Algorithmic Differentiation for Cloud Schemes (IFS Cy43r3) using CoDiPack (v1.8.1)

Manuel Baumgartner[1], Max Sagebaum[2], Nicolas R. Gauger[2], Peter Spichtinger[3], and André Brinkmann[1]

[1]Zentrum für Datenverarbeitung, Johannes Gutenberg University, Mainz, Germany
[2]Chair for Scientific Computing, Technische Universität Kaiserslautern, Germany
[3]Institute for Atmospheric Physics, Johannes Gutenberg University Mainz, Germany

Correspondence: Manuel Baumgartner (manuel.baumgartner@uni-mainz.de)

Abstract. Numerical models in atmospheric sciences do not only need to approximate the flow equations on a suitable computational grid, they also need to include subgrid effects of many non-resolved physical processes. Among others, the formation and evolution of cloud particles is an example of such subgrid processes. Moreover, to date there is no universal mathematical description of a cloud, hence many cloud schemes were proposed and these schemes typically contain several uncertain parameters. In this study, we propose the use of algorithmic differentiation (AD) as a method to identify parameters within the cloud scheme, to which the output of the cloud scheme is most sensitive. We illustrate the methodology by analyzing a scheme for liquid clouds, incorporated into a parcel model framework. Since the occurrence of uncertain parameters is not limited to cloud schemes, the AD methodology may help to identify the most sensitive uncertain parameters in any subgrid scheme and therefore help limiting the application of Uncertainty Quantification to the most crucial parameters.

1 Introduction

Modelling the atmosphere is a highly non-trivial task due to the multiscale and multicomponent nature of the atmospheric flow, where multiple physical processes on different length and timescales interact simultaneously (Orlanski, 1975). One particular result of the interaction of such processes is regularly observed in the sky: clouds appear and disappear. The evolution of a cloud (see, e.g., Lamb and Verlinde, 2011, for an in-depth treatment of cloud evolution) starts on the lengthscale of a few nanometers, where aerosol particles get wetted by the ambient water vapor leading to the formation of haze particles. If thermodynamic conditions are fulfilled, i.e. the (relative) humidity is large enough, the haze particles grow further to become cloud droplets with typical diameters of about $10\,\mu\mathrm{m} - 30\,\mu\mathrm{m}$. This growth process is described by combining Maxwell and Köhler theory (Rogers and Yau, 1989; Devenish et al., 2012; Köhler, 1936; Maxwell, 1877). Collisions of the cloud droplets eventually lead to rain drops with sizes even larger than $100\,\mu\mathrm{m}$ (see, e.g., Devenish et al., 2012; Grabowski and Wang, 2013, for a discussion including turbulence effects). Due to their weight, rain drops fall out of the cloud and form precipitation. Since all phase transitions are connected with the release or consumption of latent heat, the formation and evaporation of a cloud can affect the ambient atmospheric flow by modifying the local buoyancy (see, e.g., Cotton et al., 2010). However, all aforementioned cloud processes are microphysical processes and not resolved in numerical models, in particular not in the operational models used for weather forecasts. In these models, the cloud itself is not resolved and instead considered as a subgrid process, calling for

a representation of the impact of the cloud by using so-called "parameterizations" or "cloud schemes" (see, e.g., Khain et al., 2000). These schemes take the values of the resolved fields as input and compute the feedback of the unresolved process as an output.

In the literature, many cloud schemes are formulated as one- or two-moment schemes, predicting the mass mixing-ratio and, in case of a two-moment scheme, also the number concentration of the cloud species considered in the scheme (Khain et al., 2015), e.g. the number of cloud droplets per unit mass of dry air. However, at the moment no universal governing equation is available to describe the evolution of a cloud across all involved scales, explaining the existence of different formulations of the cloud processes. In addition, cloud schemes typically contain parameters with uncertain values, which may be introduced by artificial parameters or limited observational evidence of the precise value. A typical example for an uncertain parameter is the autoconversion rate, encoding the efficiency of colliding cloud droplets to form large and falling rain drops (Khain et al., 2015). Historically, the autoconversion rate was already introduced by Kessler, who advocated the partitioning of the whole size spectrum of droplets into small cloud droplets, with negligible sedimentation velocity, and falling rain drops (Kessler, 1969). Most cloud schemes adopt the partitioning of the size spectrum, especially all schemes of Kessler-type. The cloud scheme to be described in Section 3 below is also of Kessler-type. More examples of processes with uncertainties in their description and of how uncertainties enter the description of clouds are discussed in Khain et al. (2015).

In any case, uncertain parameters introduce uncertainty in the cloud scheme at hand, and ultimately into the numerical model as a whole. To assess the uncertainty of the parameters, one usually performs sensitivity studies, e.g. by running ensemble simulations, where each ensemble member employs slightly different values for the parameters. However, usually the number of ensemble members is limited to small values posing an additional challenge to extract the desired signal from the simulations.

In this study, we propose the use of "algorithmic differentiation" (AD) as another way of identifying the parameters with largest sensitivity. Although this method is well-known in computer science and engineering, its potential in meteorological contexts, and cloud physics in particular, is not yet fully exploited. Most applications of AD in meteorology are limited to individual studies investigating model sensitivities (usually with respect to initial values) by using adjoint models (e.g. Bischof et al., 1996b; van Oldenborgh et al., 1999; Kaminski et al., 1999; Xiao et al., 2008; Rauser et al., 2010; Zhang et al., 2013; Belikov et al., 2016) or studies targeting at applications in data assimilation (e.g. Le Dimet et al., 2002; Blessing et al., 2014). We will introduce the technique in Section 2, but, in a nutshell, it provides the derivative of a given computer code with respect to selected parameters. A cloud scheme may be described as a function f, taking the flow characteristics y from the given gridbox together with parameters η as an input, and computing the feedback z of the cloud, i.e. $z = f(y, \eta)$, as an output. Assessing the sensitivity of the output z with respect to the parameters η amounts to computing the derivative $\frac{\mathrm{d}z}{\mathrm{d}\eta}$. AD helps in evaluating this derivative by computing the derivative of the function $\widehat{f}$ with respect to the parameters, where the hat-notation indicates the implemented version of the mathematical function f in some computer code. Using this technique can help in the development of cloud schemes by providing the respective derivatives to machine accuracy in an automated fashion, i.e. without implementing finite difference approximations of $\widehat{f}$.

Recently, a field called "Uncertainty Quantification" emerged in mathematics as a more systematic combination of (numerical) analysis and statistics in order to study the propagation of uncertainties (e.g., Sullivan, 2015; Le Maître and Knio, 2010).

Although powerful methods for the investigation of uncertainties already exist, their practical use often limits the number of the considered uncertain parameters due to the curse of dimensionality, see Chertock et al. (2019) for an application in the context of cloud physics. Therefore, it is valuable to first identify the parameters with highest sensitivity using AD in order to limit a more rigorous or extensive investigation to these parameters.

We emphasize, that AD is not related to a specific application (as a cloud scheme) nor to a specific programming language. Although we use an implementation in C++ together with an AD-tool suited for this language, AD-tools for other languages like Fortran are available (e.g. Bischof et al., 1996a). A list can be found on www.autodiff.org.

In this study, we will first explain the concept of algorithmic differentiation in Section 2, introduce the warm cloud scheme used for illustration purposes within an air parcel framework in Section 3 and show results in Section 4. Some concluding remarks are given in Section 5.

2 What is Algorithmic Differentiation?

Algorithmic differentiation (AD) is a mathematical theory that describes how the computation of derivatives in a computer program can be automatized. It was developed already in the early 80's and rediscovered several times over the past years. The most known resource is the book of Griewank and Walther (2008). A nice introduction is given by Neidinger (2010).

For the purpose of computing the derivative of a program, it is considered as a sequence of simple elemental (or intrinsic) functions including the sine, cosine, multiplication, division, and addition. The concept of AD is to consider the given program as a composition of elemental functions and, by applying the chain rule, to compute its directional derivative by accumulating the derivatives of each elemental function within the program code. It is important to stress that AD does not generate a generalized representation of the derivative of the computer program. Instead, AD computes the derivatives alongside the execution path. The path might change due to conditional instructions within the code.

As an example, assume the computer program to be differentiated is given by

$$w = (a+b) * (c-d), \tag{1}$$

where a, b, c, d are the input variables and w is the output. This program can now be split into elemental functions which yields the intermediate steps t_1, t_2 required by AD as follows:

$$\begin{aligned} t_1 &= a+b, \\ t_2 &= c-d, \\ w &= t_1 * t_2. \end{aligned} \tag{2}$$

For the application of the chain rule the Jacobian matrix has to be computed for each of these intermediate steps with respect to all input variables which is quite simple, e.g. for $t_1 = t_1(a,b,c,d) = a+b$ the matrix is $(1,1,0,0)$ since $\frac{\partial t_1}{\partial a} = \frac{\partial t_1}{\partial b} = 1$ and $\frac{\partial t_1}{\partial c} = \frac{\partial t_1}{\partial d} = 0$. However, for notational convenience it is common to drop the indication of the formal dependence of the elemental function t_1 and its derivatives on the input parameters c, d since they are unchanged in t_1. In order to compute the

directional derivative, the Jacobian matrix is multiplied with the desired direction $(\dot{a}, \dot{b}, \dot{c}, \dot{d})^T$, where the dot notation is used in the AD theory to describe the corresponding derivative direction of a variable. Applying this process to the full procedure (2), the result is

$$\begin{aligned} \dot{t}_1 &= \dot{a} + \dot{b}, \\ \dot{t}_2 &= \dot{c} - \dot{d}, \\ \dot{w} &= t_2 * \dot{t}_1 + t_1 * \dot{t}_2, \end{aligned} \tag{3}$$

since, e.g.,

$$\dot{t}_1 = \left(\frac{\partial t_1}{\partial a}, \frac{\partial t_1}{\partial b} \right) \cdot \left(\dot{a}, \dot{b} \right)^T = (1, 1) \cdot \left(\dot{a}, \dot{b} \right)^T = \dot{a} + \dot{b}, \tag{4}$$

where we again neglected the formal dependency of t_1 on c and d. By computing the corresponding directional derivative statements in procedure (3) alongside the original statements in procedure (2), the directional derivative is computed for the whole computer program (1). Note that the choices $(\dot{a}, \dot{b}, \dot{c}, \dot{d}) = (1, 0, 0, 0)$ and $(\dot{a}, \dot{b}, \dot{c}, \dot{d}) = (0, 1, 0, 0)$ for the input directions for the AD computation results in the computation of the partial derivatives $\frac{\partial w}{\partial a}$ and $\frac{\partial w}{\partial b}$.

