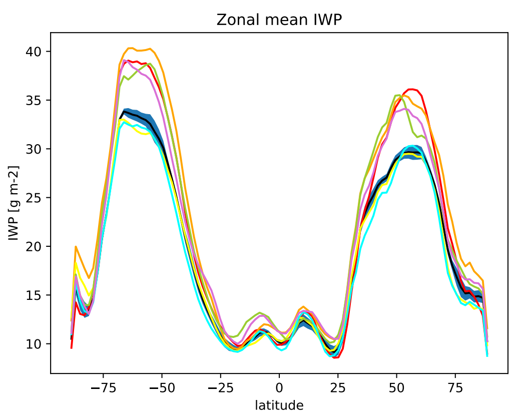
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

# Simplifying Cloud Microphysical Processes in the Climate Model ECHAM-HAM




Sina Klampt

Bachelor Thesis
November 2021

**Supervisors:**
Ulrike Proske
Prof. Dr. Ulrike Lohmann

# Abstract

Clouds are an integral part of the Earth's radiation budget. As part of the cloud, cloud microphysics have a substantial influence on Earth's climate and therefore play an important part in climate models. Since the parametrizations and processes that are dependent on each other have become more and more detailed and complex over the years, simplifying certain formulations of cloud microphysical processes may lead to a better understanding of these processes and help with the efficiency of the model while keeping the model results mostly unchanged. This bachelor thesis focuses on the two microphysical processes aggregation and accretion. For both processes, multiple simplification approaches were evaluated. While the simplifications with Taylor series expansion produced almost the exact same results to the reference model, the other approaches still produced comparable outputs. With the drastic approach, that eliminated the complex formulation for aggregation and replaced it with a constant value, the aspect of making the individual process more understandable was achieved. However, the model's efficiency could not be improved with any of the simplifications as the runtime of the two microphysical processes were much smaller and in no comparison to the runtime of the entire model.

# Acknowledgements

I would like to thank my supervisor Ulrike Proske for her time, constant support and great guidance. I have learned a lot over the last six months and am truly grateful for all the advice she has given me.

I would also like to thank Prof. Dr. Ulrike Lohmann for her valuable inputs during our meetings and giving me the opportunity to write my bachelor thesis in the Atmospheric Physics research group.

Furthermore, I would like to thank Dr. Sylvaine Ferrachat for helping in setting up the entire work environment at the beginning and answering all my technical questions regarding the model and clusters.

A big thank you also goes to David Neubauer for helping with the understanding of the implementation of the formulations and his detailed explanation of the derivation and reformulations.

Finally, I would like to thank my friends and family for their support and encouragement throughout this journey and for proof-reading my thesis.

# Contents

# List of Codes

# List of Figures

# List of Tables

# 1 Introduction

Clouds are an integral part of the Earth's radiation budget. As part of the cloud, cloud microphysics have a substantial influence on Earth's climate. They are generally shorter lived than other factors in the atmosphere like aerosol particles or greenhouse gases. Still, changes in clouds remain one of the largest uncertainties for the calculation of the response of the climate system to a given radiative forcing (Lohmann and Neubauer, 2018) [1].

Microphysical processes in mixed-phase clouds consist, among other processes, of the processes riming, accretion and aggregation. While riming might be the most important process for ice enhancement, the other two processes also describe growth of ice crystals, forming snowflakes. In riming, a snowflake grows by a water droplet freezing on said snowflake. Accretion describes the process of a snowflake's growth by collecting other ice crystals. In aggregation multiple ice crystals stick together to form a snowflake (Lamb, 2002) [2].

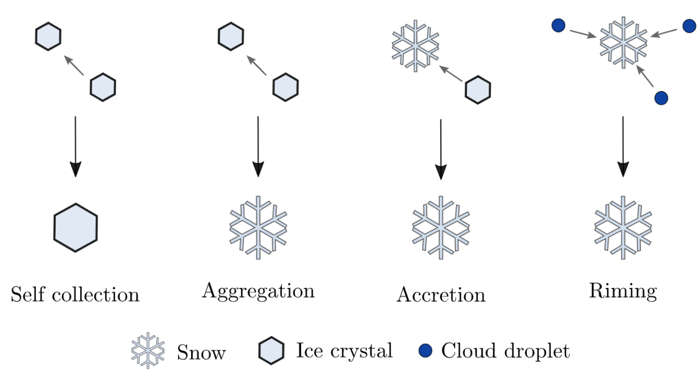



Figure 1: The four cloud microphysical processes as they are pictured in the ECHAM-HAM model. (Proske et al., 2021) [3]

The climate model used in this thesis is the global aerosol-climate model ECHAM6.3–HAM2.3. It includes the aerosol component of the fully coupled aerosol-chemistry-climate model ECHAM–HAMMOZ and evaluated using global observational data sets for clouds and precipitation (Tegen et al., 2018) [4]. The amount of low clouds, liquid and ice water path, and cloud radiative effects are more realistic in this model than in previous ones and it includes changes in the sticking efficiency for the accretion of ice crystals by snow and has consistent ice crystal shapes throughout the model. (Neubauer et al., 2019) [4, 5].

Climate models like ECHAM-HAM include parametrizations and processes that are dependent on each other and thus have become more detailed and complex over the past decades. Therefore, simplifying certain formulations of cloud microphysical processes may not only lead to a better understanding of the processes but also help with the efficiency of the model runtime, while still leaving the model results unchanged for the most part (Proske et al., 2021) [3].

This bachelor thesis focuses on two microphysical processes, namely aggregation and accretion. For both processes, multiple simplification approaches are explained and discussed. The experiments are then run by setting the switches, that were implemented for each simplification, to `TRUE`. Whereas all simplification approaches are simulated only over one year, the default model is run for 5 years to provide a more robust simulation for comparison.

The model setup for all experiments looks as follows:

- revision number: 6836
- issue number for aggregation: 793
- issue number for accretion: 791

Furthermore, all additional scripts, plots and other data can be found on my gitlab account[1].

[1] https://git.iac.ethz.ch/proskeu/bt_sina

# 2 Methods

This chapter first introduces the Linux clusters and compilers used for testing and later derives and explains the simplification for the aggregation and accretion processes.

## 2.1 Clusters and Compilers

To evaluate the experiments, different clusters and compilers were used, namely the Aerosol cluster using the Intel Fortran (ifort, version 17.0.0.098) and NAG Fortran (nagfor, version 6.0) compilers, and the Euler cluster with the Intel compiler.

### 2.1.1 Aerosol

The NAGfor compiler on Aerosol produces slow code but is in return useful for compile- and run-time debugging. For this reason, each model was run for a short 5 day trial run to make sure the code ran smoothly. Once the model was assured to be error-free, the actual simulation was moved to the Euler cluster, where the model was run for an entire year.

