}$ is in phase with $\tilde{u}$. Therefore, $\tilde{p}$ shows the same relative phase properties towards $\tilde{w}$ as $\tilde{u}$. Dividing equation 1.13 into real and imaginary part, the real part of $\tilde{\rho}$ is

$$\tilde{\rho}_R = \frac{A\rho_0 N^2}{g\Omega}\sin(mz). \tag{1.16}$$

As the prefactor is always positive, $\tilde{\rho}_R$ is always 90° out of phase with $\tilde{w}_R$. Because the potential temperature θ is related to ρ with

$$\frac{\rho_1}{\rho_0} = -\frac{\theta_1}{\theta_0} \tag{1.17}$$

there cannot be a net heat transport by a linear gravity wave. Solving equations (1.10) - (1.13) for $\tilde{w}$ and using the scaling height $H_S = \frac{RT}{g}$, where R is the universal gas constant and T the temperature a new variable is defined as

$$\hat{w} = \tilde{w}e^{-z/(2H_S)}. \tag{1.18}$$

Differentiating equation (1.18) twice, we receive the Taylor-Goldstein-Equation

$$\hat{w}'' + \left[\frac{N^2}{(c-u_0)^2} + \frac{u_0''}{(c-u_0)} - \frac{1}{H_S}\frac{u_0'}{(c-u_0)} - \frac{1}{4H_S^2} - k^2\right]\hat{w} = 0 \tag{1.19}$$

with c as the phase speed of the wave. By replacing the factor in brackets with m^2, we get

$$\hat{w}'' + m^2\hat{w} = 0. \tag{1.20}$$

With a constant m, $\hat{w}$ is

$$\hat{w} = Ae^{imz} + Be^{-imz} \tag{1.21}$$

where A and B are constant amplitudes. With a real m , the gravity wave, which is then called internal or propagating, transports energy vertically. In contrast, gravity waves with a complex m, also known as external or evanescent gravity waves, do not transport energy vertically. A vertically moving wave is associated with vertical flux of horizontal momentum or wave stress

$$\tau(z) = -\rho\overline{u_1 w_1} \tag{1.22}$$

where the horizontal bar denotes an average over at least one full wavelength. Equation 1.22 can be calculated as

$$\overline{u_1 w_1} = \frac{1}{2}\Re(\mathscr{F}(u_1)\mathscr{F}(w_1)^*). \tag{1.23}$$

Through dissipation, reflection or breaking the momentum transported by waves can be transferred to the mean flow in form of a drag, the wave drag

$$F_d = \int_k \frac{1}{\rho_0}\frac{d}{dz}(\rho\overline{u'w'}). \tag{1.24}$$

Multiplying the linearized equation for horizontal momentum (Eq. 1.3) with wave-like vertical velocity perturbations $w' = w_1 \exp[i(kx - wt)]$ and taking the real parts, we get

$$(\omega - u_0 k)\, u_1 w_1 = \frac{k}{\rho_0} w_1 p_1. \tag{1.25}$$

Through integration and use of equation (1.7), the equation reshapes to

$$\rho_0\overline{u_1 w_1} = \frac{k}{\Omega}\overline{w_1 p_1}. \tag{1.26}$$

To rearrange equation (1.26) we use the equation for the vertical group velocity

$$w_g = -\frac{Nmk}{(k^2+m^2)^{\frac{3}{2}}} \tag{1.27}$$

and for the wave energy averaged over one wave length

$$\bar{E} = \frac{1}{2}\rho_0\overline{(u_1^2+v_1^2+w_1^2)} + \frac{1}{2}\rho_0 N^2\overline{\zeta_1^2} \tag{1.28}$$

where ζ_1 is the vertical displacement of an air parcel from its equilibrium position. The energy flux density vector which describes the flux of wave energy across the surface bounding a volume $\vec{V_1} = u_1\hat{x} + v_1\hat{z}$ is defined as

$$\vec{F} = \overline{p_1\vec{V_1}} = \bar{E}(u_g\hat{x} + w_g\hat{z}) \tag{1.29}$$

with the horizontal group velocity u_g. Using equations (1.27) - (1.29), equation (1.26) can then be reformulated to the Eliassen-Palm-Theorem

$$\overline{w_1 p_1} = w_{\mathrm{g}}\bar{E} = (u-c)\overline{u_1 w_1}. \tag{1.30}$$

These properties of waves lead Shaw and Lane (2013) to develop criteria to distinguish between CMT and GWMT. According to equation (1.16) and (1.30), pressure flux as well as sensible heat flux can be used to separate the two different momentum transports. Table 1.1 shows an overview of he corresponding signs in no background wind conditions. In equation (1.30) the correlation between vertical wave velocity and pressure flux can be seen. The pressure flux is positive for upward propagating waves and negative or zero for downward propagating gravity waves or convection. The vertical sensible heat flux is zero for gravity waves (see equations (1.16) and (1.17)). For unstable moist convection it is expected to be positive. The remaining contribution of negative sensible heat flux is consistent with dissipating gravity waves. Table 1.1 is valid for no shear conditions. According to Shaw and Lane (2013), with (stronger) wind shear the sensible heat flux cannot be used as a criterion to distinguish between gravity waves and convection as a flux can be simply generated by buoyancy and shear production

$$\theta' w' \approx \frac{\theta}{g}\frac{dU}{dz}u'w'. \tag{1.31}$$

where U is the background wind speed. It can still be used as a criterion for days with lower wind shear. In contrast, the pressure flux is still reliable in strong wind shear cases. Overall, Shaw and Lane (2013) conclude, that the non-upward contribution is dominated by CMT and not by downward-propagating gravity waves. The upward contribution is dominated by upward-propagating gravity waves.

	propagation direction	pressure flux $\overline{p'w'}$	sensible heat flux $\overline{T'w'}$
gravity waves	upward	> 0	0
	downward	≤ 0	0
	dissipating	–	< 0
convection	–	≤ 0	> 0

Table 1.1.: Analysis criteria for separation of convective momentum transport (CMT) and gravity wave momentum transport (GWMT) according to Shaw and Lane (2013).

1.2. Processes of propagating waves

Now the question appears what happens to the transported momentum and under which conditions can it be transferred to the mean flow in form of drag? The following enumeration gives an overview of processes happening to waves during propagation and explains the requirements for each of them:

1. **Dissipation:** The height z_c where the horizontal phase speed of a wave propagating in the same direction as the background wind is equal to the wind speed, $c - u_0(z_c) = 0$, is called the critical level. A vertically propagating wave which approaches its critical level will be diffused or dissipated (Nappo (2014), Fritts (1982)) and (when approaching from below) not be able to propagate above the height of the critical level (Nappo (2014), Lane and Clark (2002)).

2. **Trapping and Reflection:** A vertically propagating gravity wave will become trapped when its intrinsic frequency exceeds the local

Brunt–Väisäla-frequency N_{local}. This is important for gravity waves propagating in the opposite direction to the sheared flow, when flow speed becomes too large. Especially gravity waves with frequencies close to N are trapped easily due to changes in wind speed. In this case the phase lines are orientated vertically and the amplitude decays exponentially outside the deducting layer. Trapping is a filter for waves as it decides which waves propagate vertically and which ones get reflected. For trapped waves the vertical momentum flux is zero as the upward and downward flux are of opposite sign and cancel each other out (Fig. 1.3). Instead of trapped, a wave may be reflected and propagate downwards (Nappo (2014), Lane and Clark (2002)).

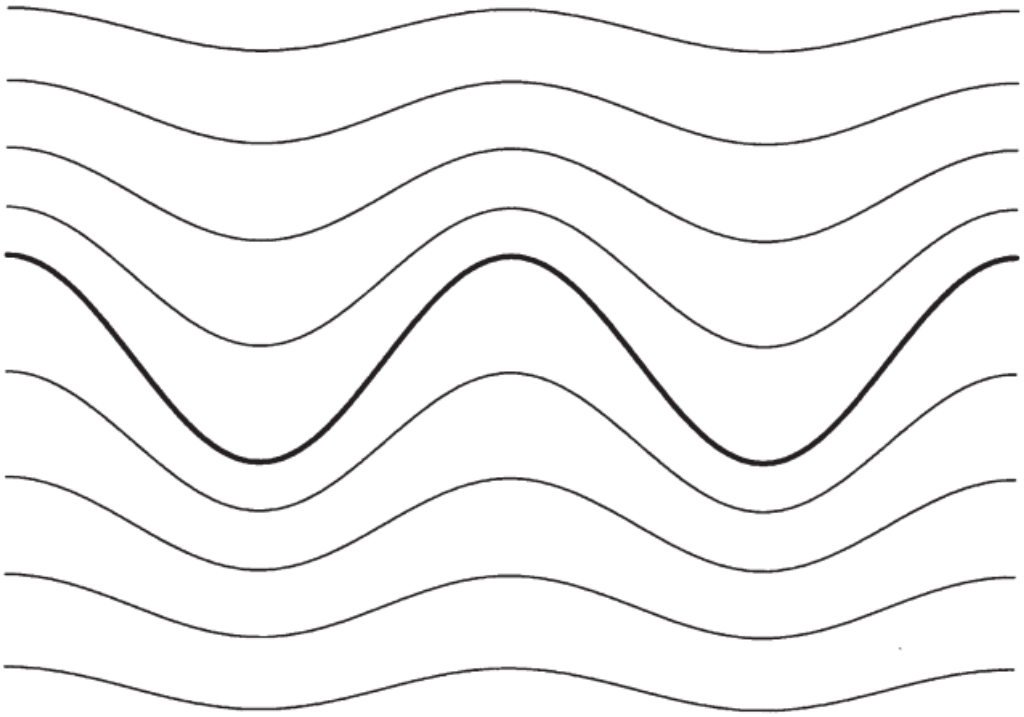

Figure 1.3.: Example of a trapped wave with exponential declining amplitude (Nappo (2014)).