The example shows the application of the forward AD mode on a simple computer program. The general theory of AD assumes that the elemental functions are evaluated at argument values where they are differentiable in a neighborhood of the argument value. Actually, many common elemental functions are differentiable everywhere as, e.g., the addition or multiplication, but there are elemental functions which are only differentiable except for some critical points. Examples for elemental functions with a critical point are the square root, the absolute value or division, which are not differentiable at the origin. It mainly depends on the AD tool how these difficulties are treated. The AD tool "CoDiPack" used in this study provides the preprocessor option `CODI_CheckExpressionArguments` to throw an exception if such a critical point is encountered. In any case, the exact derivative of each elemental function, at least apart from the critical points, may be written down explicitly. Note that conditional instructions (e.g. if-else switches) do not pose any problem, since this only alters the program path, i.e. the specific sequence of elemental instructions which are executed. Since the whole computer program is a composition of the differentiable elemental operations, the chain rule states that also the whole program is differentiable (apart from the critical points). Moreover, if the program is represented by the function $\widehat{f} : \mathbb{R}^n \to \mathbb{R}^m$, the so-called forward mode of AD computes

$$\dot{z} = \frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)\, \dot{x}. \tag{5}$$

In Equation (5), $\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)$ is the Jacobian of the full program at state x, vector $\dot{x}$ is the direction for which the program derivative is desired along the computational path and the variable x subsumes all input variables. In the notation of Section 1, we have $x = (y, \eta)$, i.e. x contains the cloud model and thermodynamic variables y together with the inherent parameters η. In this terminology, also an inherent parameter of the (cloud) model is now considered as a parameter, if this parameter is to be investigated.

Equation (5) just states which result is computed by the forward mode of AD and not how it is computed. The evaluation of the derivative is done alongside of the primal computation of $\widehat{f}(x)$ by applying Equation (5) to each elemental operation as

in the above example, i.e. the full Jacobian is never computed explicitly but accumulated by considering each code statement individually.

The second operation mode of AD is called the reverse AD mode. As will become clear, the reverse mode can be introduced by multiplying an adjoint direction from the left side to the derivative representation of the computer program, instead of multiplying a derivative direction from the right as in Equation (5). This yields the general equation $\bar{x}^T = \bar{z}^T \frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)$. The standard AD notation for the adjoint of a variable v is the bar notation $\bar{v}$ and may be thought of as containing the immediate derivative of the current statement with respect to the particular variable.

As an example, for the statement $w = t_1 * t_2$, the AD reverse mode evaluation is

$$(\bar{t_1}, \bar{t_2}) = \bar{w} \cdot \left(\frac{\partial w}{\partial t_1}, \frac{\partial w}{\partial t_2} \right) = \bar{w} \cdot (t_2, t_1) = (\bar{w} \cdot t_2, \bar{w} \cdot t_1) \,. \tag{6}$$

The information flow in Equation (6) is reversed for the adjoint variables: the input variable is $\bar{w}$ while $\bar{t}_1$ as well as $\bar{t}_2$ are output variables. Because of this reversal of the information flow, all reverse AD statements need to be evaluated in the reversed order. The reverse of the last statement of the program code $\widehat{f}$ will be evaluated first, the second last statement as second, and so on. The reverse AD procedure for the example procedure (2) is then

$$\begin{aligned}
\bar{t}_1 &= t_2 * \bar{w}, \\
\bar{t}_2 &= t_1 * \bar{w}, \\
\\
\bar{c} &= \bar{t}_2, \\
\bar{d} &= -\bar{t}_2, \\
\\
\bar{a} &= \bar{t}_1, \\
\bar{b} &= \bar{t}_1.
\end{aligned} \tag{7}$$

As discussed above, the statements from procedure (2) are now handled in the reverse order. The values $\bar{a}$, $\bar{b}$, $\bar{c}$ and $\bar{d}$ contain the derivatives of w with respect to themselfs. Taking $\bar{d}$ as an example, according to the chain rule this is

$$\frac{\mathrm{d}w}{\mathrm{d}d} = \bar{w} \frac{\mathrm{d}w}{\mathrm{d}t_2} \frac{\mathrm{d}t_2}{\mathrm{d}d} = \bar{w} \cdot t_1 \cdot (-1) = -t_1 \cdot \bar{w} \tag{8}$$

and the value of t_1 is taken from the primal evaluation of the program. By choosing $\bar{w} = 1$ as input for the reverse AD mode, the adjoint variable $\bar{d}$ contains the derivative of procedure (2) with respect to the input d.

The example (7) shows the application of the reverse AD mode on a simple computer program. According to the general theory of AD, given a computer program which can be represented by the function $\widehat{f} : \mathbb{R}^n \to \mathbb{R}^m$, the reverse AD mode computes

$$\bar{x} = \left[\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x) \right]^T \bar{z} \tag{9}$$

where, again, $\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)$ denotes the Jacobian of $\widehat{f}$ and $\left[\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x) \right]^T$ its transpose while $\bar{z}$ represents the desired direction for the derivative. The equation (9) again just states the result that is computed by the reverse mode of AD and not how it is computed.

The actual evaluation of the derivative is done without forming the Jacobian and instead by storing informations during the primal computation of $\widehat{f}(x)$. Afterwards, a reverse sweep over the stored information is performed. This reverse sweep applies a slightly modified version of Equation (9) to each elemental function as in the above example.

Both operation modes of AD are connected via the discrete adjoint operator. Let $\langle\cdot,\cdot\rangle_n$ and $\langle\cdot,\cdot\rangle_m$ denote the Eucledian scalar products in $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively. We now select an arbitrary direction $\bar{z}\in\mathbb{R}^m$, which we apply to the result of the forward mode. This yields the equality

$$\begin{aligned}\langle\bar{z},\dot{z}\rangle_m &= \left\langle \bar{z}, \frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)\,\dot{x}\right\rangle_m = \left\langle \left[\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)\right]^T \bar{z}, \dot{x}\right\rangle_n \\ &= \langle\bar{x},\dot{x}\rangle_n\end{aligned} \tag{10}$$

by shifting the Jacobian matrix of $\widehat{f}$ to the left side of the scalar product. This shows that the reverse mode is the discrete adjoint of the forward mode (see, e.g., Kalnay, 2003, for the use of adjoint models in atmospheric data assimilation).

The advantage of the reverse mode becomes clear, if we assume that we want to compute the full gradient of a function $\widehat{f}:\mathbb{R}^n\to\mathbb{R}^m$. In case of $m=1$, i.e. a computer program with n input and a single output variable, the full gradient $\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)$ is a matrix with n columns and one row, hence its transpose is a matrix with n rows and a single column. Since $m=1$, the direction $\bar{z}$ is a vector with a single entry. Consequently, we obtain the result $\bar{x}$ by computing (9) exactly once with the single input $\bar{z}=(1)\in\mathbb{R}^1$. Using the forward mode of AD, we infer from (5) that the computation of the full gradient requires n subsequent computations with the choices $\dot{x}=e_1,\dot{x}=e_2,\ldots,\dot{x}=e_n$ and $e_i\in\mathbb{R}^n$ denotes the i-th unit vector. If n is large, these n subsequent computations require much more computational effort than a single (but slightly more costly) computation using the reverse mode of AD.

In contrast, if $n=1$, i.e. the computer program has a single input and m output variables, the forward mode of AD computes the full derivative with a single computation by choosing the 1×1-matrix $\dot{x}=(1)$ as input direction, whereas the reverse mode needs m computations; being costly for large values of m.

An alternative approach to compute the derivative of $\widehat{f}:\mathbb{R}^n\to\mathbb{R}^m$ in the direction $d\in\mathbb{R}^n$ is to apply the finite difference approach

$$\frac{\mathrm{d}\widehat{f}}{\mathrm{d}x}(x)\cdot d \approx \frac{\widehat{f}(x+td)-\widehat{f}(x)}{t} \tag{11}$$

with a (small) stepsize $t>0$, requiring two evaluations of the program $\widehat{f}$. Instead of the approximation (11), one could alternatively choose a finite difference approximation of higher order (e.g. Grossmann and Roos, 2007), but these typically need even more program evaluations. In contrast to AD, the finite difference approach requires the choice and tuning of the stepsize $t>0$ to achieve the desired accuracy of the derivative. Moreover, the optimal value of the stepsize will in general depend on the selected direction d and the state x (see Elizondo et al., 2002, for a comparison with AD). These issues render the finite difference approach as quite unattractive but due to its simplicity it is often used, accepting all drawbacks of the method.

If $\widehat{f}$ is a linear function, then an arbitrary stepsize t can be chosen for all directions. For non-linear functions, t should be as small as possible to achieve the desired accuracy but large enough to avoid cancellation errors due to the difference in (11).

AD has the advantage of not having to choose and tune a stepsize. Since the derivative of each elemental function is known exactly and AD applies the chain rule, the computed derivatives are accurate up to machine precision (Griewank et al., 2012).

Moreover, using the reverse mode for computing the full gradient in case of only a small number of output variables, AD has the potential of being several times faster compared to using finite difference approximations.

AD is introduced into computer programs mostly in two ways: Either through operator overloading or through source transformation. For C++, the majority of tools (as listed at www.autodiff.org) use the operator overloading approach which is also used by the AD tool CoDiPack (Sagebaum et al., 2017a) developed by the authors from the Scientific Computing group in Kaiserslautern and employed in this study. CoDiPack uses expression templates to reduce the amount of required information for the reverse AD mode. The data layout of the information is such that a minimal memory footprint is required and caching strategies of the processors can be applied. The general focus of CoDiPack is its application in High Performance Computing environments which was successfully demonstrated in Sagebaum et al. (2017b) and Albring et al. (2016).

The source transformation approach is mostly used in Fortran source codes. Here, the code is parsed and new code is generated which adds the additional statements for the forward or reverse AD mode. Tapenade (e.g., Hascoët and Pascual, 2013) is the most wide-spread tool for source transformations in Fortran. It is written in Java and supports nearly all features of older Fortran standards. The support for more modern features in newer Fortran versions is an ongoing development.

In general AD can be applied to any computer program. After an initial effort, the derivative computations can be automatized in the sense that every change in the code will immediately affect the primal computation and also the derivative evaluation. How much time the initial effort requires depends strongly on the code and which AD tool is applied. A general effort on analyzing software and detecting problematic code constructs is done in Hück et al. (2015). In general, operator overloading tools are usually quite easy to introduce into a code if it is well written and there is a distinct place where the computation type of the program can be defined. Source transformation usually requires a much larger effort. In both cases an early introduction of AD into the code reveals wrong implementation assumptions and yields a cleaner code.