### 2.1.2 Euler

Euler (Erweiterbarer, Umweltfreundlicher, Leistungsfähiger ETH-Rechner) is a LINUX cluster consisting of 2 12-core Intel Xeon processors each[2]. On Euler, the tests were run for 1 year, from January 1st 2003 to December 31st 2003, with a 3 months spin-up that is not included in the analysis. To run such a long and complex model, 240 processors were used. However, since the Intel compiler does not provide too much detail for erroneous programs, it complicates the debugging. For this reason, the experiments were compiled and run for a test run on Aerosol first.

### 2.1.3 n2o

n2o is the 64-bit Linux server used at the Institute for Atmospheric and Climate Sciences (IAC). It has two 14-core Intel Xeon CPUs with a total of 56 cores and 512 GB Ram[3]. All our data used for plotting were copied

[2] https://redmine.hammoz.ethz.ch/projects/hammoz/wiki/Euler_ethz
[3] https://wiki.iac.ethz.ch/IT/LinuxN2O

from Euler and saved on the /wolke_scratch disk. For plotting, all codes were written in Python and were run using the 'iacpy3_2020' environment.

## 2.2 Simplification for Aggregation

In this section, the formulation for the aggregation process is derived and the simplification approaches are explained. Furthermore, the corresponding implementations in the model are shown. To run the model with the simplified code, switches were introduced that were set to `TRUE` if necessary. The documentation of the simplification for aggregation refers to the svn issue 793 on ECHAM-HAMMOZ.

### 2.2.1 Derivation of Formula

The conversion rate from cloud ice to snow by the aggregation process is defined as follows (Lohmann and Roeckner, 1996) [6]:

$$Q_{aggr} = \frac{\gamma_2 q_{ci}}{\Delta t_1} \tag{1}$$

$$= -\frac{\gamma_2 q_{ci}^2 \cdot \rho_{air} \cdot a_3 E_{ii} X (\frac{\rho_\sigma}{\rho_{air}})^{0.33}}{6 \cdot \rho_i \cdot \log \frac{r_{iv}}{r_{so}}} \tag{2}$$

Where:

- $\gamma_2 = 200$ is the microphysical constant that determines the efficiency of snow formation
- $q_{ci}$ is the cloud ice in the cloudy part of the grid box [kg/kg]
- $\rho_{air}$ is the air density [kg/m$^3$]
- $a_3 = 700$ s$^{-1}$ is an empirical constant (Murakami, 1990) [7]
- $E_{ii} = 0.1$ is the collection efficiency between ice crystals
- $X = 0.25$ is the dispersion of the fall velocity spectrum of cloud ice
- $\rho_\sigma = 1.3$
- $\rho_i$ is the density of ice [kg/m$^3$]

- $r_{iv}$ is the mean volume cloud droplet radius [m]
- $r_{so} = 10^{-4}$ m is the smallest radius of a particle in the snow class

The mass mixing ratio for cloud ice [kg/kg] is defined as [6]:

$$\frac{\partial q_i}{\partial t} = R(q_i) + b(Q^c_{dep} - Q^c_{aggr} - Q^c_{accr} + Q^c_{frc} + Q^c_{frh} + Q^c_{frs} - Q^c_{mlt}) + (1-b)Q^o_{dep} \tag{3}$$

where:

- $R(q_i)$ is the sum over all transport terms of qi
- superscripts $c$ and $o$ correspond to the cloudy and cloud-free part of the grid box, respectively
- $Q^c_{dep}$ is the deposition of water vapor and sublimation of cloud ice in the cloudy part
- $Q^c_{aggr}$ is the aggregation of ice crystals
- $Q^c_{accr}$ is the accretion of ice crystals by snow
- $Q^c_{frc}$ is the contact freezing of cloud droplets
- $Q^c_{frh}$ if the homogeneous freezing of cloud droplets
- $Q^c_{frs}$ is the stochastical and heterogeneous freezing of cloud droplets
- $Q^c_{mlt}$ is the melting of cloud ice

If only aggregation occurs, equation (3) can be simplified to:

$$\frac{\partial q_{ci}}{\partial t} = Q_{aggr} \tag{4}$$

where the superscript $c$ is omitted for simplicity and (2) can be generalized with:

$$Q_{aggr} = -\alpha q_{ci}^{\beta} \tag{5}$$

where $\alpha = \frac{\gamma_2 \cdot \rho_{air} \cdot a_3 E_{ii} X(\frac{\rho_\sigma}{\rho_{air}})^{0.33}}{6 \cdot \rho_i \cdot \log \frac{r_{iv}}{r_{so}}}$ is the pre-factor and $\beta = 2$ refers to the exponent of the cloud ice in the cloudy part of the grid box.

To get to the final formulation for aggregation of ice crystals, the expressions (4) and (5) are combined and the Leap-frog integration and forward Euler method steps are applied to make it numerically solvable:

$$\frac{\partial q_{ci}}{\partial t} = -\alpha q_{ci}^{\beta} \tag{6}$$

$$q_{ci}^{-\beta} \partial q_{ci} = -\alpha \partial t \tag{7}$$

$$\frac{1}{1-\beta} q_{ci,t+1}^{-\beta+1} - \frac{1}{1-\beta} q_{ci,t}^{-\beta+1} = -\alpha \Delta t \tag{8}$$

To obtain the difference in cloud ice between the two time steps, the expression (8) is reformulated and $q_{ci,t}$ subtracted:

$$q_{ci,t+1} = q_{ci,t} \cdot (1 - \alpha \Delta t \cdot (1-\beta) q_{ci,t}^{\beta-1})^{\frac{1}{1-\beta}} \tag{9}$$

$$q_{ci,t+1} - q_{ci,t} = -q_{ci,t} + q_{ci,t} \cdot (1 - \alpha \Delta t \cdot (1-\beta) q_{ci,t}^{\beta-1})^{-\frac{1}{\beta-1}} \tag{10}$$

$$= q_{ci,t} \cdot (-1 + (1 - \alpha \Delta t \cdot (1-\beta) q_{ci,t}^{\beta-1})^{-\frac{1}{\beta-1}}) \tag{11}$$

And with $\beta = 2$:

$$q_{ci,t+1} - q_{ci,t} = q_{ci,t} \cdot (-1 + (1 + \alpha \Delta t \cdot q_{ci,t})^{-1}) \tag{12}$$

The corresponding code in the model is written as:

```
zc1(1:kproma) = 17.5_dp/crhoi * prho(1:kproma) * pqrho(1:
    kproma)**0.33_dp
ztmp1(1:kproma) = -6._dp/zc1(1:kproma) * LOG10(1.e4_dp*zris
    (1:kproma))
ztmp1(1:kproma) = ccsaut / ztmp1(1:kproma)
ztmp1(1:kproma) = MERGE(ztmp1(1:kproma), 0._dp, ll1(1:kproma)
    )
zsaut(1:kproma) = pxib(1:kproma) * (1._dp - 1._dp/(1._dp +
    ztmp1(1:kproma)*ztmst*pxib(1:kproma)))
```

Code 1: Implementation of the Aggregation Formulation in the Model

Where, in comparison to the list above:

- $a_3 E_{ii} X =$ `17.5`

- `crhoi` corresponds to $\rho_i$: the density of ice [kg/m$^3$]
- `prho` corresponds to $\rho_{air}$: the air density [kg/m$^3$]
- `pqrho` corresponds to $\frac{\rho_\sigma}{\rho_{air}}$: the inverse of the air density [m$^3$/kg]
- `zris` corresponds to $r_{iv}$: the mean volume cloud droplet radius [m]
- `ccsaut` corresponds to $\gamma_2$
- `ll1` is the logical expression to determine the entries of `ztmp1`
- `zsaut` corresponds to the aggregation of ice crystals to snow [kg/kg]
- `pxib` corresponds to $q_{ci}$: the cloud ice in the cloudy part of the grid box [kg/kg]
- `ztmst` corresponds to the time step

With $pqrho = \frac{1.3}{\rho_{air}}$, the formulation in line 2 in Code 1 can be rewritten as:

$$ztmp1 = \frac{-6 \cdot \rho_i \cdot \log_{10}(10^4 \cdot zris)}{17.5 \cdot \rho_{air} \cdot \left(\frac{1.3}{\rho_{air}}\right)^{0.33}} = alpha \tag{13}$$

This expression for $ztmp1$ was defined as alpha, which from now on will refer to the term (13).

To simplify the aggregation process in the ECHAM-HAM model, different approaches to simplify the expression for alpha were evaluated. First, the code was simplified using Taylor series expansion and in a second attempt, a more drastic approach was analyzed by setting alpha to a constant value. In the following section, the two approaches are explained in further detail.

### 2.2.2 Taylor Series Expansion

For the Taylor series expansion, the $\log_{10}$ term in alpha is simplified in the following way:

$$\log\left(\frac{1+x}{1-x}\right) = 2\left(\frac{x}{1} + \frac{x^3}{3} + \frac{x^5}{5} + ...\right) \tag{14}$$

where $a = \frac{1+x}{1-x} = 10^4 \cdot zris$.

$x$ can then be rewritten as:

$$x = \frac{a-1}{a+1} = \frac{(10^4 \cdot zris) - 1}{(10^4 \cdot zris) + 1} \tag{15}$$

Using the third order Taylor polynomial from (14) and with (15), the simplification for aggregation with Taylor series expansion can then be implemented as:

```
ztmp1(1:kproma) = -6._dp / zc1(1:kproma) * &
                  2*((zris(1:kproma)*1.e4_dp - 1)/(zris(1:kproma)*1.e4_dp + 1) + &
                  ((zris(1:kproma)*1.e4_dp - 1)/(zris(1:kproma)*1.e4_dp + 1))**3/3 &
                  + ((zris(1:kproma)*1.e4_dp - 1)/(zris(1:kproma)*1.e4_dp + 1))**5/5)/LOG(10._dp)
```

Code 2: Implementation of the Taylor Series Expansion for log

This simplification can be tested by setting the switches `lsimple_aggr` and `lsimple_aggr_taylor` to `TRUE`.

The following plot compares the exact formulation of log to the approximated formulation. It can be seen that the approximation matches the exact values almost over the entire range.

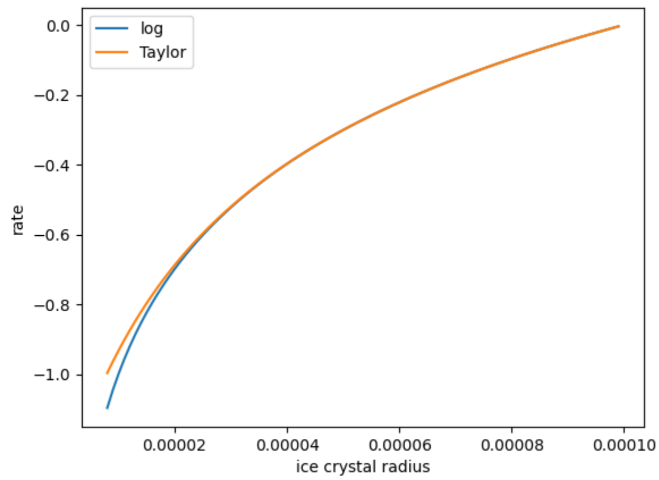



Figure 2: Comparison plot of log (blue) and its approximation with Taylor series expansion (orange).

#### 2.2.3 Drastic Simplification

For the drastic simplification, alpha, eq. (13), was set to a constant value. Using various diagnostics variables, that were implemented into the model, outputs were produced for plotting and analyzing the results.

The main diagnostics were:

- `zdiag01`: alpha
- `zdiag02`: ice crystal radius (`zris`) [m]
- `zdiag03`: air density (`prho`) [kg/m$^3$]

With these diagnostics, appropriate ranges for each variable were evaluated and all different possible values for alpha were plotted. The code used for this analysis can be found in Appendix A.1.

Once a rough estimate for alpha was found, histogram plots were created using the outputs of the diagnostic for alpha (`zdiag01`) itself. The 3D array containing the information was collapsed into a 1D array and the entries were counted and plotted. Since alpha gets merged with zeros based on the logical:

```
ll1(1:kproma) = ld_cc(1:kproma) .AND. (pxib(1:kproma) >
    cqtmin)
```

Code 3: Logical used to determine values of alpha

where:

- `ld_cc` is a switch that traces the presence of cloud cover
- `pxib` corresponds to the cloud ice in the cloudy part of the grid box [kg/kg]
- `cqtmin` corresponds to the total water minimum

most entries in this array are 0. To ignore these entries, they were first set to `NaN` before plotting. The code used to create the histogram plots can be found in Appendix A.2.

The overall mean and median values for alpha over the entire year were then evaluated as 156 and 176 respectively. Since there was not a clear indication which value would suit the model better, both values were used and evaluated in separate experiments.

This simplification can be tested by setting the switch `lsimple_aggr` to `TRUE`.

## 2.3 Simplification for Accretion

In this section, the formulation for the accretion process is derived and the corresponding implementation in the model is shown. Additionally, the simplification approaches with their corresponding implementations are explained. Switches were also introduced here to run the model with the simplified code. The documentation of the simplification for accretion refers to the svn issue 791 on ECHAM-HAMMOZ.