Due to these processes only a narrow part of the spectrum of gravity waves is able to propagate (Fig. 1.4).

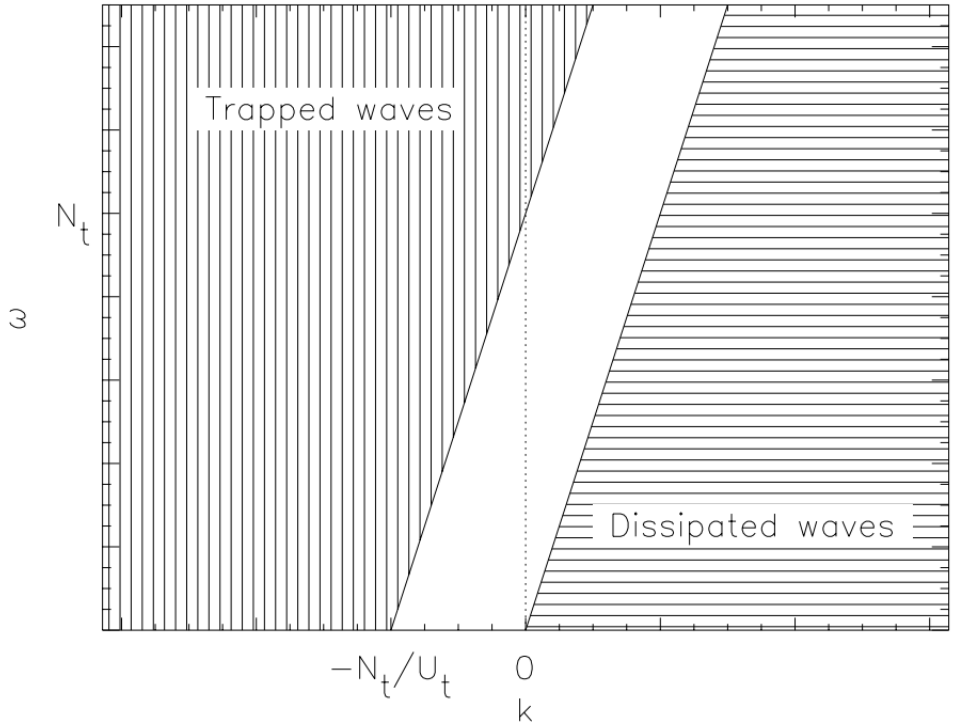

Figure 1.4.: Schematic representation of the influence of a (positive) mean wind U_t on the spectrum of vertically propagating gravity waves. The vertical lines show the conditions under which gravity waves are trapped, the horizontal ones the gravity waves which are dissipated. The empty area in the middle shows the conditions for which the waves are able to propagate vertically (Lane and Clark (2002)).

1.3. Main research questions

Since we cannot use the criterion introduced by Shaw and Lane (2013) based on equations (1.30) and (1.16) as the temporal resolution of our model data is not sufficient, we sub-sample the data into different categories namely "convection" and "no-convection". This thesis first answers the question if this method is sufficient to detect a signal in terms of the different fluxes, second if the answer depends on the background

wind shear and third if based on the results one may draw any conclusion about relative importance of CMT versus GWMT in the study area. Chapter 2 gives an overview of the used data and methods, Chapter 3 displays and discusses the results which are concluded in Chapter 4.

2. Introduction to Models and Methods

In contrast to previous studies (Shaw and Lane (2013), Lane and Clark (2002)) the model output from the control Icosahedral Nonhydrostatic Model (ICON) Large-Eddy Simulation (LES) gathered in the EUREC^{4}A field campaign and which is made available through Schulz and Stevens (2023) is more realistic as it is a nested model in the storm-resolving ICON model and has for example no an-elastic assumption. Shaw and Lane (2013), as well as Lane and Clark (2002) focused on two-dimensional model output. Previous studies showed that the wind shear in u-direction is more relevant for the study area (Stephan, Žagar, and Shepherd (2021)) and since waves propagate in the direction of the wind shear, this thesis focuses on the x - direction and therefore only on the u and w component of the wind. The used simulation is the control ICON LES with a cloud condensation nuclei concentration of 130 cm^{-1}. Figure 2.1 shows the location and size of the model area. The 3D output has a horizontal resolution of 312m and 68 vertical levels with a distance between the levels from 30m at the surface up to 120m at higher levels. The maximum investigation height is 5.6km, so only the boundary layer and lower troposphere are analysed. The temporal resolution is three hours. The differences in the models can already be seen in the comparison of the distribution of vertical velocity w in Figure 2.2 and Figure 2.3. To handle and analyse the data on a regular latitudes-longitudes grid instead of the ICON grid, python code written by Koelling and Schulz (2022) is used to calculate latitudes and longitudes to which the data is interpolated. As mentioned in Section 1.3, the criteria to subsample the data in convective and non-convective categories are the following:

1. convective: at least 20 percent of cells have somewhere in the column a specific cloud water content $qc > 0.0$
2. non-convective: amount of clouds ($qc > 0.0$) is zero in the whole area

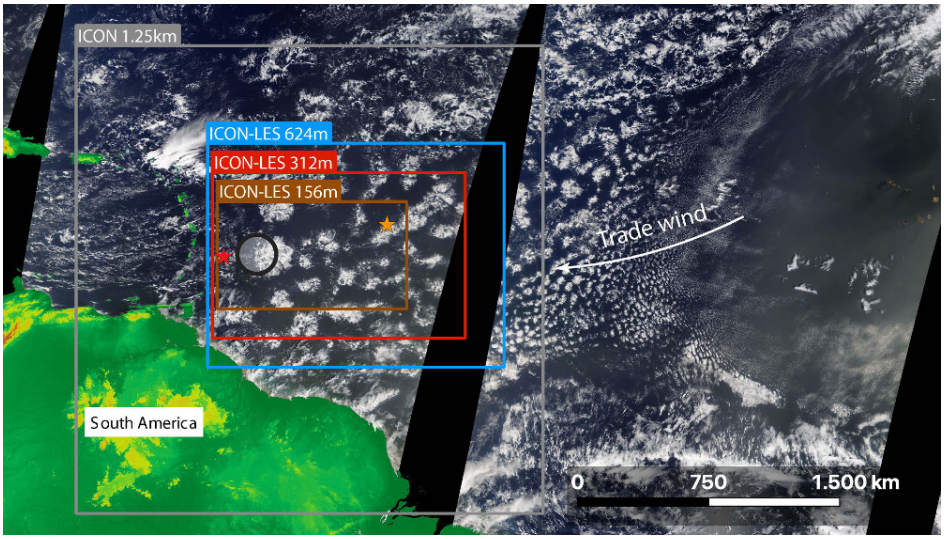

Figure 2.1.: The red box shows the borders of the simulated domain. The location of the Barbados Cloud Observatory (BCO) is marked with a red star at the western part of the domain (Schulz (2021)).

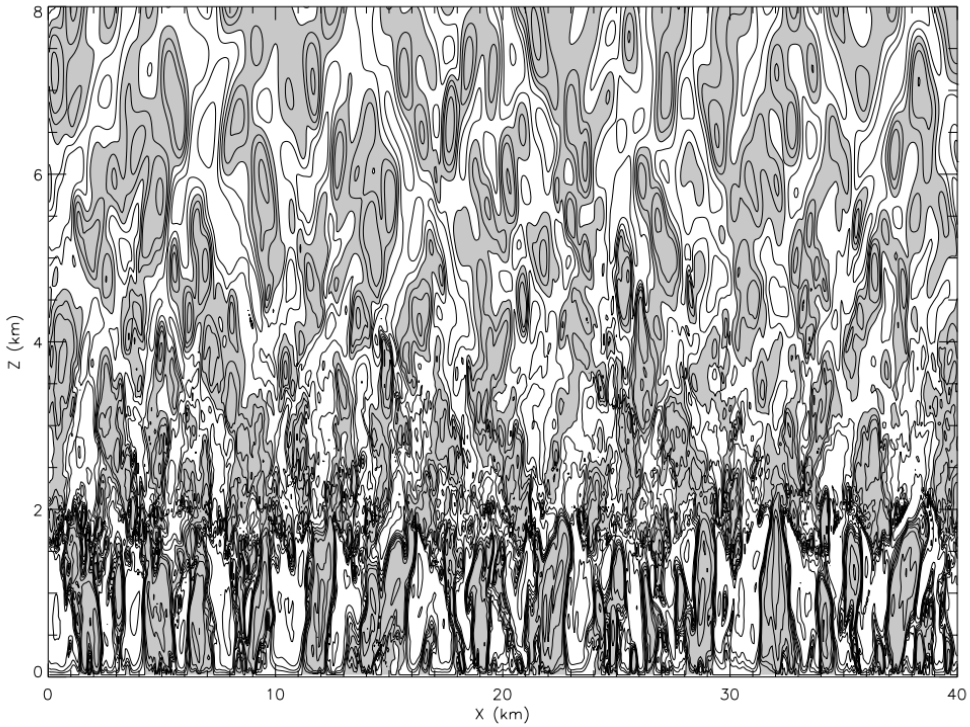

Figure 2.2.: Displaying of vertical velocity w for the model analysed by Lane and Clark (2002).