3 The Warm Cloud Scheme

As an application for AD, we consider a slightly generalized one-moment scheme for warm cloud microphysics, i.e. liquid clouds without ice, within a zero-dimensional air parcel framework. One-moment schemes are designed to predict the temporal evolution of the mass of non-sedimenting cloud droplets, rain drops and water vapor, i.e. the mixing-ratios $q_c = \frac{M_c}{M_a}$, $q_r = \frac{M_r}{M_a}$ and $q_v = \frac{M_v}{M_a}$ where M_c is the mass of cloud droplets, M_r the mass of rain drops, M_v the mass of water vapor and M_a the mass of dry air. One-moment schemes have a long history and are governed by the classical partitioning of the droplet spectrum into non-sedimenting cloud droplets and larger rain drops, which fall down due to the gravitational acceleration (Kessler, 1969). Although these schemes remain the default choice in many computational models, for example, the operational numerical weather forecast models IFS (ECMWF, 2017), run by the European Center for Medium Range Weather Forecast (ECMWF), and COSMO (Doms et al., 2011), run by the German Weather Service (DWD), much consensus exists that two-moment schemes are in general more accurate (Igel et al., 2015). The difference between a one-moment and a two-moment scheme is

that the two-moment scheme does not only predict the evolution of the mass mixing-ratios but also the corresponding number concentrations.

The one-moment warm cloud schemes of the IFS and the COSMO model may be written in generic form as (see Rosemeier et al., 2018; Porz et al., 2018)

$$\frac{\mathrm{d}q_c}{\mathrm{d}t} = \underbrace{c\cdot(S-1)\,q_c^{\frac{1}{3}}}_{\text{Condensation}} - \underbrace{a_1 q_c^{\gamma}}_{\text{Autoconversion}} - \underbrace{a_2 q_c^{\beta_c} q_r^{\beta_r}}_{\text{Accretion}}, \tag{12a}$$

$$\frac{\mathrm{d}q_r}{\mathrm{d}t} = \underbrace{a_1 q_c^{\gamma}}_{\text{Autoconversion}} + \underbrace{a_2 q_c^{\beta_c} q_r^{\beta_r}}_{\text{Accretion}} \tag{12b}$$

$$+ \underbrace{\left(e_1 q_r^{\delta_1} + e_2 q_r^{\delta_2}\right)\min\left(S-1,0\right)}_{\text{Evaporation}} \tag{12c}$$

$$+ B - \underbrace{d q_r^{\zeta}}_{\text{Sedimentation}}, \tag{12d}$$

$$\frac{\mathrm{d}q_v}{\mathrm{d}t} = -c\cdot(S-1)\,q_c^{\frac{1}{3}} - \left(e_1 q_r^{\delta_1} + e_2 q_r^{\delta_2}\right)\min\left(S-1,0\right) \tag{12e}$$

with the coefficients c, a_1, a_2, e_1, e_2, d, the exponents $\gamma, \beta_c, \beta_r, \delta_1, \delta_2, \zeta$ and the saturation ratio $S = \frac{p_v}{p_{\mathrm{sat}}}$, comparing the partial pressure p_v of water vapor to the saturation vapor pressure p_{sat} over a flat surface of water. The scheme includes the following processes: (i) Condensational growth of cloud droplets; (ii) Autoconversion, describing the formation of rain drops by colliding cloud droplets; (iii) Accretion, describing the collection of cloud droplets by falling rain drops; (iv) Evaporation of rain drops; (v) Sedimentation of rain drops out of the grid box. The term B subsumes the flux of rain drops falling from above into the considered gridbox. The splitting of the evaporation term in (12d) is due to the appearance of the ventilation factor in the diffusional growth equation for the rain drops, taking a non-uniform distribution of water vapor around the falling rain drop into account.

The major differences of Equations (12) to the actual scheme in the operational model are the formulations of the sedimentation process as a sum of the incoming and outgoing fluxes B and dq_r^{ζ}, and the use of an explicit condensation term, which is usually circumvented by employing a saturation adjustment scheme (Asai, 1965; McDonald, 1963; Langlois, 1973; Soong and Ogura, 1973; Kogan and Martin, 1994). Note, that the values of the coefficients may also depend on the environmental conditions. Also note, that the formulation in (12) does not contain a term for the activation of new cloud droplets. Within the operational models, the activation of cloud droplets is done with the help of the saturation adjustment, where the excess water vapor is converted into mass of cloud droplets, thus always activating the maximal number if not restricted otherwise.

After choosing an appropriate set of coefficients and exponents, Equations (12) represent a cloud scheme for a warm cloud in the spirit of Kessler, although not every choice of parameters yields a physically meaningful scheme. Apart from the condensation term, the parameterizations of the other processes are not based on purely physical reasoning. Rather the structure of the terms represent in some sense ad-hoc formulations and approximations, but may also be motivated in the sense of population dynamics. The values of the coefficients and the exponents are usually obtained by fitting to observational data or results of

detailed models (e.g. Khairoutdinov and Kogan, 2000). In any case, the precise values of the coefficients and the exponents are uncertain to some degree.

For simplicity, we consider the cloud scheme (12) within an adiabatic air parcel, providing a natural framework to start with in the development of cloud schemes. The closure of (12) is given by the evolution equations for pressure p and temperature T

$$\frac{\mathrm{d}p}{\mathrm{d}t} = -\frac{g}{\overline{R}T} wp, \tag{13a}$$

$$\left(c_{pa} + c_{pv} q_v + c_{pl}\left(q_c + q_r\right)\right) \frac{\mathrm{d}T}{\mathrm{d}t} = -gw - L\frac{\mathrm{d}q_v}{\mathrm{d}t}, \tag{13b}$$

where g denotes the gravitational acceleration, $\overline{R} = R_a \left(1 + \frac{1-\varepsilon}{\varepsilon} \frac{q_v}{1+q_v}\right)$ the gas constant for moist air, R_a the gas constant for dry air, R_v the gas constant for water vapor, $\varepsilon = \frac{R_a}{R_v}$ the quotient of the gas constants of dry air and water vapor, w the vertical velocity and L the latent heat of vaporization. The coefficient c of the condensation is given by

$$c = 4\pi \left(\frac{3}{4\pi\rho_l}\right)^{\frac{1}{3}} H N_c^{\frac{2}{3}} \tag{14}$$

with ρ_l the density of water, N_c an assumed constant number concentration of the cloud droplets and the thermodynamic correction (Howell factor)

$$H = \left[\left(\frac{L}{R_v T} - 1\right)\frac{L}{KT} + \frac{R_v T}{D p_{\mathrm{sat}}}\right]^{-1} \tag{15}$$

for the condensational growth of a cloud droplet. In (15), K denotes the thermal conductivity of dry air and D the diffusivity of water vapor in air. Note that the choice of a fixed constant cloud droplet number N_c in (14) is motivated by the default Kessler-type warm cloud scheme of the IFS model (ECMWF, 2017).

Using the notation introduced in Section 1, the combined discretization of the governing Equations (12) and (13) represents the (mathematical) function $f = f(y, \eta)$, taking the values of the foregoing timestep

$$y = \left(p^{\mathrm{old}}, T^{\mathrm{old}}, q_c^{\mathrm{old}}, q_r^{\mathrm{old}}, q_v^{\mathrm{old}}\right) \tag{16}$$

together with the parameters

$$\eta = (a_1, a_2, e_1, e_2, d, \gamma, \beta_c, \beta_r, \delta_1, \delta_2, \zeta) \tag{17}$$

to compute the state of the system at the new time-level, i.e. computing $z = f(y, \eta)$. Implementing f yields the function $\widehat{f}$, from which AD can compute the derivatives with respect to the parameters η.

4 Application of Algorithmic Differentiation

We implemented the air parcel model in C++ and discretized the ordinary differential equations using the classical explicit Runge-Kutta method of order four (Hairer et al., 1993), although any other method could be used as well. We chose the

Table 1. Values of the coefficients and exponents used for the cloud scheme. All coefficients except for e_1, e_2 are independent of the environmental pressure and temperature. The values for the exponents are exact; the values for the coefficients are rounded to three digits. The coefficient values displayed for e_1, e_2 are for environmental pressure $850\,\mathrm{hPa}$ and temperature $270\,\mathrm{K}$.

a_1	a_2	e_1	e_2	d	
$1.842\,\mathrm{s}^{-1}$	$134\,\mathrm{s}^{-1}$	$-5.025\cdot10^{-7}\,\mathrm{s}^{-1}$	$1.772\cdot10^{-4}\,\mathrm{s}^{-1}$	$4\cdot10^{-3}\,\mathrm{s}^{-1}$	
γ	β_c	β_r	δ_1	δ_2	ζ
2.47	1.15	1.15	$\frac{10}{9}$	$\frac{127}{360}$	1

values of the parameters according to the warm rain scheme used in the operational forecast model IFS (ECMWF, 2017). In this case, all exponents are independent of the environmental conditions and only the coefficients e_1, e_2 of the evaporation depend on the environmental conditions. Table 1 collects the values of the constant coefficients and exponents together with the values of e_1, e_2 at pressure $850\,\mathrm{hPa}$ and temperature $270\,\mathrm{K}$. Note that e_1, e_2 vary only weakly with pressure and temperature. Prior to each timestep, we compute the values of the parameters e_1, e_2 for the cloud model using the environmental values of pressure and temperature from the old timestep. This fixes the values of the parameters for the call to the function $\widehat{f}$, computing numerically a single timestep of the governing Equations (12) and (13). Using AD, we compute the derivative of the implemented code $\widehat{f}$ with respect to the parameters (17) at the current timestep.

In the following, we always assume a constant vertical velocity w, the initial environmental conditions $270\,\mathrm{K}$ and $850\,\mathrm{hPa}$, the constant cloud droplet number density $\rho N_c = 50\,\mathrm{cm}^{-3}$ (as suggested in ECMWF (2017) over ocean), no infall of rain drops from above $B = 0\,\mathrm{s}^{-1}$ and the constant timestep $\Delta t = 0.01\,\mathrm{s}$. To arrive at a meaningful sedimentation rate, we adopt the assumption of constant terminal velocity of the rain drops from ECMWF (2017) and assume an air parcel with height $h = 1000\,\mathrm{m}$.

4.1 Cloud Formation in Updraft

As the first example we consider an updraft velocity $w = 1\,\mathrm{m\,s}^{-1}$, the initial conditions

$$(q_c(0), q_r(0)) = \left(10^{-10}\,\mathrm{kg\,kg}^{-1}, 0\,\mathrm{kg\,kg}^{-1}\right) \tag{18}$$

for the mixing-ratios, and $S(0) = 1$ for the saturation ratio, integrated for $1950\,\mathrm{s}$. The reason of not choosing $q_c(0) = 0\,\mathrm{kg\,kg}^{-1}$ is that Equation (12a) needs a non-zero initial value to allow a non-constant and non-zero solution. Physically, this reflects the fact that the cloud scheme (12) does not include a precise activation mechanism which predicts the number of activated cloud droplets as a function of saturation ratio. Such a mechanism is not included in the operational model due to their usage of a saturation adjustment scheme (e.g. Kogan and Martin, 1994).