### 2.3.1 Derivation of Formula

The accretion rate of ice crystals by snow is defined as follows (Lohmann and Roeckner, 1996) [6]:

$$Q_{accr} = \frac{\pi E_{si} n_{os} a_6 q_{ci} \Gamma(3+b_6)}{4\lambda_s^{3+b_6}} \left(\frac{\rho_\sigma}{\rho_{air}}\right)^{0.5} \tag{16}$$

$$= \frac{\pi e^{0.025(T-T_o)} n_{os} a_6 q_{ci} \Gamma(3+b_6)}{4\left(\frac{\pi\rho_s n_{os}}{q_s}\right)^{0.8125}} \left(\frac{\rho_\sigma}{\rho_{air}}\right)^{0.5} \tag{17}$$

Where:

- $E_{si} = e^{0.025(T-T_0)}$ is the collection efficiency of snow with cloud ice
- $n_{os} = 3 * 10^6 \mathrm{m}^{-4}$ is the intercept parameter obtained from measurements (Gunn and Marshall, 1958) [8]
- $a_6 = 4.83$
- $q_{ci}$ is the cloud ice in the cloudy part of the grid box [kg/kg]
- $\Gamma$ is the Gamma function
- $b_6 = 0.25$
- $\lambda_s = \left(\frac{\pi\rho_{snow} n_{os}}{q_s}\right)^{0.25}$
- $\rho_s$ is the snow density [kg/m$^3$]
- $q_s$ is the snow mass mixing ratio
- $\rho_\sigma = 1.3$
- $\rho_{air}$ is the air density [kg/m$^3$]

If only accretion occurs, the prognostic equation for the cloud ice mass mixing ratio, eq. (3), can be simplified to:

$$\frac{\partial q_{ci}}{\partial t} = Q_{accr} \tag{18}$$

where the superscript $c$ is omitted for simplicity and (17) can be generalized as:

$$Q_{accr} = -\alpha q_{ci}^{\beta} \tag{19}$$

where $\alpha = \frac{\pi e^{0.025(T-T_o)} n_{os} a_6 \Gamma(3+b_6)}{4\left(\frac{\pi \rho_s n_{os}}{q_s}\right)^{0.8125}} \left(\frac{\rho_\sigma}{\rho_{air}}\right)^{0.5}$ is the pre-factor and $\beta = 1$ refers to the exponent of the cloud ice in the cloudy part of the grid box.

To obtain the final formulation for accretion, the above expressions (18) and (19) are combined and again the Leap-frog integration and forward Euler method steps are applied with $\beta = 1$ to make it numerically solvable:

$$\frac{\partial q_{ci}}{\partial t} = -\alpha q_{ci} \tag{20}$$

$$q_{ci}^{-1} \partial q_{ci} = -\alpha \partial t \tag{21}$$

$$\int_t^{t+1} \frac{dq_{ci}}{q_{ci}} = -\alpha \Delta t \tag{22}$$

$$\ln \frac{q_{ci,t+1}}{q_{ci,t}} = -\alpha \Delta t \tag{23}$$

To obtain the difference of cloud ice between the two time steps, the expression (23) is reformulated and $q_{ci,t}$ subtracted:

$$q_{ci,t+1} = q_{ci,t} \cdot \exp(-\alpha \Delta t) \tag{24}$$

$$q_{ci,t+1} - q_{ci,t} = -q_{ci,t}(1 - \exp(-\alpha \Delta t)) \tag{25}$$

$$-(q_{ci,t+1} - q_{ci,t}) = q_{ci,t}(1 - \exp(-\alpha \Delta t)) \tag{26}$$

The code in the model is then written as:

```
ztmp1(1:kproma) = cons4*zxsp(1:kproma)**0.8125_dp
ztmp1(1:kproma) = pi*cn0s*3.078_dp*ztmp1(1:kproma)*pqrho(1:
    kproma)**0.5_dp
ztmp1(1:kproma) = -ztmst*ztmp1(1:kproma)*zcolleffi(1:kproma)
ztmp1(1:kproma) = EXP(ztmp1(1:kproma))
ztmp1(1:kproma) = pxib(1:kproma) * (1._dp - ztmp1(1:kproma))
zsaci(1:kproma) = MERGE(ztmp1(1:kproma), 0._dp, ll2(1:kproma)
    )
```

Code 4: Implementation of the Accretion Formulation in the Model

Where, in comparison to the list above:

- `cons4` $= \left(\frac{1}{\pi \rho_s n_{os}}\right)^{0.8125}$
- `zxsp` corresponds to $q_s$: the snow mass mixing ratio

- `cn0s` corresponds to $n_{os}$: the intercept parameter obtained from measurements
- $\frac{a_6 \Gamma(3+b_6)}{4}$ = `3.078`
- `pqrho` corresponds to $\frac{1}{\rho_{air}}$: the inverse of the air density [$m^3$/kg]
- `ztmst` corresponds to the time step
- `zcolleffi` corresponds to $E_{si}$: the collection efficiency of snow for cloud ice
- `pxib` corresponds to $q_{ci}$: the cloud ice in the cloudy part of the grid box [kg/kg]
- `zsaci` corresponds to the accretion of snow with ice crystals [kg/kg]
- `ll2` is the logical expression to determine the entries of `ztmp1`

### 2.3.2 Taylor Series Expansion

The simplification for the accretion process was again done by using Taylor series expansion. This time the EXP term in the formulations was simplified.

The first order Taylor polynomial is given by:

$$e^x = 1 + \frac{x}{1!} \tag{27}$$

In the derivation for the accretion formulation, there were two situations where an EXP term was used. For this reason, the simplification was applied in both cases and the results were analyzed separately. The two cases will further be explained in the following sections.

#### 2.3.2.1 Taylor Series Expansion for the Collection Efficiency

First, the EXP term can be found in the formulation for the collection efficiency of snow with cloud ice (zcolleffi/$E_{si}$). In the code it is defined as follows:

```
1 ztmp1(1:kproma)     = fact_coll_eff*(ptp1tmp(1:kproma)-tmelt)
2 ztmp1(1:kproma)     = EXP(ztmp1(1:kproma))
3 zcolleffi(1:kproma) = MERGE(ztmp1(1:kproma), 0._dp, ll1(1:kproma))
```

Code 5: Implementation for zcolleffi

Where:

- `fact_collf_eff` corresponds to the factor for the temperature-dependent collection efficiency of snow with cold hydrometeors (Seifert and Beheng, 2006)
- `ptp1tmp` corresponds to the temporary value of the updated temperature [K]
- `tmelt` corresponds to the melting temperature of ice and snow [K]
- `zcolleffi` corresponds to the collection efficiency of snow for cloud ice
- `ll1` is the logical expression to determine the entries of `ztmp1`

The simplification of this EXP formulation in line 2 of Code 5, can then be written as:

```
ztmp1(1:kproma) = MERGE(1._dp + ztmp1(1:kproma), eps, ztmp1(1:kproma) > -1._dp)
```

Code 6: Implementation for the EXP term in zcolleffi

This simplification can be tested by setting the switch `lsimple_accr_taylor` to `TRUE`.

#### 2.3.2.2 Taylor Series Expansion in the Accretion Formulation

Later, the EXP term was also found in the formulation for accretion, as seen above in line 4 of Code 4.

The simplification of this EXP formulation was then again written in the same way as in Code 6. The final simplification was then implemented as:

```
ztmp1(1:kproma) = cons4*zxsp(1:kproma)**0.8125_dp
ztmp1(1:kproma) = pi*cn0s*3.078_dp*ztmp1(1:kproma)*pqrho(1:kproma)**0.5_dp
ztmp1(1:kproma) = -ztmst*ztmp1(1:kproma)*zcolleffi(1:kproma)
```

```
ztmp1(1:kproma) = MERGE(1._dp + ztmp1(1:kproma), eps, ztmp1
    (1:kproma) > -1._dp)
ztmp1(1:kproma) = pxib(1:kproma) * (1._dp - ztmp1(1:kproma))
```

Code 7: Implementation of the Taylor Series Expansion in the Accretion Formulation

This simplification can be tested by setting the switch `lsimple_accr` to `TRUE`.