Afterwards for each time step and for every fifth latitude a section of $40km$ is detected where the aforementioned criterion is fulfilled. In total, 1523 latitudes are calculated for every time step. The data is then

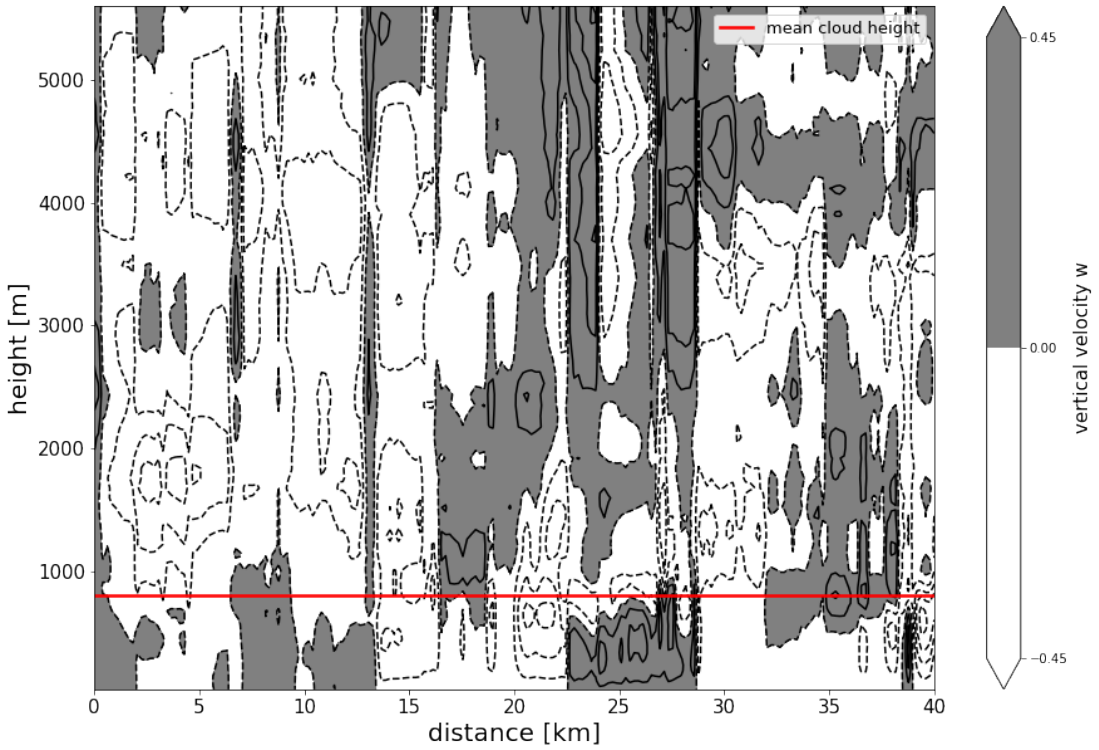

Figure 2.3.: Same quantities as in Figure 2.1 but for the ICON-LES-312m simulation.

filtered to wavelengths below 5km with a high-pass filter and between 5 and 10km with a band pass filter. The maximum wavelength is 20km (due to the size of the selected area) and therefore synoptic scale influence can be neglected. Afterwards, we apply the 1-D discrete Fourier Transform to calculate the co-spectrum of p and w, T and w and u and w according equation (1.22) for all variables. The wave drag is calculated with help of equation (1.24). Additionally, the mean cloud top is calculated for every latitude. For more details of the programmed code see Appendix A. Furthermore, the results are compared to the Dutch Atmospheric Large-Eddy Simulation (DALES). DALES is a special model to resolve turbulent scales, it resolves in normal setups more than 90 % of the turbulent kinetic energy (Heus et al. (2010)). It covers an area of 150km x 150km and has a horizontal resolution of 100m. The 3D output has a vertical resolution of 123 levels with a vertical distance of 20m close to the ground and 85m higher up. The temporal resolution is 30 minutes. In contrast to ICON, DALES is not a nested model.

3. Results and Discussion

3.1. Wind profiles

Figure 3.1 shows the wind profile averaged over the whole domain for all simulated days. Up to a height of 2000m, there are always easterly winds, above that the wind can be from western directions too. The mean profile for all days shows only above 5000m westerly winds. On 07.02.2020, the day for further analysis (Fig. 3.2) in the upcoming sections, the wind has mainly eastern directions.

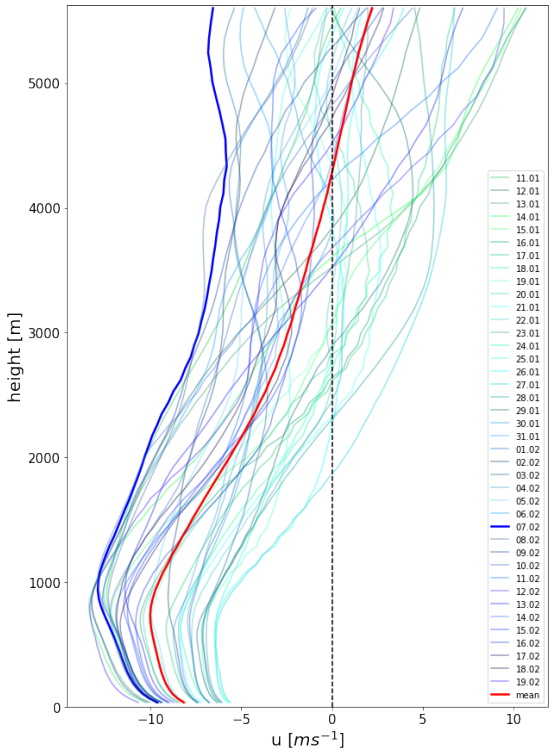

Figure 3.1.: Mean wind profiles of the whole domain for all simulated days. The red line shows the total averaged wind profile and the thicker blue line the wind profile of the 07.02.2020.

3.2. Pressure flux, sensible heat flux and drag

Figure 3.2 shows exemplarily the analysed criteria, which are the pressure flux, the sensible heat flux, the vertical flux of horizontal momentum

and the drag, for 07.02.2020. The wind profile (a) shows a day with easterly winds over the whole profile and a maximum wind speed of $u = -12.5\ ms^{-1}$ at 1000m and above a decrease to the minimum of -7.5 ms^{-1}. For the convective region, the pressure flux (b) is positive with a maximum magnitude of $\overline{p'w'} = 13\ Pa \cdot ms^{-2}$ at the mean cloud top height. Above the cloud top, the pressure flux decreases with a minimum of around 2000m. Higher up, a second and third positive peak but with smaller magnitude are visible. The sensible heat flux (c) shows positive values close to the surface while it is negative between 200 and 600m. Afterwards, a strong positive flux up to a height of 4000m with a maximum magnitude of $\overline{T'w'} = 1.2\ K \cdot ms^{-1}$ is observed.