Figure 1 shows the temporal evolution of the saturation ratio S (left panel), the cloud droplet mixing-ratio q_c (middle panel), and the rain drop mixing-ratio q_r (right panel). Apparently, the saturation ratio increases initially due to the adiabatic cooling, until cloud droplet mass increased enough to balance the source for saturation ratio from the adiabatic cooling by

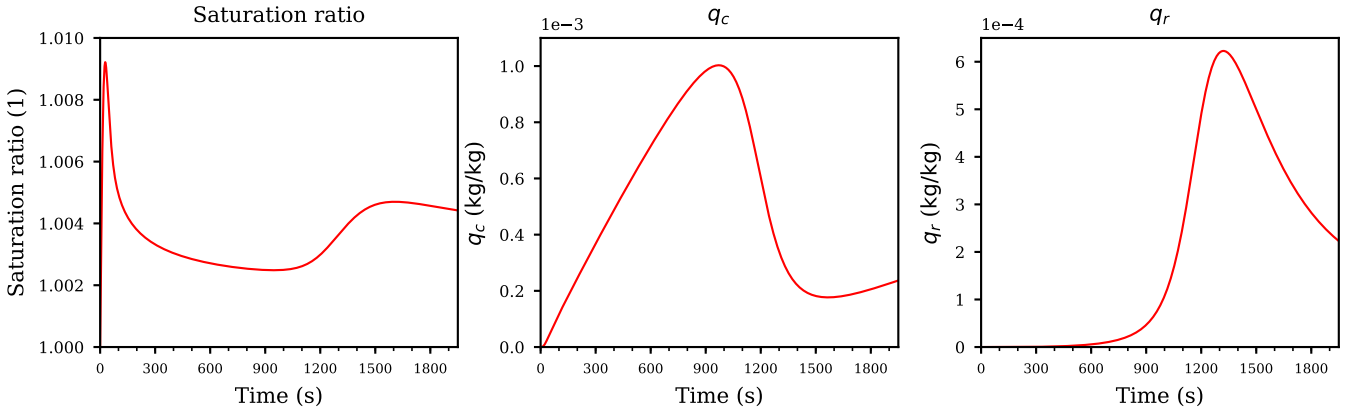

Figure 1. Temporal evolution of the saturation ratio S (left panel), the cloud droplet mixing-ratio q_c (middle panel), and the rain drop mixing-ratio q_r (right panel) for the ascending air parcel with no pre-existing cloud.

the diffusional growth of the cloud droplets. Since autoconversion is the only process to form rain, its formation starts after enough cloud droplet mass is available. The time interval between $900\,\mathrm{s}$ and $1200\,\mathrm{s}$ is a transition region, where the influence of autoconversion decreases while the influence of accretion increases, i.e. falling rain drops start to effectively collect cloud droplets and consequently increase the rain drop mass mixing-ratio q_r. This may also be seen directly from the model equation (12): Accretion is modelled as $a_2 q_c^{\beta_c} q_r^{\beta_r}$, hence an increase of rain drop mass directly increases the influence of this process and the further decrease in cloud droplet mass may be attributed to accretion. At about $1200\,\mathrm{s}$ the saturation ratio starts to increase again, since the decreasing cloud droplet mass diminishes the sink for water vapor due to their condensational growth. Note that, according to Equation (12d), the evaporation term is inactive for supersaturated conditions with $S \geq 1$.

The derivatives of the mixing-ratios with respect to the coefficients are shown in Figure 2, whereas Figure 3 shows the derivatives with respect to the exponents. As the upper left panel in Figure 2 shows, the coefficient with the largest sensitivity for the cloud droplet mass q_c up to about $1000\,\mathrm{s}$ is the coefficient a_1 for autoconversion (red curve). The large sensitivity during the initial stage of the cloud evolution implies that the main loss of cloud droplet mass can be attributed to the autoconversion process, rendering the autoconversion rate a critical parameter. Given that autoconversion is the only process to produce rain drops out of the cloud droplets, this result may be anticipated. The negative value of the derivative $\frac{\partial q_c}{\partial a_1}$ indicates a decrease in q_c if the value of a_1 is increased by a small amount, i.e. a larger autoconversion rate results in a faster decrease of the cloud droplet mass. The transition region between $900\,\mathrm{s}$ and $1200\,\mathrm{s}$, where the influence of autoconversion decreases and accretion increases, is also clearly visible as a decreasing magnitude of $\frac{\partial q_c}{\partial a_1}$ and an increasing magnitude of the derivative $\frac{\partial q_c}{\partial a_2}$ of cloud droplet mass mixing-ratio with respect to the accretion parameter a_2 (blue curve).

Inspecting the upper right panel in Figure 2, we observe a positive derivative $\frac{\partial q_r}{\partial a_1}$ of the rain mixing-ratio with respect to the autoconversion coefficient with the same magnitude as $\frac{\partial q_c}{\partial a_1}$. This simply resembles the mass continuity, since a faster autoconversion implies a faster decrease in the cloud droplet mass q_c and an equally fast increase in the rain drop mass q_r. The same is true for the accretion, i.e. the derivatives $\frac{\partial q_c}{\partial a_2}$, $\frac{\partial q_r}{\partial a_2}$ (blue curves in the upper panels).

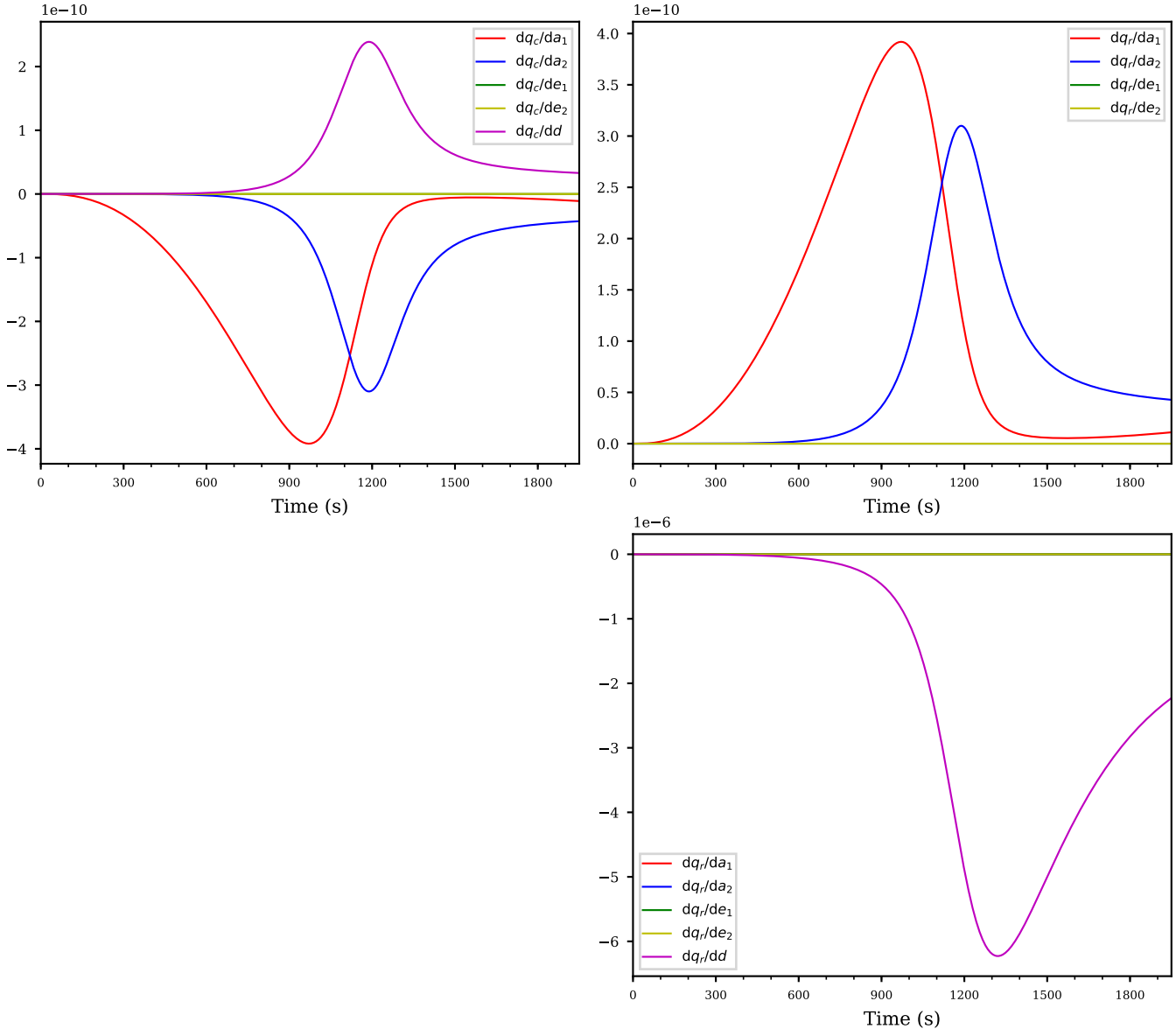

Figure 2. Temporal evolution of the derivatives of the cloud droplet mixing-ratio q_c (upper left panel) and the derivatives of the rain drop mixing-ratio q_r (lower panel) with respect to the coefficients. The upper right panel shows the derivatives of the rain drop mixing-ratio q_r with respect to the coefficients but without the derivative $\frac{\partial q_r}{\partial d}$ with respect to the sedimentation coefficient; the lower panel shows the derivatives of q_r including the derivative with respect to the sedimentation coefficient d (purple curve). All panels correspond to the first case of an ascending air parcel with no pre-existing cloud.

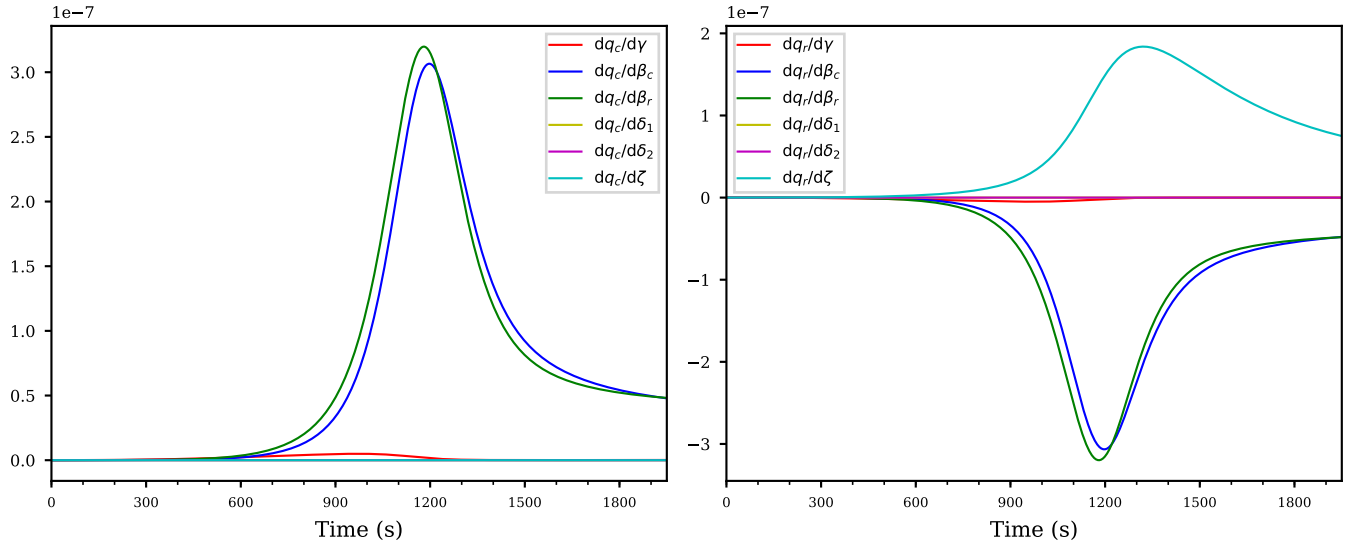

Figure 3. As in Figure 2, but for the derivatives of q_c (left panel) and q_r (right panel) with respect to the exponents.