#### 2.3.2.3 Double Taylor Series Expansion

The last experiment that was evaluated was the combination of both simplifications together.

This approach can be tested in the model by setting both the switches `lsimple_accr_taylor` and `lsimple_accr` to `TRUE`.

# 3 Analysis (Results and Discussion)

In the first two parts of this chapter, the results of all the experiments are evaluated and the corresponding plots discussed. The third part of the chapter covers the problems that occurred during the implementation of the code or other aspects of the approaches.

The following table provides a better overview of all the experiments that have been evaluated:

| Label | Simplification Form | Switches == TRUE |
|---|---|---|
| default | - | - |
| aggr_156 | alpha = 156 | lsimple_aggr |
| aggr_176 | alpha = 176 | lsimple_aggr |
| aggr_taylor | LOG as Taylor | lsimple_aggr, lsimple_aggr_taylor |
| accr_zcolleffi | EXP as Taylor | lsimple_accr_taylor |
| accr_taylor | EXP as Taylor | lsimple_accr |
| accr_double | EXP as Taylor | lsimple_accr_taylor, lsimple_accr |

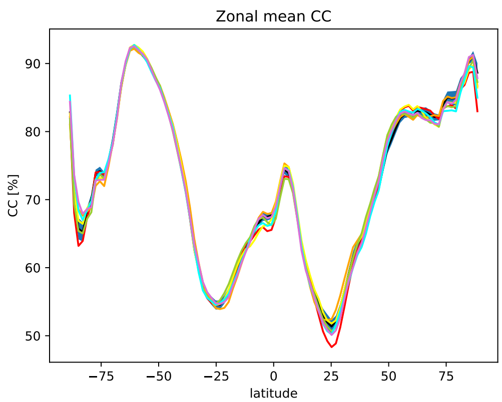
## 3.1 Plots

The default model was always used as the reference model for all our experiments. To get the best comparison between the default model and our approaches, the following variables were plotted:

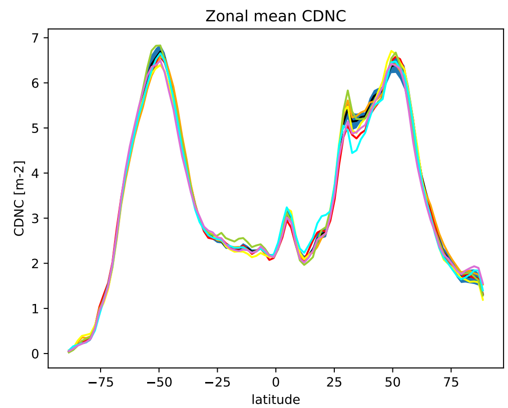
- Cloud Cover (CC) [%]
- vertically integrated Cloud Droplet Number Concentration (CDNC) [1/m$^2$]
- Ice Water Path (IWP) [g/m$^2$]
- Liquid Water Path (LWP) [g/m$^2$]
- Short Wave Radiation (SW) [W/m$^2$]
- Long Wave Radiation (LW) [W/m$^2$]

As the two processes, aggregation and accretion, both describe the process of forming a snowflake and its growth, the most influential plot is the change in IWP in the annual mean.

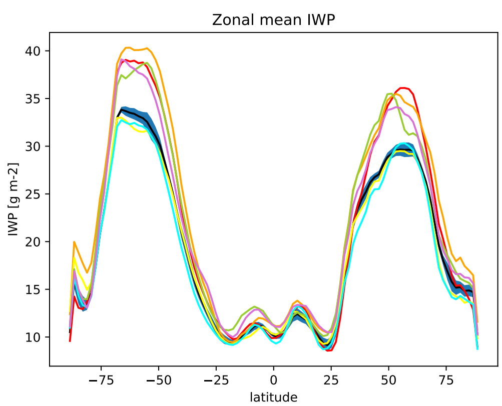
For all the different approaches, a difference plot between the default model and experiment was generated, as well as a comparison plot for annual zonal mean values. In the following figure (Fig. 3), the zonal mean plots for all experiments are summarized in one comparison plot for various variables.



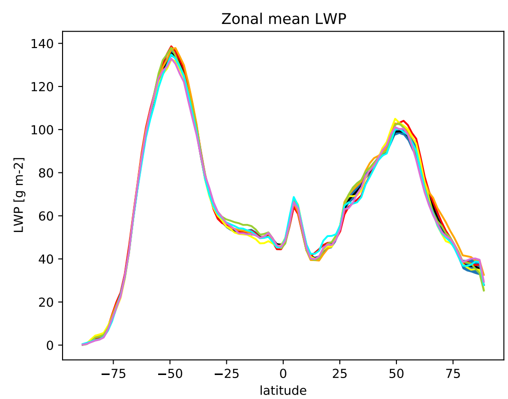

(a) CC




(b) CDNC




(c) IWP




(d) LWP

Figure 3: Zonal mean comparison plots

Where the colors correspond to following experiments:


For the default model, not only the mean was plotted but also the standard deviation, which is shown as a blue shaded region around the black line. This way, it was easier to see if the experiments are within the uncertainty range of the default model.

As already mentioned above, the most drastic difference can be seen in the IWP plot, i.e. that the difference are largest in the mid-latitudes where IWP is the highest and where they exceed 10%. This is why in the following sections only the IWP plots are shown and discussed.

### 3.1.1 Difference between Default Model and Experiments

The following plots show the difference of IWP from the experiments to the default model.

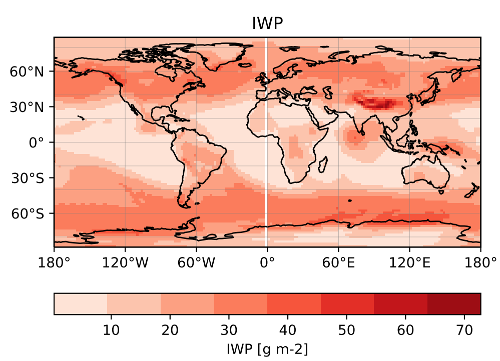
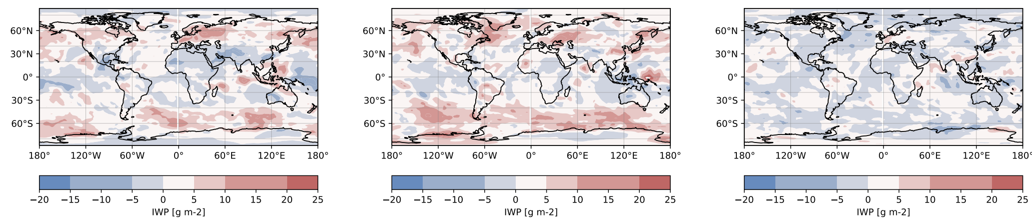
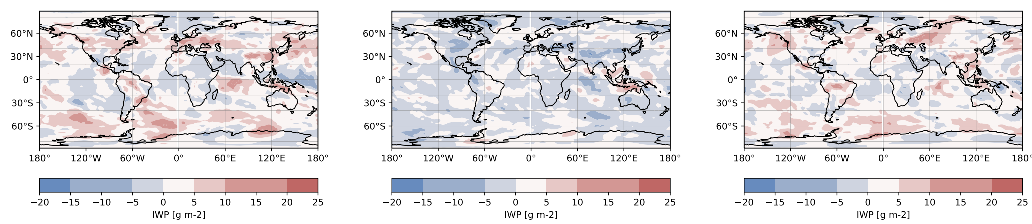