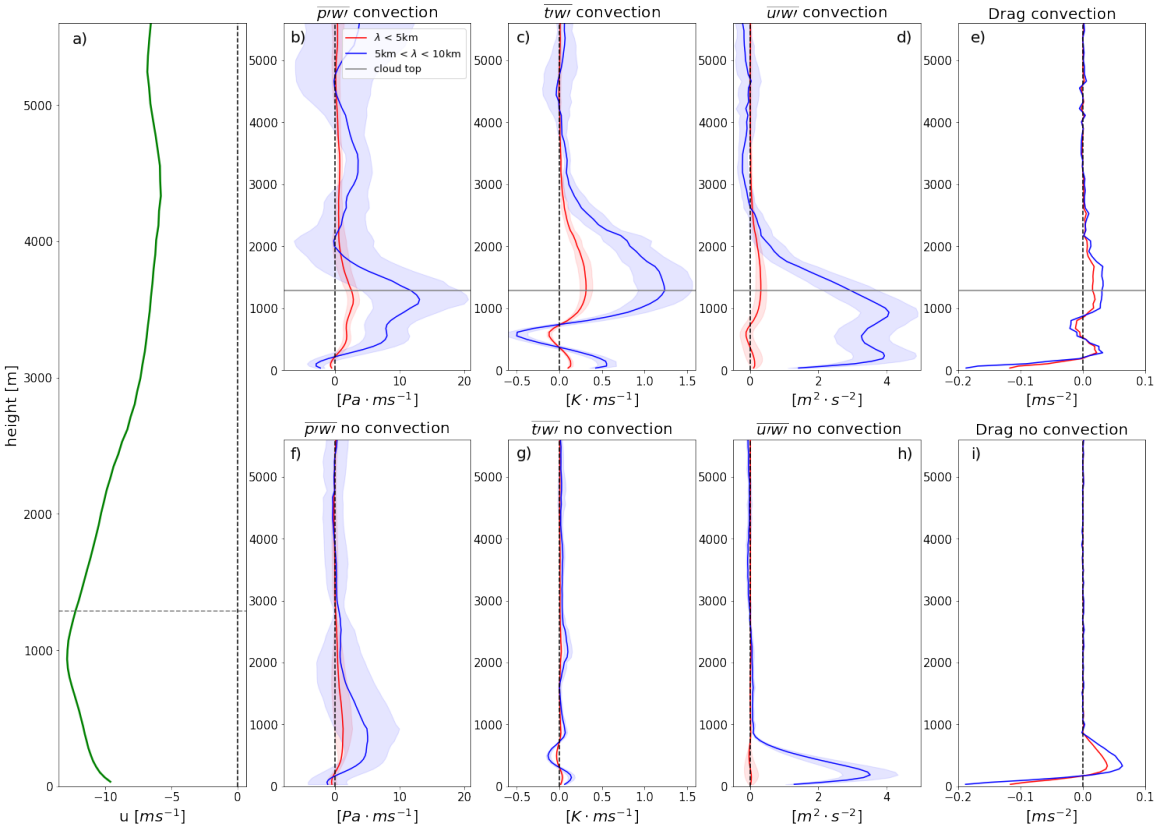

Figure 3.2.: a) Wind profile for u, b) pressure flux $\overline{p'w'}$, c) sensible heat flux $\overline{T'w'}$, d) momentum flux $\overline{u'w'}$ and e) drag for the convective case. Same quantities for the non-convective case in f), g), h) and i) for 07.02.2020. Red colour shows wavelengths λ < 5km, blue for 5km < λ <10km.

The corresponding drag (e) is negative close to the ground and positive within the mean cloud top, which is equivalent to a deceleration of the

mean flow there. In contrast, the non-convective pressure flux (f) has only one peak at 1000m with a magnitude of 5 $Pa \cdot ms^{-2}$. The sensible heat flux (g) is over the whole profile close to zero. The associated drag is negative at the bottom, up to a height of 1000m slightly positive and approximately zero above. The sensible heat flux fits in both cases the expectations, a strong positive flux in the cloud layer where convection happens and a flux close to zero in the non-convective region where gravity waves dominate. The pressure flux on the other hand corresponds well with the expectation for the non-convective region while it differs for the convective one. Across all days and profiles a mainly positive pressure flux in the layer close to the mean cloud top can be observed even though it should be negative according to Shaw and Lane (2013). To analyse this further, $\overline{p'w'}$ is calculated differently for a latitude (Fig. 3.3). Instead of the fast-Fourier-transform, a running mean of 5km is used where $x' = x - \bar{x}_{5km}$.

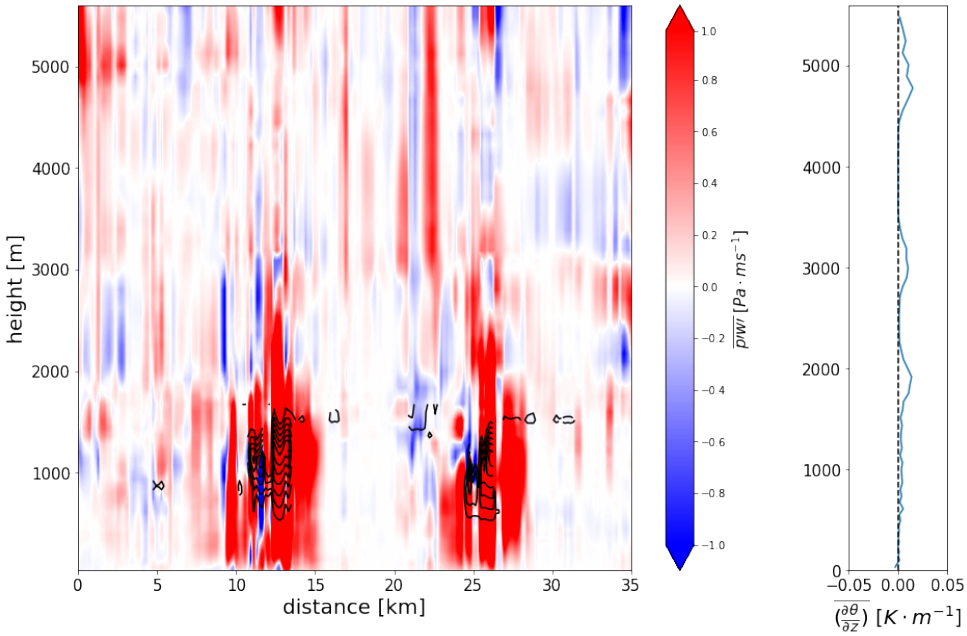

Figure 3.3.: Pressure flux (left) and stability (right) for one section. Black lines indicate a $qc > 0.0$ and therefore clouds. Positive pressure flux is shown in red, negative one in blue.

One can see that the pressure flux is positive even in clouds. This seems surprising at first but considering the stability profile for the columns with

a $qc > 0.0$ on the right side we see that the change with height of the potential temperature θ is close to zero and hence stable. According to this, the clouds visible in the section might be formed outside the section area and the pressure flux is negative only in the unstable boundary layer.

3.3. Dependence on wind stress

As Shaw and Lane (2013) set up the theoretical disentanglement only for conditions without wind stress, we analyse the dependence of the pressure flux on wind shear between $100m$ and $400m$. For the convective case (Fig. 3.4) there is no tendency visible. Close to the ground, the pressure flux is always negative with magnitudes around $-0.45\ Pa{\cdot}ms^{-1}$. This is consistent with unstable convection close to the ground. Above the unstable boundary layer the pressure flux is positive with magnitudes up to $0.75\ Pa \cdot ms^{-1}$.

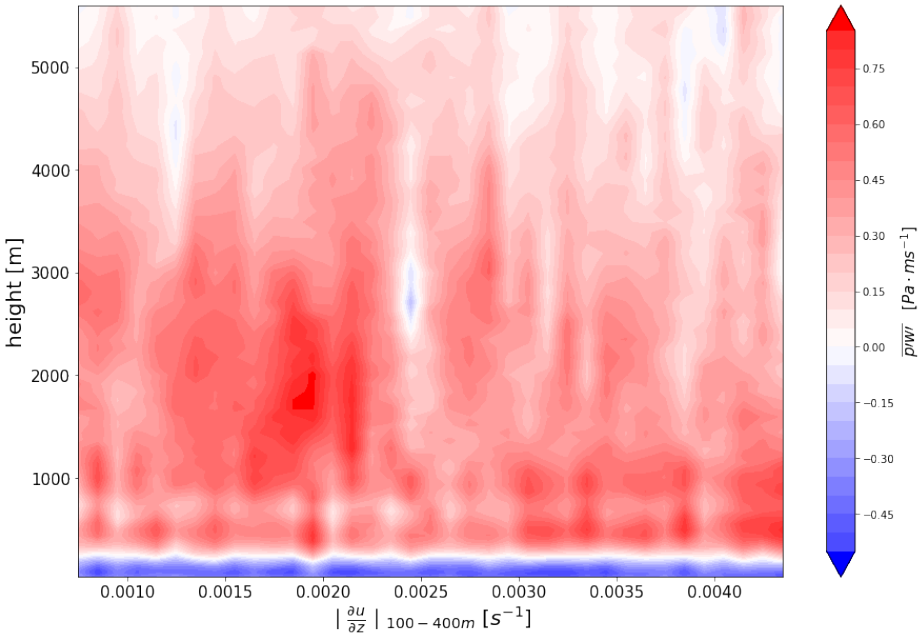

Figure 3.4.: Dependence of pressure flux in the convective case on wind stress between 100 and 400m.

In the non-convective case (Fig. 3.5), the negative pressure flux in the boundary layer is still visible. But in contrast to the convective case,

one can see an influence of wind shear. First, the height levels directly above the unstable boundary layer show an increase in height of positive pressure fluxes with increasing wind shear. This is consistent with gravity waves initiated by the dry boundary layer thermals due to the obstacle effect and can be explained by a stronger effect with stronger wind shear. Above roughly 1000m and up to a wind shear of $0.0020s^{-1}$, positive pressure flux is visible while for wind shear greater than $0.0020s^{-1}$ the pressure flux becomes negative. As this feature is also visible for the equivalent analysis with wind shear between 1 and 2km (not shown), this could be downward propagating waves which have been reflected at the tropopause. Supporting this theory is that trapping and reflection become more likely with higher wind shear.