The derivatives $\frac{\partial q_c}{\partial e_1}$, $\frac{\partial q_c}{\partial e_2}$ of the cloud droplet mixing-ratio with respect to the rain-evaporation rate coefficients e_1, e_2 as well as the derivatives $\frac{\partial q_r}{\partial e_1}$, $\frac{\partial q_r}{\partial e_2}$ of the rain mixing-ratio with respect to the same coefficients are identically zero, being consistent with the fact, that the evaporation term is inactive within a supersaturated cloud parcel, see Equation (12d). However, the purple curve in the upper left panel, representing the derivative $\frac{\partial q_c}{\partial d}$ of the cloud droplet mixing-ratio with respect to the sedimentation coefficient, is not identically zero although the sedimentation term is absent in Equation (12a) for the cloud droplet mixing-ratio q_c. This is an example of an indirect sensitivity of q_c to this coefficient: Altering the sedimentation coefficient modifies the sedimentation rate which obviously directly affects the rain mixing-ratio q_r. This in turn feeds back to the cloud droplet mixing-ratio q_c since the rain mixing-ratio q_r enters Equation (12a) through the accretion term. We conclude that the AD methodology is able to detect such indirect effects. Moreover, as may be concluded from the left panel in Figure 2, this indirect sensitivity could easily be masked due to the comparable magnitude of the positive sensitivity $\frac{\partial q_c}{\partial d}$ (purple curve) and the negative sensitivity $\frac{\partial q_c}{\partial a_2}$ (blue curve).

The lower panel in Figure 2 shows the derivatives of q_r with respect to all coefficients, in particular also with respect to the sedimentation parameter d (purple curve; this curve was not included in the upper right panel). From this figure it is evident, that q_r is most sensitive to the sedimentation coefficient. Comparing the different scalings of the ordinate in the upper right and the lower panel corroborates this result. To summarize, at the beginning of this simulation, the most sensitive parameter is the autoconversion rate to create rain drop mass. Towards the end of the simulation, enough rain drop mass formed and the sedimentation becomes more important, actually much more important than the autoconversion or the accretion since the derivatives $\frac{\partial q_r}{\partial a_1}$, $\frac{\partial q_r}{\partial a_2}$ and $\frac{\partial q_r}{\partial d}$ differ by almost four orders of magnitude.

Figure 3 shows the derivatives of the mixing-ratios q_c, q_r with respect to the exponents. For both mixing-ratios, obviously the exponents β_c, β_r from the accretion process are most sensitive (blue and green curves).

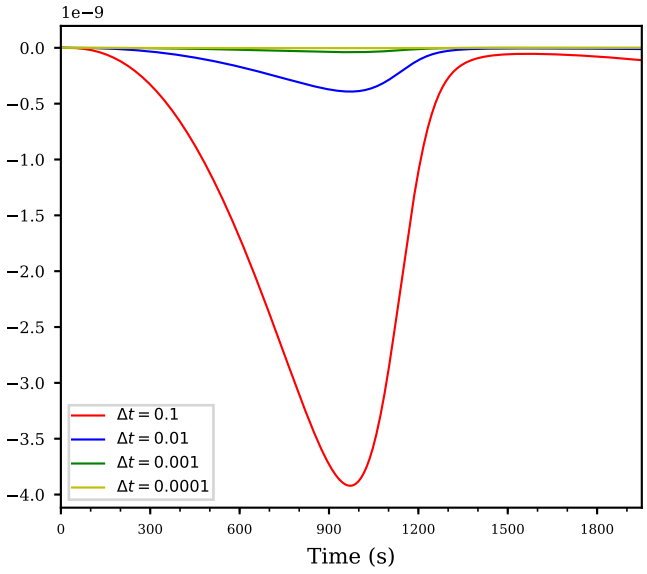

Figure 4. Temporal evolution of the derivative $\frac{\partial q_c}{\partial a_1}$ using the timesteps $\Delta t \in \{0.1\,\mathrm{s}, 0.01\,\mathrm{s}, 0.001\,\mathrm{s}, 0.0001\,\mathrm{s}\}$ for the computation of the same case as in Figure 2.

Note that the sign of the curves is counter intuitive, because, e.g., positive values of the blue and green curves imply slower accretion after increasing the values of those exponents by a small amount. This behaviour is easily explained with the values of both mixing-ratios being smaller than unity: Increasing the exponent in the expression $a_2 q_c^{\beta_c} q_r^{\beta_r}$ with $0 \leq q_c, q_r < 1$ leads to decreased values of the expression $a_2 q_c^{\beta_c} q_r^{\beta_r}$ and consequently to a slower accretion process. It is important to keep such a behaviour in mind in interpreting the derivatives.

The second most sensitive exponent for the rain mixing-ratio is given by the exponent ζ from the sedimentation process, but only after enough rain drop mass has formed after about $1000\,\mathrm{s}$. The magnitude of the derivatives with respect to ζ and the accretion exponents β_c, β_r are comparable and of opposite sign. Therefore, the influence of increasing these exponents simultaneously may cancel out.

Observe that the derivatives of both mixing-ratios with respect to the exponents δ_1, δ_2 in Figure 3 are exactly zero, again resembling the in-activeness of the evaporation term in Equation (12d) within the supersaturated cloud parcel.

As was already indicated, AD computes the derivatives of the *implemented* code, which in our case involves a numerical method, rather than derivatives of the (unknown) continuous solution of the governing differential equation. This distinction is important for the interpretation of the computed sensitivities and we are now going to discuss the subtleties connected with them.

First of all, we outline the fact, that since the numerical discretization of the governing differential equation depends on the timestep Δt, the magnitude of all computed derivatives also depend on the timestep. Figure 4 illustrates this fact by showing the derivative $\frac{\partial q_c}{\partial a_1}$ of the cloud droplet mass on the parameter a_1 for autoconversion for several timesteps. In this figure, the

dependency on Δt becomes obvious, however, the shape of the curve does actually not change. Rescaling the curves in Figure 4 by multiplying the values of the blue curve by $\frac{0.1}{0.01} = 10$, the values of the green curve by $\frac{0.1}{0.001} = 100$, and the values of the yellow curve by $\frac{0.1}{0.0001} = 1000$, all rescaled curves do coincide with the red curve (not shown), representing the derivative computed with timestep $\Delta t = 0.1$. Although, this scaling with the timestep is quite intuitive due to the fact that if the timestep is larger, the simulation is done for a larger time interval and the sensitivities are expected to increase, one should be aware of this scaling when trying to interpret the magnitudes of the derivatives.

A second issue regarding the meaning of the computed sensitivities is addressed in Bischof and Eberhard (1999). There are two possibilities of computing sensitivities:

1. Apply AD on the implemented code, which numerically approximates the unknown solution of the governing differential equation, or

2. derive an analytical differential equation (called the sensitivity or variational equation as an analogue of the forward AD mode or the adjoint equation as an analogue of the reverse AD mode), describing how the sensitivities evolve with time (see, e.g., Sandu et al. (2003) or Walther (2007) for details). In this approach, AD may enter the computation of the derivatives of the right-hand side of the governing equation.

The essential difference between both possibilities is, that the first approach explicitly includes the numerical method in the computation of the sensitivities, while the second approach excludes any numerical method in the first place and instead creates an analytical differential equation, e.g. the sensitivity equation, for the description of the sensitivities and only in a second step approximates the exact solution of this equation using a numerical method.

In our study we choose the first option and include the numerical method in the differentiation, since this seems more appropriate if one is interested in how the sensitivities propagate through a given, already implemented numerical model. We use a one-step numerical method with constant timestep Δt to approximate the ordinary differential equation, hence the numerical solution y^{new} at the new time-level is connected to the old solution y^{old} at the old time-level by $y^{\mathrm{new}} = y^{\mathrm{old}} + \Delta t \cdot \Phi(y^{\mathrm{old}}, \eta, \Delta t)$, where η represents the parameter vector as in Equation (17) and Φ is the numerical method. Moreover, we compute the derivatives at each timestep separately, i.e. the approximation y^{old} is considered as independent of the parameters at the current time-level, hence $\frac{\partial y^{\mathrm{old}}}{\partial \eta} = 0$. Consequently, in our case the AD methodology factually computes the derivative

$$\begin{aligned}
\frac{\partial y^{\mathrm{new}}}{\partial \eta} &= \frac{\partial y^{\mathrm{old}}}{\partial \eta} + \Delta t \cdot \frac{\partial}{\partial \eta}\left[\Phi(y^{\mathrm{old}}, \eta, \Delta t)\right] \\
&= \frac{\partial y^{\mathrm{old}}}{\partial \eta} + \Delta t \cdot \left[\frac{\partial \Phi}{\partial y}(y^{\mathrm{old}}, \eta, \Delta t)\frac{\partial y^{\mathrm{old}}}{\partial \eta}\right. \\
&\qquad \left. + \frac{\partial \Phi}{\partial \eta}(y^{\mathrm{old}}, \eta, \Delta t) + \frac{\partial \Phi}{\partial \Delta t}(y^{\mathrm{old}}, \eta, \Delta t)\frac{\partial \Delta t}{\partial \eta}\right] \\
&= \Delta t \cdot \frac{\partial \Phi}{\partial \eta}(y^{\mathrm{old}}, \eta, \Delta t),
\end{aligned} \tag{19}$$

showing once again the observed dependency of the computed sensitivities on the timestep. In addition, (19) shows explicitly that the numerical method Φ is included in the differentiation, thus the computed sensitivities depend on the numerical method.