Figure 4: Default




(a) aggr_156 (b) aggr_176 (c) aggr_taylor



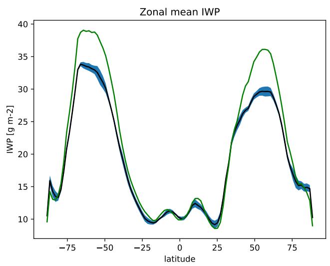

(d) accr_zcolleffi (e) accr_taylor (f) accr_double

Figure 5: Difference plots for IWP where the data from the default model were subtracted from the data of the experiments.

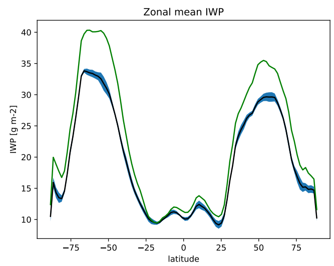
In the default model (Fig. 4), it can be seen that the values for IWP range between 0 and 75 $gm^{-2}$. In comparison, the experiments differ by up to -20 and 25 $gm^{-2}$ in certain regions or a difference of 35%. The largest difference can be seen in the subplots 5a), 5b), 5d) and 5f). However, for the experiments with the simplifications with Taylor series expansion for aggregation, subplot 5c), and for accretion, subplot 5e), the results only differ by around 7%.

#### 3.1.2 Comparison of Zonal Mean between Default Model and Experiments

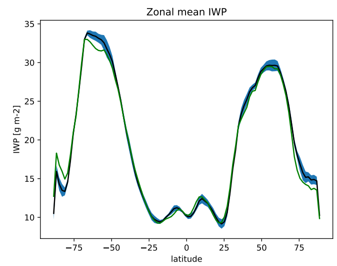
The plots shown in this section describe the comparison of the zonal mean IWP in the experiments to the default model. The black line with the blue range corresponds to the default model with its standard deviation and the green line corresponds to the experiment defined in the caption respectively.



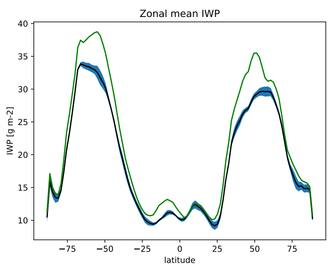

(a) aggr_156



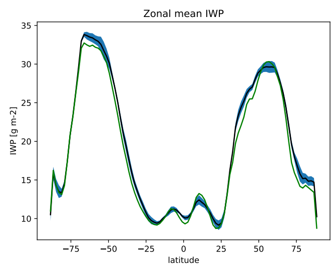

(b) aggr_176



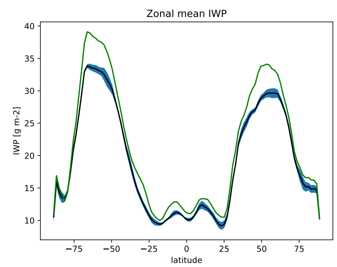

(c) aggr_taylor




(d) accr_zcolleffi




(e) accr_taylor




(f) accr_double

Figure 6: Comparison plot of Zonal Mean for IWP

It can again be seen that the data match well in the experiment with Taylor series expansion for aggregation, subplot 6c), and for accretion, subplot 6e). As in the previous section, the drastic simplification for aggregation, subplots 6a) and 6b), as well as the accretion simplification for zcolleffi, subplot 6d), and the double simplification for accretion, subplot 6f) overlap with the default model in the majority of the plots, but still show some differences, especially in the regions 50-75°N and 50-75°S by up to 20%.

The experiment usually overestimates the zonal mean in comparison to the default. Other studies (Li et al., 2012 [9] or Neubauer et al., 2019 [5]) have shown that the observed IWP values are slightly higher than in their models in the regions mentioned above. In the case of Neubauer et al. [5] this was due to a more efficient removal of ice crystals by snow due to their sticking efficiency, which allowed for a reduction of the stratiform snow formation rate by aggregation that increased IWP.

## 3.2 Comparison of Runtime

In this section the differences in runtime are shown. With simplifications such as the Taylor series expansion, where complicated functions are linearly approximated, or setting certain functions even constant, one would expect the runtime to improve significantly. However, since the individual processes, such as aggregation or accretion, make up only a small part of the entire model, these simplifications did not have an effect on the runtime at all. One can see that the ratios of the total runtime for both processes lie in the range of $10^{-4}$.

The following table summarizes the results for all the experiments:

| Default | | | |
|---|---|---|---|
| label | t_avg | t_sum | ratio t_sum |
| total | 6080.7 | 1459372. | 1 |
| aggregation | 0.9513 | 228.32 | 1.56e-4 |
| accretion | 0.5761 | 138.27 | 9.47e-5 |
| Aggregation, drastic - alpha = 156 | | | |
| label | t_avg | t_sum | ratio t_sum |
| total | 12661. | 3038541. | 1 |
| aggregation | 5.0017 | 1200.4 | 3.95e-4 |
| accretion | 3.6048 | 865.15 | 2.85e-4 |
| Aggregation, drastic - alpha = 176 | | | |
| label | t_avg | t_sum | ratio t_sum |
| total | 12820. | 3076906. | 1 |
| aggregation | 5.0174 | 1204.2 | 3.91e-4 |
| accretion | 3.6017 | 864.41 | 2.81e-4 |
| Aggregation, taylor | | | |
| label | t_avg | t_sum | ratio t_sum |
| total | 30972. | 7433173. | 1 |
| aggregation | 5.0108 | 1202.6 | 1.62e-4 |
| accretion | 3.0373 | 728.95 | 9.81e-5 |
| Accretion, zcolleffi | | | |
| label | t_avg | t_sum | ratio t_sum |
| total | 23157. | 5557733. | 1 |
| aggregation | 4.3090 | 1034.2 | 1.86e-4 |
| accretion | 2.7528 | 660.67 | 1.19e-4 |
| Accretion, taylor | | | |
| label | t_avg | t_sum | ratio t_sum |
| total | 30614. | 7347275. | 1 |
| aggregation | 4.3376 | 1041.0 | 1.42e-4 |
| accretion | 2.6421 | 634.10 | 8.63e-5 |
| Aggregation, double | | | |
| label | t_avg | t_sum | ratio t_sum |
| total | 32970. | 7912799. | 1 |
| aggregation | 4.7365 | 1136.8 | 1.44e-4 |
| accretion | 2.6319 | 631.66 | 7.98e-5 |

Table 1: The average runtime (t_avg), sum of all runtimes (t_sum) and the ratio from the experiment runtime to the total runtime (ratio t_sum) are listed.