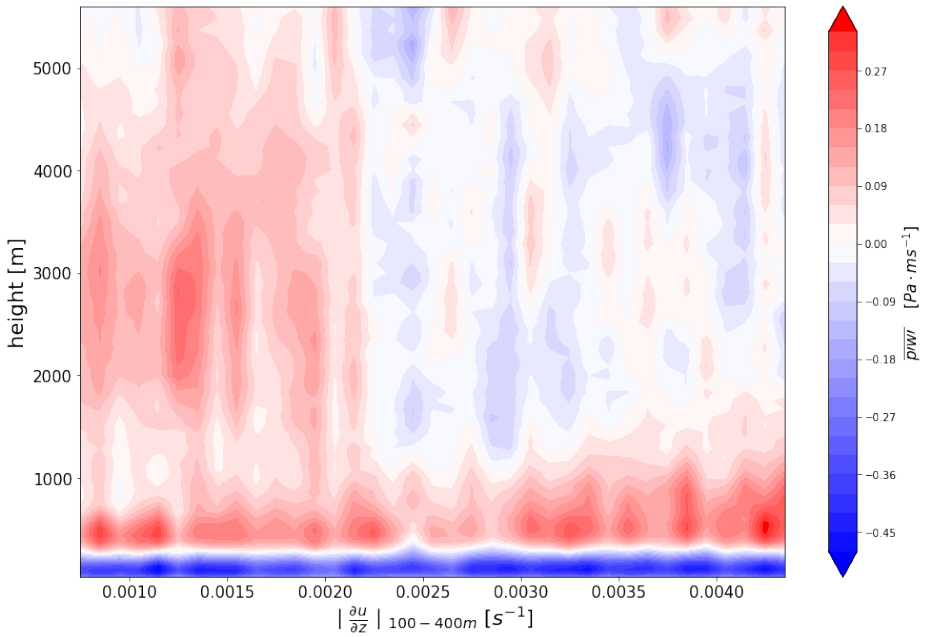

Figure 3.5.: Dependence of pressure flux in the non-convective case on wind stress between 100 and 400m.

3.4. Clustering with k-means

To analyse if convective and non-convective profiles can be separated from each other, the k-means algorithm is applied to all profiles to clus-

ter them into k specific groups whose profiles are most similar to each other. The number k of different clusters is chosen by the optimum number of clusters to divide the profiles into convective and non-convective ones. As the sensible heat flux is the best criterion to distinguish between the groups (not shown), it is chosen for the clustering (Fig. 3.6). All of the non-convective cases are sorted into group number 1 and only

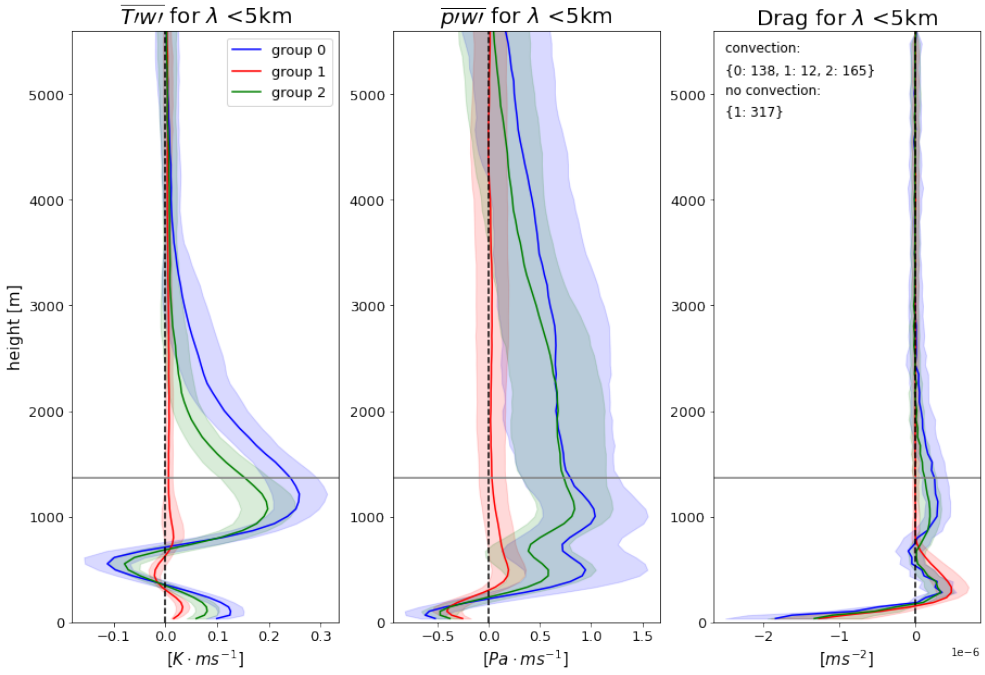

Figure 3.6.: Clustered a) sensible flux, b) pressure flux and c) drag. Number of profiles in a specific cluster are given in the figure. Shaded area shows the standard deviation. The grey line shows the mean cloud top for the convective cases.

12 profiles of convective cases are falsely sorted into the same group. The shape of the non-convective profiles agrees with theory as the flux is zero for gravity waves. The convective cases are sorted into two different groups (group 0 and group 2) with slightly different profiles. Hence, 100% of the non-convective profiles are in the correct groups as well as 95% of the convective profiles. Applying the same cluster groups to the pressure flux a difference between non-convective and convective pressure flux can be seen. While the non-convective pressure flux has a lower magnitude and only one maximum below 1000m, the pressure

fluxes for the convective case show higher magnitudes and two peaks below 1500m. The first peak corresponds in height well with the peak in the non-convective case but has a different magnitude. The sensible heat flux is slightly negative right above the corresponding height for both cases. The related drag shows a positive sign and is greater for the non-convective case than for the convective one. As the wind is in an easterly direction, a positive sign in drag indicates a deceleration. Combining all the aspects for this specific height, it can be explained by gravity waves which are formed due to the obstacle effect right above the unstable boundary layer and dissipate (sensible heat flux is negative), leading to a deceleration of the mean flow. The second peak in the height of the mean cloud top layer corresponds well with the mean cloud top height and a strong positive sensible heat flux in the convective case while it is still close to zero in the non-convective case. The drag for the convective groups is positive in the observed height while it is zero for the non-convective case. This leads to the conclusion that convection acts as an obstacle to the flow and is therefore a source for gravity waves which lead to a positive pressure flux. Additionally, convection is responsible for a deceleration of the mean flow at a height between 1 and 2km.

3.5. Comparison with DALES

To estimate sensitivity to model resolution and formulation and to understand to what extent our model represents reality, we now compare ICON with DALES, which is another 3D model. Comparing the vertical velocity w of DALES (Fig. 3.7) with the one from ICON (Fig. 2.3) one can see that DALES shows stronger and finely organized thermals. The mean cloud height corresponds well with the top of the boundary layer thermals. It shows more similarities with the 2D model output from Lane and Clark (2002) (Fig. 2.2) than ICON does. We also see that the mean cloud top is much higher than in the ICON simulation.

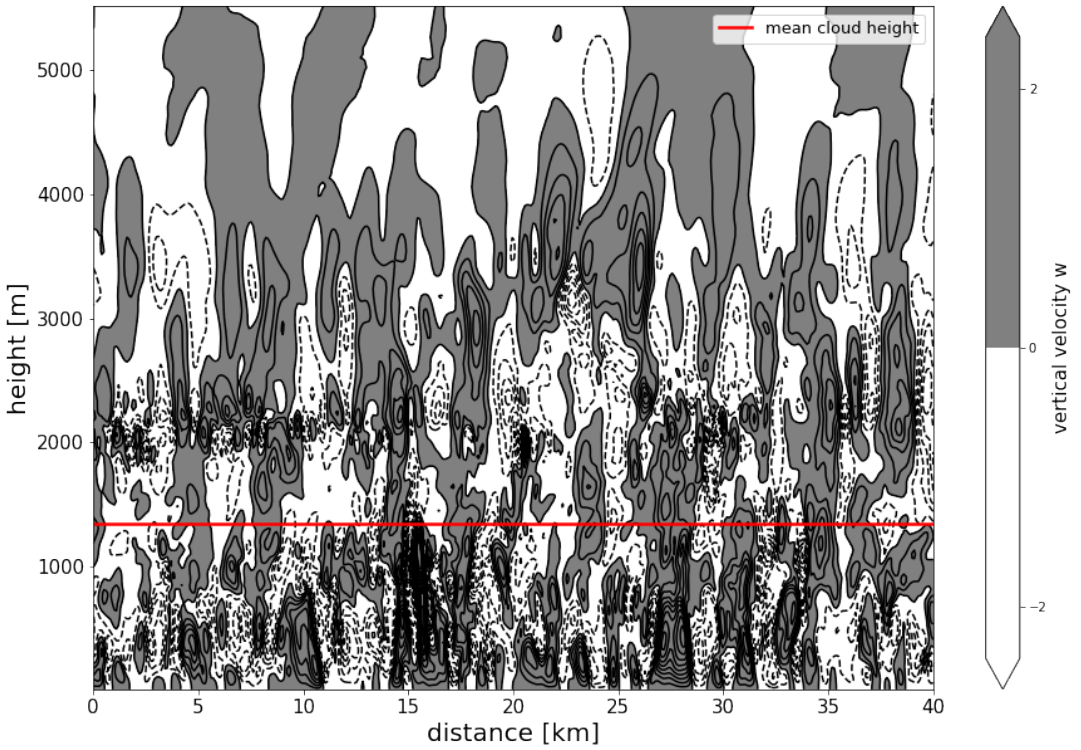

Figure 3.7.: Displaying of vertical velocity w for the DALES simulation.

One can see that higher resolution helps to realize the small convective structures which are there and missed out by the ICON simulation. Due to the implemented anelasticity, a comparison of the pressure flux was not possible.