Δt	$\frac{\partial q_c}{\partial a_1} / \frac{\partial q_c}{\partial a_2}$	$\frac{\partial q_c}{\partial a_1} / \frac{\partial q_c}{\partial \gamma}$	$\frac{\partial q_c}{\partial a_1} / \frac{\partial q_c}{\partial \beta_c}$	$\frac{\partial q_c}{\partial a_1} / \frac{\partial q_r}{\partial d}$
$0.1\,\mathrm{s}$	4.0228	$-7.8577 \cdot 10^{-2}$	$-4.3450 \cdot 10^{-3}$	$3.6168 \cdot 10^{-4}$
$0.01\,\mathrm{s}$	4.0209	$-7.8577 \cdot 10^{-2}$	$-4.3429 \cdot 10^{-3}$	$3.6148 \cdot 10^{-4}$
$0.001\,\mathrm{s}$	4.0207	$-7.8577 \cdot 10^{-2}$	$-4.3427 \cdot 10^{-3}$	$3.6146 \cdot 10^{-4}$
$0.0001\,\mathrm{s}$	4.0207	$-7.8577 \cdot 10^{-2}$	$-4.3427 \cdot 10^{-3}$	$3.6146 \cdot 10^{-4}$

Table 2. Ratios of the derivatives of the mixing-ratios with respect to different parameters at $t = 1000\,\mathrm{s}$, computed for several timesteps Δt. All numbers are taken from the computations of the first case of an ascending air parcel with no pre-existing cloud. The numbers are rounded to four digits.

Note that we employed a constant timestep which, according to the results in Bischof and Eberhard (1999), ensures that the computed sensitivities are indeed approximations of the sensitivities of the unknown, exact solution. If we had used a numerical method with adaptive timestepping, this is not necessarily true, since in this case the current timestep depends on the solution which in turn depends on the parameters η, implying $\frac{\partial \Delta t}{\partial \eta} \neq 0$ in general. Again, if one is interested in an approximate computation of the sensitivities of the exact solution, a correction is needed for this non-zero contribution of $\frac{\partial \Delta t}{\partial \eta} \neq 0$, see Bischof and Eberhard (1999). However, if one is interested in the sensitivities of the *implemented* code, no correction is needed.

Finally, it is not obvious that the computed sensitivities, which depend on the timestep, converge to the sensitivities of the exact but unknown solution in the limit $\Delta t \to 0$. In our case, since we do rely on an explicit Runge-Kutta method of order 4, the desired convergence is established in Walther (2007). An extension to more general Runge-Kutta method for the adjoint equation is presented in Sandu (2006).

Although the magnitude of the computed derivatives depend on the timestep of the numerical method, the *relative* magnitudes of the individual derivatives are independent of the timestep. Table 2 highlights this observation by comparing the ratios of some derivatives of the mixing-ratios at $t = 1000\,\mathrm{s}$, computed with several timesteps Δt. The ratios of the derivatives shown in Table 2 are indeed approximately constant (except for effects of a coarse time resolution), implying that these ratios are indeed independent of the timestep. Therefore, the derivative with respect to the most sensitive parameter will show the largest magnitude compared to the derivatives with respect to the other parameters, regardless of the chosen timestep of the numerical method involved. This retains the possibility to use the computed derivatives to identify the most sensitive parameters of the cloud scheme.

4.2 Cloud Evaporation in Downdraft

As the second example, we consider a downdraft with velocity $w = -1\,\mathrm{m\,s^{-1}}$, the initial conditions

$$(q_c(0), q_r(0)) = \left(2 \cdot 10^{-4}\,\mathrm{kg\,kg^{-1}}, 10^{-4}\,\mathrm{kg\,kg^{-1}}\right) \tag{20}$$

for the mixing-ratios and $S(0) = 1.01$ for the saturation ratio, representing an initial supersaturation of $1\,\%$.

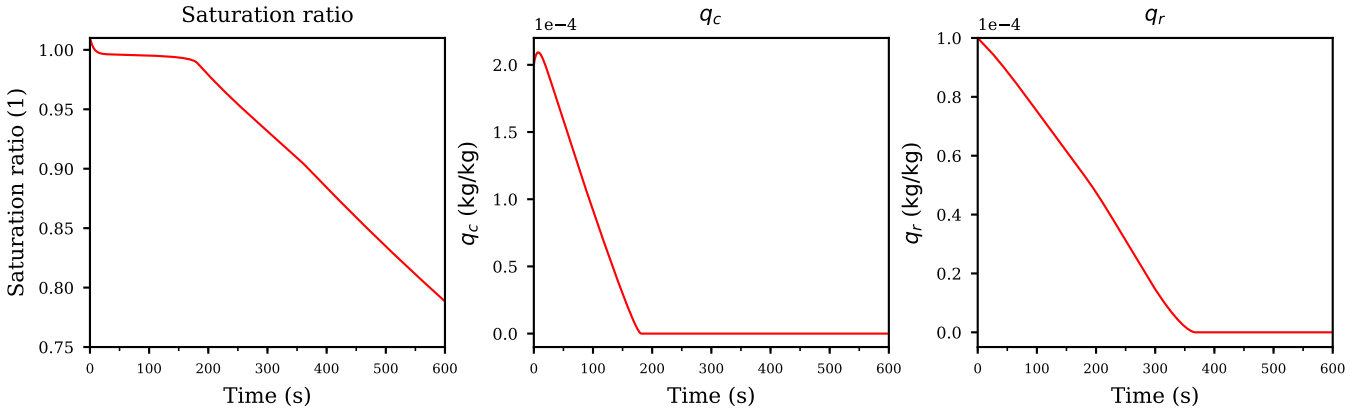

Figure 5. Temporal evolution of the saturation ratio S (left panel), the cloud droplet mixing-ratio q_c (middle panel), and the rain drop mixing-ratio q_r (right panel) for the descending air parcel with evaporating cloud.

The temporal evolution of the saturation ratio and the mixing-ratios are shown in Figure 5. The downward vertical motion of the air parcel causes the saturation ratio to decrease due to the adiabatic heating. However, until the complete evaporation of the cloud droplets at about $175\,\mathrm{s}$ (see the middle panel in Figure 5), the release of water vapor of the evaporating cloud droplets counteracts the decrease of the saturation ratio and keeps the air parcel only slightly subsaturated (left panel in Figure 5). After roughly $175\,\mathrm{s}$, the saturation ratio decreases continuously, resulting in a substantially subsaturated air parcel. Consequently, the available rain drops do not only sediment out of the air parcel but also evaporate due to the subsaturation (right panel in Figure 5). However, the release of water vapor of the evaporating rain drops is seemingly not able to counteract the decrease of the saturation ratio as was the case for the evaporating cloud droplets at the beginning of the simulation, but the precise sensitivities of rain drop evaporation and sedimentation cannot be deduced from the temporal evolution of the mass mixing-ratio q_r.

The temporal evolutions of the derivatives of the mixing-ratios with respect to the coefficients are shown in Figure 6. Contrary to the first case from Section 4.1, the air parcel rapidly becomes subsaturated with $S \leq 1$ and the evaporation process in Equation (12d) is now active, hence no derivative is identically zero. Inspecting Figure 6, the most sensitive coefficient for both mixing-ratios is e_2 (yellow curve), corresponding to the ventilation coefficient within the formulation of the evaporation process of the rain drops, see Equation (12d). This result may be anticipated regarding the rain drop mass mixing-ratio q_r, because the air parcel is subsaturated and the evaporation process directly affects the rain drop mixing-ratio. However, we observe also a large sensitivity of the same parameter on the cloud droplet mixing-ratio q_c. This feedback is, again, an indirect sensitivity originating from the accretion process: If the rain drop mass decreases faster due to a slight increase of the coefficient e_2, the accretion gets slower and therefore less cloud droplets get collected by the falling rain drops. Consequently, the decrease of cloud droplet mixing-ratio is diminished.

The right panel in Figure 6 also shows that the second most sensitive coefficient for q_r is given by the sedimentation rate coefficient. This observation also answers the question which process is more sensitive to changes in its rate coefficient for the decrease of rain drop mass, seen in the right panel in Figure 5. Due to the larger absolute values of $\frac{\partial q_r}{\partial e_2}$ compared to $\frac{\partial q_r}{\partial d}$

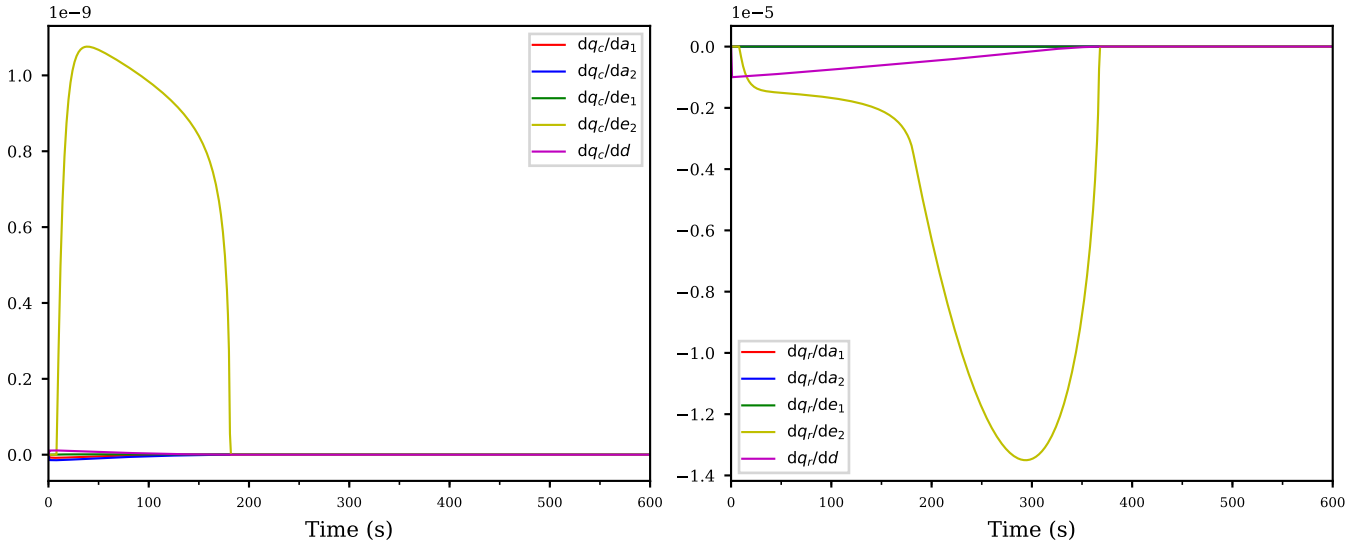

Figure 6. Temporal evolution of the derivatives of the cloud droplet mixing-ratio q_c (left panel) and the rain drop mixing-ratio q_r (right panel) with respect to the coefficients for the descending air parcel with evaporating cloud.