## 3.3 Problems with Approaches

### 3.3.1 Implementation

In the beginning, there were a lot of problems when compiling, namely the model containing bugs unrelated to the simplified parts. This led to various attempts trying to resolve the issue. In a first step, single lines of code were exchanged to check whether the thought process of the simplification, or the implementation itself contained any errors. Once it was assured that this was not the case, other reasons for this problem were sought. Since in both implementations with Taylor series expansion the process was killed due to a lookup table overflow, the MERGE function was applied, as this function has also been used in previous situations to avoid overflows. Luckily, the issues could simultaneously be resolved and both experiments could finally be tested.

### 3.3.2 General

For the drastic simplification of the aggregation process, alpha was just set to a constant value. When looking at the output of that variable, it could be seen that the range was extremely large. However, randomly choosing the value for alpha led to some insights for the issue, namely the net radiation at the top of the atmosphere (TOA) reached values that were not realistic (Fnet = -4.4 $Wm^{-2}$ for alpha = 2). Knowing that the absolute value of this Fnet value needed to stay between 0 and 1, a more appropriate guess could be made. Once it was assured that the most accurate value for alpha was to be found in the range 100-200, attempts to verify the exact value were made. In our experiments, the values were Fnet = 0.38 $Wm^{-2}$ for our default model, Fnet = -0.54 $Wm^{-2}$ for alpha = 156 and Fnet = -0.15 $Wm^{-2}$ for alpha = 176, which all lie within the specified range.

# 4 Conclusions

By working with a model that consists of many complex layers and various processes that are dependent on each other, it makes sense to try and simplify certain parts. Not only does this make the model more understandable but it also makes it more efficient. This bachelor thesis approached these problems for two microphysical cloud processes, namely aggregation and accretion.

As shown in the previous chapter, simplifications exist that provide relatively similar outputs to the original model. On the one hand, when looking at the aspect of making the individual process more understandable, the drastic approach for the aggregation process was implemented. This approach eliminated the complex formula for aggregation and replaced it with a constant value. On the other hand, when trying to improve the model's efficiency, the simplifications with Taylor series expansion in both the aggregation and the accretion process were implemented. However, as the two processes account only for a minor part of the entire model, these simplifications did not have the desired outcome. Still, all the experiments resulted in comparable alternatives to the default model.

To further simplify the climate model, one could look at other processes with complex structures and formulas and try to reformulate and replace them with more straightforward expressions. If the main goal is to improve efficiency, it would be best to tackle processes that take up the most time of the model's runtime.

# A Appendix

## A.1 Code to evaluate the ranges of alpha

```python
#######################################################
# Script to estimate the ranges of alpha in the CMP routines
# Process: aggregation
# use conda env: iacpy3_2020
# 2021 09 06
# author: Sina Klampt
# sources:
#   Lohmann and Roeckner (1996): Design and performance of a new cloud microphysics scheme developed for the ECHAM general circulation model
#   Murakami (1990): NUmerical Modeling of Dynamical and Microphysical Evolution of an isolated convective Cloud: The 19 July 1981 CCOPE Cloud
#   Ulrike Proske: estimate_ranges.py
#######################################################

# packages
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as matplotlib
#from matplotlib import cm
import math
import seaborn as sns

plt.rc('axes', axisbelow=True)

cm = matplotlib.colors.ListedColormap(sns.color_palette('Blues', 6).as_hex())

output_path = 'out/'

# range for air density and ice crystal size, inv. air density calculated with rho_air
def alpha1(rho_air, r_iv):
    rho_sigma = 1.3 # [kg/m^3]
    rho_ice = 500   # crhoi [kg/m^3]
    r_so = 1e-4     # [m]

    zc1 = 17.5 * rho_air * (rho_sigma/rho_air)**(0.33) / rho_ice
    alpha = -(6/zc1 * np.log10(r_iv/r_so))
```

```python
    return alpha

# range for air density and inverse air density, ice crystal size fix
def alpha2(rho_air, rho_inv, ris):
    rho_ice = 500       # crhoi [kg/m^3]
    r_so    = 1e-4      # [m]
    ceffmin = 10e-6     # [m]
    ceffmax = 150e-6    # [m]

    # calculation ris
    ris = min(max(ris, ceffmin), ceffmax)
    ris = math.sqrt(5113188. + 2809.*ris**3) - 2261.
    ris = 1e-6 * ris**(1./3.)

    zc1 = 17.5 * rho_air * (rho_inv)**(0.33) / rho_ice
    alpha = -(6/zc1 * np.log10(ris/r_so))
    return alpha

def plot_mesh1(x, y, z):
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12/2.54,12/2.54/2), sharex=True, gridspec_kw={'wspace':0.5})
    plt.subplots_adjust(0,0,1,1)
    im = plt.pcolormesh(y, x, z)
    fig.colorbar(im, label='alpha')
    plt.xlabel(r'$r_{ice crystal}$ [m]')
    plt.ylabel(r'$\rho_{\mathrm{air}}$ [kg/m ]')
    plt.savefig(output_path+'mesh'+save_attr+'.pdf', bbox_inches='tight')
    plt.close()

def plot_mesh2(x, y, z):
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12/2.54,12/2.54/2), sharex=True, gridspec_kw={'wspace':0.5})
    plt.subplots_adjust(0,0,1,1)
    im = plt.pcolormesh(y, x, z)
    fig.colorbar(im, label='alpha')
    plt.xlabel(r'$\rho_{\mathrm{inv}}$ [m /kg]')
    plt.ylabel(r'$\rho_{\mathrm{air}}$ [kg/m ]')
    plt.savefig(output_path+'mesh'+save_attr+'.pdf', bbox_inches='tight')
    plt.close()