4. Conclusion

The EUREC^{4}A field campaign with its great amount of observational and model data made it possible to analyse the small-scale momentum transport in the trades. We were able to disentangle GWMT and CMT based on theory as devised by Shaw and Lane (2013) in a more realistic 3D model output. We could confirm that the sensible heat flux is close to zero for gravity waves, positive for convection and negative for dissipating gravity waves. We see that theory fits well for this aspect to disentangle the signal between convection and non-convection, even though it might not be accurate as we have wind shear in our analysed profiles. In contrast to theory, we found out that the pressure flux is still positive for clouds and only in the unstable boundary layer negative for the convective case. This might be because we only analyse shallow convection in contrast to Shaw and Lane (2013) which focussed on deep convection. Additionally, we used a 3D model instead of a 2D model and even though we ignore the meridional wind component for our analysis we cannot neglect that there could be an influence due to the advection of clouds to our sample areas (away of their forming zones). For further research, a higher-frequency output would be needed so that the Eliassen-Palm-Theorem can be properly applied. Further, we analysed the influence of wind stress. While in the convective case, no tendency could be observed, in the non-convective case there is an increase in maximum height of positive pressure flux with increasing wind shear. Also, a negative pressure flux above 1000m for a wind shear above 0.00020s^{-1} could be observed which can be possibly explained by the tropopause-reflected and downward propagating gravity waves. By analysing the drag it can be seen that both, convection and gravity waves, lead to a deceleration of the mean flow. While gravity waves are responsible for the deceleration directly above the unstable boundary layer, convection is accountable for the deceleration between 1000m and 2000m. Applying the k-means algorithm to all profiles we found out

the that the sensible heat flux is a good criterion to distinguish between convection and no-convection. Overall, we propose further analysis with higher-frequency and higher horizontal resolution data to investigate the finer structures of convection and the pressure flux in particular.

References

Brueck, Matthias, Louise Nuijens, and Bjorn Stevens ((2015)), "On the Seasonal and Synoptic Time-Scale Variability of the North Atlantic Trade Wind Region and Its Low-Level Clouds", in: *Journal of the Atmospheric Sciences* 72.4, pp. 1428–1446, ISSN: 0022-4928, DOI: 10.1175/JAS-D-14-0054.1.

Fritts, David C. ((1982)), "The transient critical-level interaction in a Boussinesq fluid", in: *Journal of Geophysical Research* 87.C10, p. 7997, ISSN: 0148-0227, DOI: 10.1029/JC087iC10p07997.

Heus, T., C. C. van Heerwaarden, H. J. J. Jonker, A. Pier Siebesma, S. Axelsen, K. van den Dries, O. Geoffroy, A. F. Moene, D. Pino, S. R. de Roode, and J. Vilà-Guerau de Arellano ((2010)), "Formulation of the Dutch Atmospheric Large-Eddy Simulation (DALES) and overview of its applications", in: *Geoscientific Model Development* 3.2, pp. 415–444, DOI: 10.5194/gmd-3-415-2010.

Klein, Stephen A. ((1997)), "Synoptic Variability of Low-Cloud Properties and Meteorological Parameters in the Subtropical Trade Wind Boundary Layer", in: *Journal of Climate* 10.8, pp. 2018–2039, ISSN: 0894-8755, DOI: 10.1175/1520-0442(1997)010<2018:SVOLCP>2.0.CO;2.

Koelling, Tobias and Hauke Schulz ((2022)), *Conversion of the ICON Grid*, URL: https://gitlab.gwdg.de/structured/slice_server/-/blob/main/app.py#L210 (visited on 04/24/2023).

Lalas, D. P. and F. Einaudi ((1976)), "On the Characteristics of Gravity Waves Generated by Atmospheric Shear Layers", in: *Journal of the Atmospheric Sciences* 33.7, pp. 1248–1259, ISSN: 0022-4928, DOI: 10.1175/1520-0469(1976)033<1248:OTCOGW>2.0.CO;2.

Lane, Todd P. and Terry L. Clark ((2002)), "Gravity waves generated by the dry convective boundary layer: Two-dimensional scale selection and boundary-layer feedback", in: *Quarterly Journal of the Royal Meteorological Society* 128.583, pp. 1543–1570, ISSN: 0035-9009, DOI: 10.1002/qj.200212858308.

Lane, Todd P. and Mitchell W. Moncrieff ((2010)), "Characterization of Momentum Transport Associated with Organized Moist Convection and Gravity Waves", in: *Journal of the Atmospheric Sciences* 67.10, pp. 3208–3225, ISSN: 0022-4928, DOI: 10.1175/2010JAS3418.1.

Nappo, C. J. ((2014)), *An Introduction to Atmospheric Gravity Waves*, English, 2nd ed., vol. v. 102, International geophysics, Nappo, C. J. (VerfasserIn), Saint Louis: Elsevier Science, 385 pp., ISBN: 9780123852243, URL: https://ebookcentral.proquest.com/lib/kxp/detail.action?docID=1042788.

Nuijens, L., A. Savazzi, G. de Boer, P-E Brilouet, G. George, M. Lothon, and D. Zhang ((2022)), "The frictional layer in the observed momentum budget of the trades", eng,

in: *Quarterly Journal of the Royal Meteorological Society* 148.748, Journal Article, pp. 3343–3365, ISSN: 0035-9009, DOI: 10.1002/qj.4364, eprint: 36636229.

Savazzi, Alessandro Carlo Maria, Louise Nuijens, Irina Sandu, Geet George, and Peter Bechtold ((2022)), "The representation of the trade winds in ECMWF forecasts and reanalyses during EUREC 4 A", in: *Atmospheric Chemistry and Physics* 22.19, pp. 13049–13066, DOI: 10.5194/acp-22-13049-2022.

Schulz, Hauke ((2021)), *Picture of simulated domains*, URL: https://howto.eurec4a.eu/icon_les.html (visited on 04/24/2023).

Schulz, Hauke and Bjorn Stevens ((2023)), *On the representation of shallow convection in the trades by large-domain, hecto-meter, large-eddy simulations*, DOI: 10.31223/X5H651.

Shaw, Tiffany A. and Todd P. Lane ((2013)), "Toward an Understanding of Vertical Momentum Transports in Cloud-System-Resolving Model Simulations of Multiscale Tropical Convection", in: *Journal of the Atmospheric Sciences* 70.10, pp. 3231–3247, ISSN: 0022-4928, DOI: 10.1175/JAS-D-13-068.1.

Stephan, Claudia C., Nedjeljka Žagar, and Theodore G. Shepherd ((2021)), "Waves and coherent flows in the tropical atmosphere: New opportunities, old challenges", in: *Quarterly Journal of the Royal Meteorological Society* 147.738, pp. 2597–2624, ISSN: 0035-9009, DOI: 10.1002/qj.4109.

Stevens, Bjorn et al. ((2021)), "EUREC^{4}A", in: *Earth System Science Data* 13.8, pp. 4067–4119, DOI: 10.5194/essd-13-4067-2021.

A. Programmed code for the calculation

0.1 Calculate different cospectra and drag for specific sections

[1]:
```
### import libaries
from scipy.fft import fft
from scipy import integrate
from scipy import interpolate
import matplotlib.pyplot as plt
import numpy as np
import xarray as xr
from intake import open_catalog
```

[2]:
```
## Conversion of model half level height to full level height
cat = open_catalog("https://raw.githubusercontent.com/observingClouds/
 ↪eurec4a-intake/simulations/catalog.yml")
dh = cat.simulations.grids.EUREC4A_constants_DOM01.to_dask()

hoehe_half_level = []

for i in range(83,151):
    mi = dh.isel(cell = 450000).sel(height = i).z_ifc.values
    mi_round = np.round(mi, decimals = 2)
    hoehe_half_level.append(mi_round)
    i = i+1

# Umrechnung auf ganze Modelllevel
hoehe_ganze_level = []
for i in range(67):
    neu1 = hoehe_half_level[i]
    neu2 = hoehe_half_level[i+1]
    neu = (neu1 + neu2)/2
    neu_round = np.round(neu, decimals = 2)
    hoehe_ganze_level.append(neu_round)
```

[3]:
```
### load in original data
cat2 = open_catalog("https://raw.githubusercontent.com/eurec4a/eurec4a-intake/
 ↪master/catalog.yml")
#control simulation with horizontal resolution of 312m
```

```
ds_original = xr.open_zarr('https://swift.dkrz.de/v1/
 ↪dkrz_948e7d4bbfbb445fbff5315fc433e36a/EUREC4A_LES/experiment_2/
 ↪EUREC4A_ICON-LES_control_DOM02_3D_native.zarr')
# Load grid
grid_original = cat2.simulations.grids['3c7523ba-f78c-c23e-6654-012805ba3300'].
 ↪to_dask() #Grid mit Auflösung 312m
```