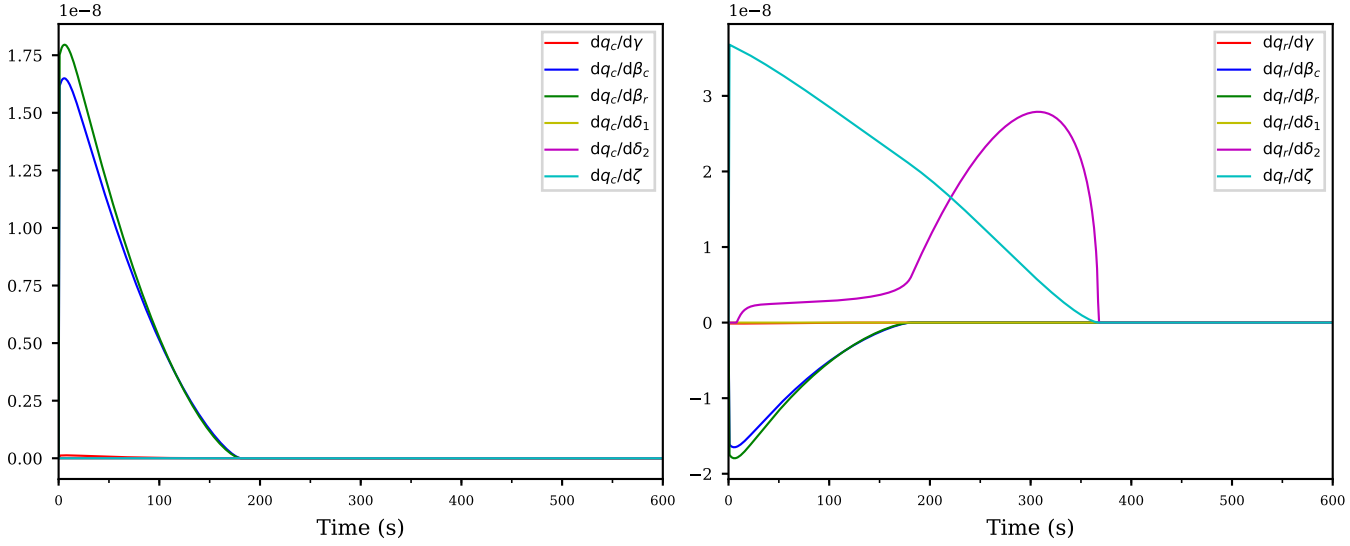

Figure 7. As in Figure 6, but for the derivatives of q_c (left panel) and q_r (right panel) with respect to the exponents.

(yellow and purple curves in the right panel in Figure 6), a slight change in the evaporation rate coefficient e_2 will result in larger responses than a change in the sedimentation rate.

Although not visible in the left panel in Figure 6, the derivatives with respect to the coefficients a_1, a_2, d are all of about the same magnitude, while the sensitivity to the second evaporation coefficient e_1 is significantly smaller.

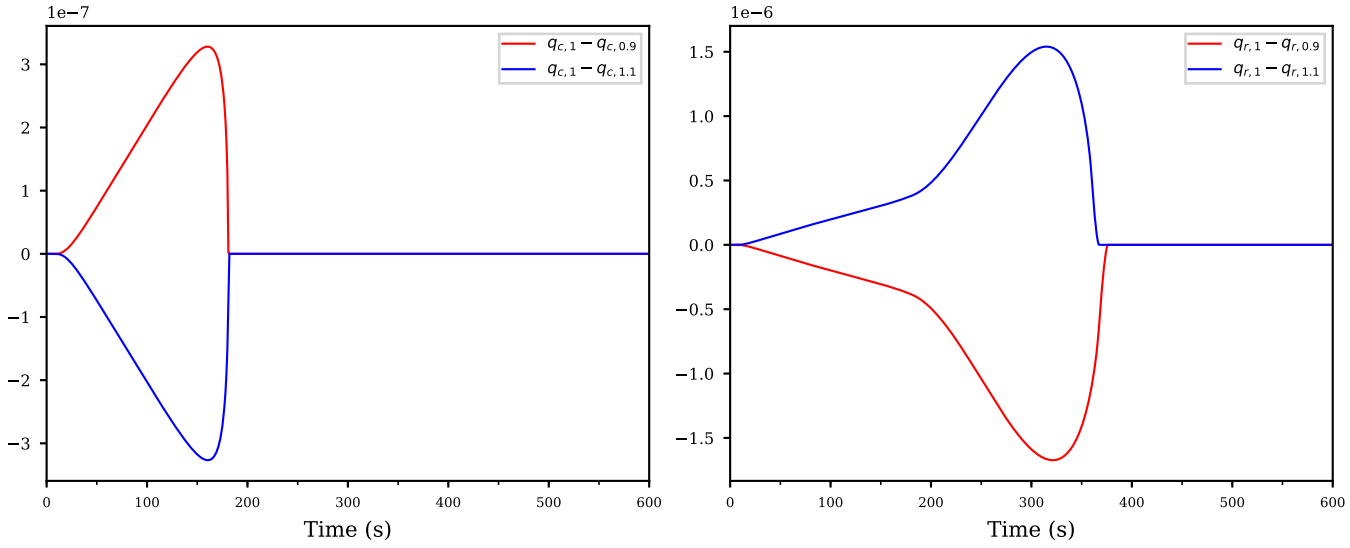

Figure 8. Difference between the reference run with unperturbed coefficient e_2, denoted as $q_{x,1}$ for $x \in \{c, r\}$ and using the perturbed second evaporation coefficient $s \cdot e_2$ with $s \in \{0.9, 1.1\}$, denoted as $q_{x,0.9}$ or $q_{x,1.1}$, for the cloud droplet mixing-ratio (left panel) and the rain mixing-ratio (right panel), respectively.

Inspecting Figure 7, illustrating the derivatives of both mixing-ratios to the exponents, the most sensitive exponents for the cloud droplet mixing-ratio q_c are again the exponents β_c, β_r corresponding to the accretion process (blue and green curve in the left panel). For the rain drop mixing-ratio q_r (right panel), the most sensitive exponent changes from the sedimentation exponent ζ (cyan curve) at the beginning to the exponent δ_2, occurring in the second term $e_2 q_r^{\delta_2}$ of the evaporation term, being consistent with the large sensitivity of the corresponding rate coefficient e_2, see the yellow curve in the left panel of Figure 6.

To summarize the second example: The AD methodology pinpoints the second summand $e_2 q_r^{\delta_2}$ of the evaporation term together with the exponents β_c, β_r of the accretion process to introduce the largest sensitivity in the model results. Although one could find the same sensitivities using classical sensitivity studies instead of AD, the AD methodology provides an immediate hint on, e.g., the sensitive coefficient e_2 for both mixing-ratios, see Figure 6, without having to carry out multiple model runs, where one perturbs each coefficient of the cloud scheme separately, one after the other. Moreover, the sensitivity of the cloud droplet mixing-ratio q_c to e_2 is indirect, rendering it difficult to identify this sensitivity directly using the ensemble approach, in particular because the governing Equation (12a) provides no hint due to the absence of coefficient e_2. Given that even the simple cloud scheme (12) already contains five rate coefficients, perturbing each coefficient within a separate model run results in a significant total number of runs.

After the identification of the most sensitive parameters using AD, one can carry out further model runs, targeted at the parameters which were identified beforehand. Figure 8 illustrates a possible further analysis step. It shows the difference between an unperturbed run, denoted by $q_{x,1}$ with $x \in \{c, r\}$ and two further runs with perturbed rate coefficient $s \cdot e_2$ instead of e_2, where $s \in \{0.9, 1.1\}$ is a scaling parameter. Observe that the signal is consistent with the derivative, computed by AD,

in Figure 6: The derivative $\frac{\partial q_r}{\partial e_2}$ is negative, hence a slight increase of the coefficient should result in a smaller rain mixing-ratio q_r and, consequently, the difference $q_{r,1} - q_{r,1.1}$ should be positive. Similarly, the derivative $\frac{\partial q_c}{\partial e_2}$ is positive, hence a slight increase of the coefficient should result in negative values for the difference $q_{c,1} - q_{c,1.1}$. Figure 8 shows exactly these tendencies (blue curves). The red curves show the resulting differences using the scaling parameter $s = 0.9$; note that the curves are not symmetric to each other.

4.3 Dependency on the Model Trajectory

After having discussed both exemplary cases individually, we now point at another important aspect of the AD methodology. Given that AD was applied to the exactly same computational code, a comparison between, e.g., the derivatives of the cloud droplet mixing-ratio q_c with respect to the rate coefficients (see the upper left panel in Figure 2 and the left panel in Figure 6) reveals that the corresponding curves are not equal to each other, but show a significant different behaviour. The only difference between both examples were the values of the initial conditions. Consequently, the model trajectories between both runs evolved differently despite the fact that the computational code was unchanged. This observation is a crucial aspect of the AD methodology since it underlines that AD computes the derivative of the model, i.e. the computational code, along the particular model trajectory, rather than providing the derivatives for each possible state of the model. Therefore, the AD approach can provide the desired sensitivities for the particular evolution of the model state, posing the same limitations as the computation of the derivatives using the aforementioned finite difference approach.

5 Conclusion

In this study, we presented and applied the technique of algorithmic differentiation (AD) in the context of cloud schemes, representing an important example of a subgrid parameterization of numerical models within the atmospheric sciences. In the literature, many different cloud schemes are suggested since at the moment, a universal governing equation for the full description of a cloud is not available (in contrast to the Navier-Stokes equation for the description of a non-reacting flow), making it not possible to derive cloud schemes from a common universal basis. As a consequence, many ad-hoc assumptions are made within the formulations of the cloud processes typically leading to the introduction of uncertain parameters.

We propose the use of algorithmic differentiation in the development of cloud schemes in order to identify the most sensitive parameters in the adopted formulation along the simulated solution trajectory. The AD methodology is based on the observation that each computer code is a large composition of only a few differentiable elemental operations, hence, by the chain rule, the code itself is differentiable apart from critical points, where the elemental functions are not differentiable. Since the derivatives of the elemental operations are known, the full computational code can be differentiated in a (semi-)automatic fashion. Moreover, the resulting derivatives are accurate to machine precision because the AD approach merely evaluates the exact derivative.