```

```python
def plot_mesh3(x, y, z):
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12/2.54,12/2.54/2), sharex=True, gridspec_kw={'wspace':0.5})
    plt.subplots_adjust(0,0,1,1)
    im = plt.pcolormesh(y, x, z)
    fig.colorbar(im, label='alpha')
    plt.xlabel(r'$r_{ice crystal}$ [m]')
    plt.ylabel(r'$\rho_{\mathrm{inv}}$ [m  /kg]')
    plt.savefig(output_path+'mesh'+save_attr+'.pdf', bbox_inches='tight')
    plt.close()

if __name__ == '__main__':
    rho_air  = np.arange(0.15,1.47,0.0075)      # prho -> air density [kg/m3]
    rho_inv  = np.arange(1., 8.5, 0.05)         # pqrho -> inv. air density [m3/kg]
    r_iv     = np.arange(1e-6, 1e-4, 1e-6)      # zris -> ice crystal radius [m]
    r_iv_eff = [2e-6, 3.86e-5, 7.52e-5, 1.118e-4, 1.484e-4, 1.85e-4]

    # alpha1
    array_alpha = np.zeros((np.shape(rho_air)[0],np.shape(r_iv)[0]))
    for i, rho in enumerate(rho_air):
        for j, r in enumerate(r_iv):
            array_alpha[i,j] = alpha1(rho, r)
    x = np.zeros(np.shape(rho_air)[0]+1)
    x[0] = rho_air[0]-0.0075/2
    x[1:] = rho_air[:]+0.0075/2
    y = np.zeros(np.shape(r_iv)[0]+1)
    y[0] = r_iv[0]-1e-6/2
    y[1:] = r_iv[:]+1e-6/2

    save_attr = '_alpha1'
    plot_mesh1(x, y, array_alpha)

    # alpha2
    for k in range(len(r_iv_eff)):
        ris = r_iv_eff[k]
        array_alpha = np.zeros((np.shape(rho_air)[0],np.shape(rho_inv)[0]))
        for i, rho in enumerate(rho_air):
```

```python
            for j, r in enumerate(rho_inv):
                array_alpha[i,j] = alpha2(rho, r, ris)
        x = np.zeros(np.shape(rho_air)[0]+1)
        x[0] = rho_air[0]-0.0075/2
        x[1:] = rho_air[:]+0.0075/2
        y = np.zeros(np.shape(rho_inv)[0]+1)
        y[0] = rho_inv[0]-0.05/2
        y[1:] = rho_inv[:]+0.05/2

        save_attr = str('_icr_alpha2_' + str(r_iv_eff[k]))
        plot_mesh2(x, y, array_alpha)
```

## A.2 Code to estimate alpha with histogram plots

```python
############################################################
# Script to plot the range of alpha in the CMP routines
# Process: aggregation
# use conda env: iacpy3_2020
# 2021 09 22
# author: Sina Klampt
############################################################

# packages
import matplotlib.pyplot as plt
import matplotlib as matplotlib
import seaborn as sns
import numpy as np
# netCDF
from netCDF4 import Dataset
import xarray as xr

# ignore runtime warning
import warnings
warnings.simplefilter(action = "ignore", category =
    RuntimeWarning)

from pathlib import Path
import re

months = {
    1: "jan",
    2: "feb",
    3: "mar",
    4: "apr",
```

```
    5: "may",
    6: "jun",
    7: "jul",
    8: "aug",
    9: "sep",
    10: "oct",
    11: "nov",
    12: "dec",
}

# source files
folder_path = Path('source_location/')

for file in sorted(folder_path.glob('*init_diags_2003*.01_activ*.nc')):
    print(file)
    parameters = ['DIAG01'] # name of diagnostics

    match = re.search(r'\S*init_diags_2003(\S{2})', str(file))
    month = match.group(1)
    month = months.get(int(month), 'default')
    print(month)

    for parameter in parameters:
        d1  = Dataset(file)

        run_names = [f'{month}_all', f'{month}_0-300', f'{month}_0-100', f'{month}_40-70']

        for run_name in run_names:
            var = parameter
            fig, ax = plt.subplots(nrows=1, ncols=1)

            # collapse 3D array into 1D array
            d2 = d1[var][:,:,:].ravel()
            for j in range(len(d2)):
                if d2[j] == 0:
                    d2[j] = np.nan

            d2 = d2[~np.isnan(d2)]

            # print mean and median
            print("run name:", run_name)
            print("mean:", np.mean(d2))
```

```python
        print("median:", np.median(d2))

        # create histogram plots for different ranges
        if run_name == f'{month}_all':
            plt.hist(d2, label=month)
        elif run_name == f'{month}_0-300':
            b = np.arange(0, 300, 5)
            plt.hist(d2, bins=b, label=month)
        elif run_name == f'{month}_0-100':
            b = np.arange(0, 100, 2)
            plt.hist(d2, bins=b, label=month)
        elif run_name == f'{month}_40-70':
            b = np.arange(40, 70, 0.5)
            plt.hist(d2, bins=b, label=month)

        # rename variable, for easier understanding in plot
        var = 'alpha'
        unit = ''

        # plotting
        plot_name= f'frequency_{var}_{run_name}'
        plt.legend(bbox_to_anchor=(1,1), loc="upper left")
        ax.set_ylabel('frequency')
        ax.set_xlabel(var + unit)
        #ax.set_xscale('log')

        plt.savefig(f'destination_location/plot_{plot_name}.pdf', bbox_inches='tight')
```

# References

[1] U. Lohmann and D. Neubauer, *The importance of mixed-phase and ice clouds for climate sensitivity in the global aerosol–climate model ECHAM6-HAM2*. 2018.

[2] D. Lamb, "Cloud microphysics," in *Encyclopedia of Atmospheric Sciences*, pp. 459–467, Academic Press, 2002.

[3] U. Proske, S. Ferrachat, D. Neubauer, M. Staab, and U. Lohmann, "Assessing the potential for simplification in global climate model cloud microphysics," *Atmospheric Chemistry and Physics*, 2021.

[4] I. Tegen, D. Neubauer, S. Ferrachat, C. S.-L. Drian, I. Bey, N. Schutgens, P. Stier, D. Watson-Parris, T. Stanelle, H. Schmidt, S. Rast, H. Kokkola, M. Schultz, S. Schroeder, N. Daskalakis, S. Barthel, B. Heinold, and U. Lohmann, "The global aerosol–climate model echam6.3–ham2.3 – part 1: Aerosol evaluation," *Geoscientific Model Development*, 2019.

[5] D. Neubauer, S. Ferrachat, C. S.-L. Drian, P. Stier, D. G. Partridge, I. Tegen, I. Bey, T. Stanelle, H. Kokkola, and U. Lohmann, "The global aerosol–climate model echam6.3–ham2.3 – part 2: Cloud evaluation, aerosol radiative forcing, and climate sensitivity," *Geoscientific Model Development*, 2019.

[6] U. Lohmann and E. Roeckner, "Design and performance of a new cloud microphysics scheme developed for the echam general circulation model," *Climate Dynamics*, 1996.

[7] M. Murakami *Journal of the Meteorological Society of Japan*, 1990.

[8] K. L. S. Gunn and J. S. Marshall, *The distribution with size of aggregate snowflakes*. 1958.

[9] J.-L. F. Li, D. E. Waliser, W.-T. Chen, B. Guan, T. Kubar, G. Stephens, H.-Y. Ma, M. Deng, L. Donner, C. Seman, and L. Horowitz, "An observationally based evaluation of cloud ice water in cmip3 and cmip5 gcms and contemporary reanalyses using contemporary satellite data," *Journal of Geophysical Research*, vol. 117, 2012.