[4]:
```
##functions based on Koelling and Schulz(2022) to switch from ICON grid to␣
 ↪longitude latitude grid
def create_vectors(lat1,lat2,lon1,lon2):
    o = np.array([lon1, lat1])  # origin vector of line segment
    v = np.array([lon2 - lon1, lat2 - lat1])  # vector along the line segment
    n = np.array([v[1], -v[0]])  # vector normal to line segment
    n /= np.linalg.norm(n) # make normal vector a unit vector
    A = np.stack([v,n],axis=-1)  # stack to matrix
    return A, o, v
def create_points(lons, lats):
    pts = np.stack([lons, lats],axis=1)
    return pts
def get_distances(A, pts, origin):
    """
    s : np.array
      signed distances
    """
    l, s = np.linalg.inv(A)@(pts-origin).T
    return l, s
def create_idx(l, s, indices_lookup):
    """
    indices_lookup : np.array()
        indices of nodes, edges or cells, depending on l and s
    """

    idx = ( np.any(s[indices_lookup] >0, axis=0 )
      & np.any(s[indices_lookup] <0, axis=0 )
      & np.any(l[indices_lookup] >0, axis=0 )
      & np.any(l[indices_lookup] <1, axis=0 )
      )
    return idx
def sort_idx_by_lambda(idx, l):
    cl_idx_sorted = np.argsort(l)
    idx_sorted = np.where(idx)[0][cl_idx_sorted]
    return idx_sorted
def get_idx_along_line(lat1,lat2,lon1,lon2, grid):
    A, o, v = create_vectors(lat1,lat2,lon1,lon2)
    vlons_deg = np.rad2deg(grid.vlon)
    vlats_deg = np.rad2deg(grid.vlat)
    clons_deg = np.rad2deg(grid.clon)
```

```
    clats_deg = np.rad2deg(grid.clat)
    vpts = create_points(vlons_deg, vlats_deg)

    vl, vs = get_distances(A, vpts, o)
    vertex_of_cells = (grid.vertex_of_cell-1).values
    vidx = create_idx(vl, vs, vertex_of_cells)
    cpts = create_points(clons_deg[vidx], clats_deg[vidx])
    cl, cs = get_distances(A, cpts, o)

    # Get coords along line that are orthogonal to the center points
    pts=(cl * v[:, None]).T+o  #pts[:,0] --> longitudes; pts[:,1] -->latitudes

    vidx_sorted = sort_idx_by_lambda(vidx, cl)
    return vidx_sorted, pts[np.argsort(cl)]
```

[10]:
```
### mask nan cells and load in parts of the data to local storage for better performance
nan_cells = np.squeeze(np.where(np.isnan(ds_original.w.isel(time = 1).isel(height = 50).values)))
mask = ~np.isin(ds_original.w.cell.values, nan_cells)
grid_nonan = grid_original.sel(cell = mask)
grid_nonan = grid_nonan[["clon", "clat", "vertex_of_cell", "vlon", "vlat"]]
grid_nonan.load()
ds_original_nonan = ds_original.sel(cell = mask)
```

```
/sw/spack-levante/mambaforge-4.11.0-0-Linux-x86_64-sobz6z/lib/python3.9/site-
packages/xarray/core/indexing.py:1228: PerformanceWarning: Slicing is producing
a large chunk. To accept the large
chunk and silence this warning, set the option
    >>> with dask.config.set(**{'array.slicing.split_large_chunks': False}):
    ...     array[indexer]

To avoid creating the large chunks, set the option
    >>> with dask.config.set(**{'array.slicing.split_large_chunks': True}):
    ...     array[indexer]
  return self.array[key]
```

[11]:
```
## Conversion of the complete grid
lat_min = min(np.rad2deg(grid_nonan.clat.values))
lat_max = max(np.rad2deg(grid_nonan.clat.values))
lon_min = min(np.rad2deg(grid_nonan.clon.values))
lon_max = max(np.rad2deg(grid_nonan.clon.values))
# Result
result = get_idx_along_line(lat_min, lat_max, lon_min, lon_max, grid_nonan)
sorted_latitude_indices = result[0]
result_pts = result[1]
latitudes = result_pts[:,1]
```

```
longitudes = result_pts[:,0]
#Equal longitudes und latitudes
liste_longitudes = np.sort(longitudes) #sortierte longitudes
equal_longitudes = np.linspace(liste_longitudes.min(), liste_longitudes.max(),␣
 ↪len(liste_longitudes))
liste_latitudes = np.sort(latitudes) #sortierte longitudes
equal_latitudes_f = np.linspace(liste_latitudes.min(), liste_latitudes.max(),␣
 ↪len(liste_latitudes))
## optional: only every fifth latitude
equal_latitudes = equal_latitudes_f[::5]
```

[16]:
```
## function to generate a new dataset which can be saved in a netcdf file
def Compute_for_latitude_con(cospectrum_pw, cospectrum_tw, cospectrum_uw,␣
 ↪integrierter_wave_drag_con,
                             cospectrum_pw_5_10, cospectrum_tw_5_10,␣
 ↪cospectrum_uw_5_10, integrierter_wave_drag_con_5_10,
                             cospectrum_pw_10, cospectrum_tw_10,␣
 ↪cospectrum_uw_10, integrierter_wave_drag_con_10,
                             mean_cloud, cloud_per, time_fun):
    return xr.Dataset(coords = {'time': (['time'], np.array([ds_original.time.
 ↪isel(time = time_fun).values])),
                                'latitude': (['latitude'], equal_latitudes[:
 ↪-1]),
                            'height' : (['height'], np.array(hoehe_ganze_level))},
                data_vars = {'copw': (['time', 'latitude','height'], np.
 ↪array([cospectrum_pw]), {"long_name": "cospectrum pw convection"}),
                             'cotw': (['time', 'latitude','height'], np.
 ↪array([cospectrum_tw]), {"long_name": "cospectrum tw convection"}),
                             'couw': (['time', 'latitude','height'], np.
 ↪array([cospectrum_uw]), {"long_name": "cospectrum uw convection"}),
                             'wdcon': (['time', 'latitude', 'height'], np.
 ↪array([integrierter_wave_drag_con]), {"long_name": "wave drag convection"}),
                             'copw_5_10': (['time', 'latitude','height'], np.
 ↪array([cospectrum_pw_5_10]), {"long_name": "cospectrum pw convection >5km"}),
                            'cotw_5_10': (['time', 'latitude','height'], np.
 ↪array([cospectrum_tw_5_10]), {"long_name": "cospectrum tw convection >5km"}),
                             'couw_5_10': (['time', 'latitude','height'], np.
 ↪array([cospectrum_uw_5_10]), {"long_name": "cospectrum uw convection >5km"}),
                             'wdcon_5_10': (['time', 'latitude', 'height'], np.
 ↪array([integrierter_wave_drag_con_5_10]), {"long_name": "wave drag␣
 ↪convection >5km"}),
                             'copw_10': (['time', 'latitude','height'], np.
 ↪array([cospectrum_pw_10]), {"long_name": "cospectrum pw convection >10km"}),
                            'cotw_10': (['time', 'latitude','height'], np.
 ↪array([cospectrum_tw_10]), {"long_name": "cospectrum tw convection >10km"}),
```

```
                            'couw_10': (['time', 'latitude','height'], np.
↪array([cospectrum_uw_10]), {"long_name": "cospectrum uw convection >10km"}),
                            'wdcon_10': (['time', 'latitude', 'height'], np.
↪array([integrierter_wave_drag_con_10]), {"long_name": "wave drag convection␣
↪>10km"}),
                            'cloud': (['time', 'latitude'], np.
↪array([mean_cloud]), {"long_name": "mean cloud level for every latitude"}),
                            'cloud_per': (['time', 'latitude'], np.
↪array([cloud_per]), {"long_name": "cloud percentage for every latitude"})
                           })
```

[ ]:
```
%%time

#Definitions
Re = 6371 # mean radius of the earth
R_Abschnitt = 40 # section length of 40 km
variablen_liste = ['qc', 'w', 'u', 'temp', 'rho', 'pres'] # used variables
#handling missing data
no_values = -999*np.ones(67)
single_value = -999

for t in range(319): #for all time steps
    cospectrum_pw_all_latitudes = [] #lists to save values for every time step
    cospectrum_tw_all_latitudes = []
    cospectrum_uw_all_latitudes = []
    wavedrag_all_latitudes = []
    cospectrum_pw_all_latitudes_5_10 = []
    cospectrum_tw_all_latitudes_5_10 = []
    cospectrum_uw_all_latitudes_5_10 = []
    wavedrag_all_latitudes_5_10 = []
    cloud_level_all_latitudes = []
    cloud_percentage = []

    lists = [cospectrum_pw_all_latitudes, cospectrum_tw_all_latitudes,␣
↪cospectrum_uw_all_latitudes, wavedrag_all_latitudes,
           cospectrum_pw_all_latitudes_5_10, cospectrum_tw_all_latitudes_5_10,␣
↪cospectrum_uw_all_latitudes_5_10,
           wavedrag_all_latitudes_5_10]

    ds_local = ds_original_nonan[variablen_liste].isel(time=t).load()

    for y in range (int(np.shape(equal_latitudes)[0])-1): #all latitudes
        cell = ((grid_nonan.clat.values > np.deg2rad(equal_latitudes[y])) &
                (grid_nonan.clat.values < np.deg2rad(equal_latitudes[y+1])) )

        #find the correct cell indizes
```

```
        cell_idx = np.where(cell == True)[0]
        clon_cell = grid_nonan.clon.sel(cell= cell_idx).values
        new_cell_idx = cell_idx[~np.isnan(clon_cell)]
        cell_selection = grid_nonan.clon.sel(cell = new_cell_idx)
        values = ~np.isnan(cell_selection.values)
        nonan_cells = cell_selection.values[values]
        new_clon_cell = clon_cell[values]
        equal_longitudes_new = np.linspace(new_clon_cell.min(), new_clon_cell.
↪max(), len(new_clon_cell))
        inside_cell = np.sort(nonan_cells)
        final_cell_idx = new_cell_idx[values]
        #load datasets for better performance and deduction of running time
        # interpolation of the data on an equal longitudes-latitudes-grid
        data = []
        for variable in variablen_liste:
            load_dataset = ds_local[variable].sel(cell = final_cell_idx)
            dataset = load_dataset.to_dataset(name = variable)
            dataset_1 = dataset.assign_coords({"lon": ("cell", inside_cell)})
            dataset_lon = dataset_1.swap_dims({'cell':'lon'})
            final_dataset = dataset_lon.interp(lon = equal_longitudes_new)
            data.append(final_dataset)
        #Calculation how many cells are 40km on a latitude
        alpha =  R_Abschnitt/(Re*np.cos(np.deg2rad(equal_latitudes[y])))
        number_of_cells = int(np.round(alpha/
↪(equal_longitudes_new[1]-equal_longitudes_new[0]))) #number of cells for 40km
        qc = data[0].qc.values
        data_liste = []
        for x in range(len(data[0].lon)):
            cloud_per = np.sum(np.max(qc[:, x:x+number_of_cells] > 0.0,␣
↪axis=0))/number_of_cells #percentage of clouds in the section
            if cloud_per > 0.2: #define cloud minimum (20%)
                for i, variable in zip (range(len(data)), variablen_liste): ␣
↪#cutting the data for the section
                    cutted_data = data[i][variable][:,x:(x+number_of_cells)]
                    data_liste.append(cutted_data)
                    #calculate mean cloud height for the section
                cloud_height_mean = int(np.mean(np.where(np.max(qc[:, x:
↪x+number_of_cells] > 0.0, axis=1))))
                cloud_level_all_latitudes.append(cloud_height_mean)
                cloud_percentage.append(cloud_per)
                break
            elif len(data[0].lon) - x < 2:
                break
            else:
                continue
        if data_liste == []:
            print('empty list')
```

```
            for l in lists:
                l.append(no_values)
            cloud_level_all_latitudes.append(single_value)
            cloud_percentage.append(single_value)
            continue
        elif int(np.shape(data_liste[0].lon)[0]) < 44:
            print('to little amount of cells')
            for l in lists:
                l.append(no_values)
            cloud_level_all_latitudes.append(single_value)
            cloud_percentage.append(single_value)
            continue
        else:
            #Basics for fft
            N_signal = int(np.shape(data_liste[1].lon)[0]) # number of signal␣
↪points
            freq_signal = np.fft.fftfreq(N_signal, d = 1/(N_signal/40000)) #k␣
↪71.43

            #fft w
            w_fft = fft(data_liste[1].values, axis = 1)
            fft_complex_w = w_fft[:,:N_signal//2]

            #Filtern
            peak_freq = 0.00125 # lambda ~ 5km
            freq_signal2 = freq_signal[:N_signal//2] #symmetric result of fft
            high_wavenum_fft = fft_complex_w[5].copy() ## indizes are always␣
↪equal
            high_wavenum_fft[np.abs(freq_signal2) < peak_freq] = 0
            indizes_filter = np.nonzero(high_wavenum_fft)
            #Filter lambda = 5-10km
            wavenum_fft2 = fft_complex_w[5].copy()
            wavenum_fft2[(np.abs(freq_signal2) > peak_freq)] = 0
            wavenum_fft2[(np.abs(freq_signal2) < 0.00063)] = 0
            indizes_5_10 = np.nonzero(wavenum_fft2)
            #filtered frequency
            filtered_freq_con = freq_signal2[indizes_filter]
            freq_filter_5_10 = freq_signal2[indizes_5_10]

            #filter + conjugate
            filtered_fft_w = np.squeeze(fft_complex_w[:, indizes_filter])
            konjugiert_fft_w = filtered_fft_w.conjugate()
            filtered_fft_w_5_10 = np.squeeze(fft_complex_w[:, indizes_5_10])␣
↪#for 5-10km
            konjugiert_fft_w_5_10 = filtered_fft_w_5_10.conjugate()

            #other variables
```

```
            u_fft = fft(data_liste[2].values, axis = 1)
            fft_complex_u = u_fft[:,:N_signal//2]
            filtered_fft_u = np.squeeze(fft_complex_u[:,indizes_filter])
            filtered_fft_u_5_10 = np.squeeze(fft_complex_u[:,indizes_5_10])

            pres_fft = fft(data_liste[5].values, axis = 1)
            fft_complex_pres = pres_fft[:,:N_signal//2]
            filtered_fft_pres = np.squeeze(fft_complex_pres[:, indizes_filter])
            filtered_fft_pres_5_10 = np.squeeze(fft_complex_pres[:,
↪indizes_5_10])

            temp_fft = fft(data_liste[3].values, axis = 1)
            fft_complex_temp = temp_fft[:,:N_signal//2]
            filtered_fft_temp = np.squeeze(fft_complex_temp[:, indizes_filter])
            filtered_fft_temp_5_10 = np.squeeze(fft_complex_temp[:,
↪indizes_5_10])

            #rho and rho0
            rho_filtered = np.squeeze(data_liste[4].values[:, indizes_filter])
            rho_filtered_5_10 = np.squeeze(data_liste[4].values[:,
↪indizes_5_10])

            rho0 = []
            for i in range(67):
                r = np.mean(rho_filtered[i]+rho_filtered[i+1])
                rho0.append(r)
            rho0_5_10 = []
            for i in range(67):
                r = np.mean(rho_filtered_5_10[i]+rho_filtered_5_10[i+1])
                rho0_5_10.append(r)

            #calculation of cospectra
            cospectrum_pw = (filtered_fft_pres*konjugiert_fft_w).real
            cospectrum_tw = (filtered_fft_temp*konjugiert_fft_w).real
            cospectrum_uw = (filtered_fft_u*konjugiert_fft_w).real

            mean_cospectrum_pw = np.mean(cospectrum_pw[:-1], axis = 1) #mean
↪over cells
            mean_cospectrum_tw = np.mean(cospectrum_tw[:-1], axis = 1)
            mean_cospectrum_uw = np.mean(cospectrum_uw[:-1], axis = 1)

            cospectrum_pw_all_latitudes.append(mean_cospectrum_pw)
            cospectrum_tw_all_latitudes.append(mean_cospectrum_tw)
            cospectrum_uw_all_latitudes.append(mean_cospectrum_uw)

            cospectrum_pw_5_10 = (filtered_fft_pres_5_10*konjugiert_fft_w_5_10).
↪real
```

```
            cospectrum_tw_5_10 = (filtered_fft_temp_5_10*konjugiert_fft_w_5_10).
↪real
            cospectrum_uw_5_10 = (filtered_fft_u_5_10*konjugiert_fft_w_5_10).
↪real

            mean_cospectrum_pw_5_10 = np.mean(cospectrum_pw_5_10[:-1], axis =␣
↪1) #mittel über die Zellen
            mean_cospectrum_tw_5_10 = np.mean(cospectrum_tw_5_10[:-1], axis = 1)
            mean_cospectrum_uw_5_10 = np.mean(cospectrum_uw_5_10[:-1], axis = 1)

            cospectrum_pw_all_latitudes_5_10.append(mean_cospectrum_pw_5_10)
            cospectrum_tw_all_latitudes_5_10.append(mean_cospectrum_tw_5_10)
            cospectrum_uw_all_latitudes_5_10.append(mean_cospectrum_uw_5_10)

            #calculation drag
            klammer_con = rho_filtered*cospectrum_uw
            wave_drag_con = []
            for z in range(67):
                wd = -1/2 * (1/rho0[z])* ((klammer_con[z-1]-klammer_con[z])/
↪(hoehe_ganze_level[z-1]-hoehe_ganze_level[z]))
                integrierter_wave_drag = integrate.simpson(wd,␣
↪filtered_freq_con)
                wave_drag_con.append(integrierter_wave_drag)
            wavedrag_all_latitudes.append(wave_drag_con)


            klammer_con_5_10 = rho_filtered_5_10*cospectrum_uw_5_10
            wave_drag_con_5_10 = []
            for z in range(67):
                wd_1 = -1/2 * (1/rho0_5_10[z])*␣
↪((klammer_con_5_10[z-1]-klammer_con_5_10[z])/
↪(hoehe_ganze_level[z-1]-hoehe_ganze_level[z]))
                integrierter_wave_drag_1 = integrate.simpson(wd_1,␣
↪freq_filter_5_10)
                wave_drag_con_5_10.append(integrierter_wave_drag_1)
            wavedrag_all_latitudes_5_10.append(wave_drag_con_5_10)

    out_latitude = Compute_for_latitude_con(cospectrum_pw_all_latitudes,␣
↪cospectrum_tw_all_latitudes,
                                            cospectrum_uw_all_latitudes,␣
↪wavedrag_all_latitudes,
                                              cospectrum_pw_all_latitudes_5_10,␣
↪cospectrum_tw_all_latitudes_5_10,
                                             cospectrum_uw_all_latitudes_5_10,␣
↪wavedrag_all_latitudes_5_10,
```

```
                                        cloud_level_all_latitudes,␣
↪cloud_percentage, t)
   timestamp = str(np.array(ds_original.isel(time = t).time.values, dtype =␣
↪'datetime64[h]'))
   ## save file for every time step
   out_latitude.to_netcdf(f'/work/mh1119/m300947/results/convection/
↪cospectra_convection_{timestamp}.nc')
```