In the context of sensitivity studies, the AD approach yields the desired sensitivities of the parameters on the result of the computation requiring only a constant additional computational effort, see Griewank and Walther (2008). The forward mode

of AD roughly doubles the number of code instructions since each statement is complemented with its derivative, hence also roughly doubles the code execution time for each run. Note that the forward mode needs the specification of the desired direction for the directional derivative in advance, thus it only computes a single directional derivative per run. This may rapidly become very computational intensive if there are many input parameters or one is interested in the full gradient. In this case one should use the vector forward mode which leads to a significant reduction in complexity. In contrast, the reverse mode introduces more overhead than the forward mode for a single run, but the amount is independent of the number of input variables, among which are also the parameters to be investigated. Therefore, the reverse AD mode has the ability to outperform the forward mode in case of many input variables but only a small number of output variables because only a single run of the reverse mode is required to establish the derivatives with respect to each input variable, whereas the forward mode requires as many runs as the number of input variables. The fact that AD introduces only a constant computational overhead is especially useful if the number of parameters is comparably high, since establishing an ensemble to investigate the sensitivity of the parameters quickly results in a high number of model runs. Moreover, using an ensemble of model runs to study parameter sensitivities additionally involves a thorough post-processing of all model output. In contrast, the AD approach clearly identifies the parameters with high sensitivity, regardless if the sensitivity is direct or indirect, and allows to focus a further post-processing on only the relevant parameters.

AD helps in computing the derivatives, but one has to keep in mind the subtleties: Using AD as advocated in this study implies that the computed derivatives involve the differentiation of the numerical method used to approximate the solution of the differential equation (12), (13). Consequently, the derivatives depend on the timestep and may change if the numerical method is replaced by another method. Nevertheless, the computed sensitivities resemble the actual sensitivities of the *implemented* code and comprise the correct result to tackle the question how uncertainties propagate through a given numerical model as, e.g., a weather forecasting model.

We emphasize that the technique of AD is not restricted to a specific programming language nor to the analysis of cloud schemes. It is a generic technique which may help in the development of any (subgrid) scheme for a (geophysical) numerical model by providing informations about the sensitivities of the involved parameters.

In adopting a cloud scheme or any subgrid scheme, the question of how the inherent uncertainties of the scheme influence the (numerical) solution of the model arises. In the context of the topic of this study, an example of the influences of a single parameter within a typical cloud scheme on the overall cloud development is discussed in Igel and van den Heever (2017a, b, c). Answering the question of how uncertainties propagate within a given model is a highly non-trivial task. As already indicated in Section 1, methods from "Uncertainty Quantification" allow to assess this propagation (e.g., Sullivan, 2015; Le Maître and Knio, 2010), but taking many parameters simultaneously into account is challenging and computationally expensive. Algorithmic differentiation allows to first identify the parameters which influence the result of a given parameterization at most, e.g. a cloud scheme, and limit the more extensive investigation to only these parameters; see Chertock et al. (2019) for an example of such an investigation. However, as AD by design computes the derivatives along the numerical solution trajectory, one might not detect all possible sensitivities, but at least the most sensitive parameters in "typical" situations.

Code availability. The code for the cloud model together with all scripts needed to reproduce the results in this study is available at https://doi.org/10.5281/zenodo.3461483. The algorithmic differentiation tool "CoDiPack" may be found online at https://github.com/scicompkl/codipack and the version 1.8.1, which is used in this study, is available at https://doi.org/10.5281/zenodo.3460682.

Author contributions. MB developed the cloud model code and carried out the numerical experiments under the supervision of AB and PS. MS developed the algorithmic differentiation tool "CoDiPack" under the supervision of NRG. MB and MS prepared the manuscript with reviews from NRG, AB and PS.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgements. The authors thank the anonymous referees for their valuable comments and suggestions, which helped to considerably improve the manuscript. The authors acknowledge support of the "Deutsche Forschungsgemeinschaft" (DFG) within the project "Enabling Performance Engineering in Hesse and Rhineland-Palatinate" (grant number 320898076). Manuel Baumgartner, Peter Spichtinger and André Brinkmann acknowledge support by the DFG within the Transregional Collaborative Research Centre TRR165 "Waves to Weather" (www.wavestoweather.de), projects B7 and Z2.

References

Albring, T., Sagebaum, M., and Gauger, N. R.: Efficient Aerodynamic Design using the Discrete Adjoint Method in SU2, AIAA 2016-3518, 2016.

Asai, T.: A Numerical Study of the Air-Mass Transformation over the Japan Sea in Winter, Journal of the Meteorological Society of Japan. Ser. II, 43, 1–15, 1965.

Baumgartner, M.: Algorithmic Differentiation for Cloud Schemes using CoDiPack (v1.8.1), https://doi.org/10.5281/zenodo.3461483, 2019.

Belikov, D. A., Maksyutov, S., Yaremchuk, A., Ganshin, A., Kaminski, T., Blessing, S., Sasakawa, M., Gomez-Pelaez, A. J., and Starchenko, A.: Adjoint of the global Eulerian–Lagrangian coupled atmospheric transport model (A-GELCA v1.0): development and validation, Geoscientific Model Development, 9, 749–764, https://doi.org/10.5194/gmd-9-749-2016, https://www.geosci-model-dev.net/9/749/2016/, 2016.

Bischof, C. H. and Eberhard, P.: Automatic differentiation of numerical integration algorithms, Math. Comp., 68, 717–731, https://doi.org/10.1090/S0025-5718-99-01027-3, https://doi.org/10.1090/S0025-5718-99-01027-3, 1999.

Bischof, C. H., Khademi, P., Mauer, A., and Carle, A.: Adifor 2.0: automatic differentiation of Fortran 77 programs, IEEE Computational Science and Engineering, 3, 18–32, https://doi.org/10.1109/99.537089, 1996a.

Bischof, C. H., Pusch, G. D., and Knoesel, R.: Sensitivity analysis of the MM5 weather model using automatic differentiation, Computers in Physics, 10, 605–612, https://doi.org/10.1063/1.168585, https://aip.scitation.org/doi/abs/10.1063/1.168585, 1996b.

Blessing, S., Kaminski, T., Lunkeit, F., Matei, I., Giering, R., Köhl, A., Scholze, M., Herrmann, P., Fraedrich, K., and Stammer, D.: Testing variational estimation of process parameters and initial conditions of an earth system model, Tellus A: Dynamic Meteorology and Oceanography, 66, 22 606, https://doi.org/10.3402/tellusa.v66.22606, https://doi.org/10.3402/tellusa.v66.22606, 2014.

Chertock, A., Kurganov, A., Lukáčová-Medvid'ová, M., Spichtinger, P., and Wiebe, B.: Stochastic Galerkin Method for Cloud Simulation, Mathematics of Climate and Weather Forecasting, revised version, 2019.

Cotton, W. R., Bryan, G. H., and van den Heever, S. C.: Storm and Cloud Dynamics, Academic Press, second edition edn., 2010.

Devenish, B. J., Bartello, P., Brenguier, J.-L., Collins, L. R., Grabowski, W. W., IJzermans, R. H. A., Malinowski, S. P., Reeks, M. W., Vassilicos, J. C., Wang, L.-P., and Warhaft, Z.: Droplet growth in warm turbulent clouds, Quarterly Journal of the Royal Meteorological Society, 138, 1401–1429, https://doi.org/10.1002/qj.1897, http://dx.doi.org/10.1002/qj.1897, 2012.

Doms, G., Förstner, J., Heise, E., Herzog, H.-J., Mironow, D., Raschendorfer, M., Reinhardt, T., Ritter, B., Schrodin, R., Schulz, J.-P., and Vogel, G.: A Description of the Nonhydrostatic Regional COSMO Model. Part II: Physical Parameterization, 2011.

ECMWF: IFS DOCUMENTATION – Cy43r3. Part IV: Physical Processes, 2017.

Elizondo, D., Cappelaere, B., and Faure, C.: Automatic versus manual model differentiation to compute sensitivities and solve non-linear inverse problems, Computers & Geosciences, 28, 309 – 326, https://doi.org/https://doi.org/10.1016/S0098-3004(01)00048-6, http://www.sciencedirect.com/science/article/pii/S0098300401000486, 2002.

Grabowski, W. W. and Wang, L.-P.: Growth of Cloud Droplets in a Turbulent Environment, Annual Review of Fluid Mechanics, 45, 293–324, https://doi.org/10.1146/annurev-fluid-011212-140750, http://dx.doi.org/10.1146/annurev-fluid-011212-140750, 2013.

Griewank, A. and Walther, A.: Evaluating Derivatives, second edition, SIAM, 2008.

Griewank, A., Kulshreshtha, K., and Walther, A.: On the numerical stability of algorithmic differentiation, Computing, 94, 125–149, https://doi.org/10.1007/s00607-011-0162-z, https://doi.org/10.1007/s00607-011-0162-z, 2012.

Grossmann, C. and Roos, H.-G.: Numerical Treatment of Partial Differential Equations, Universitext, Springer-Verlag, Berlin Heidelberg, 2007.

Hairer, E., Nørsett, S. P., and Wanner, G.: Solving Ordinary Differential Equations I, vol. 8 of *Springer Series in Computational Mathematics*, Springer-Verlag, Berlin Heidelberg, second revised edition edn., https://doi.org/10.1007/978-3-540-78862-1, 1993.

Hascoët, L. and Pascual, V.: The Tapenade Automatic Differentiation tool: Principles, model, and specification, ACM TOMS, 39, http://dx.doi.org/10.1145/2450153.2450158, 2013.

Hück, A., Bischof, C. H., and Utke, J.: Checking C++ codes for compatibility with operator overloading, in: 2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM), pp. 91–100, https://doi.org/10.1109/SCAM.2015.7335405, 2015.

Igel, A. L. and van den Heever, S. C.: The Importance of the Shape of Cloud Droplet Size Distributions in Shallow Cumulus Clouds. Part I: Bin Microphysics Simulations, Journal of the Atmospheric Sciences, 74, 249–258, https://doi.org/10.1175/JAS-D-15-0382.1, https://doi.org/10.1175/JAS-D-15-0382.1, 2017a.

Igel, A. L. and van den Heever, S. C.: The Importance of the Shape of Cloud Droplet Size Distributions in Shallow Cumulus Clouds. Part II: Bulk Microphysics Simulations, Journal of the Atmospheric Sciences, 74, 259–273, https://doi.org/10.1175/JAS-D-15-0383.1, https://doi.org/10.1175/JAS-D-15-0383.1, 2017b.

Igel, A. L. and van den Heever, S. C.: The role of the gamma function shape parameter in determining differences between condensation rates in bin and bulk microphysics schemes, Atmospheric Chemistry and Physics, 17, 4599–4609, https://doi.org/10.5194/acp-17-4599-2017, https://www.atmos-chem-phys.net/17/4599/2017/, 2017c.

Igel, A. L., Igel, M. R., and van den Heever, S. C.: Make It a Double? Sobering Results from Simulations Using Single-Moment Microphysics Schemes, Journal of the Atmospheric Sciences, 72, 910–925, https://doi.org/10.1175/JAS-D-14-0107.1, https://doi.org/10.1175/JAS-D-14-0107.1, 2015.

Kalnay, E.: Atmospheric modeling, data assimilation and predictability, Cambridge University Press, 2003.

Kaminski, T., Heimann, M., and Giering, R.: A coarse grid three-dimensional global inverse model of the atmospheric transport: 1. Adjoint model and Jacobian matrix, Journal of Geophysical Research: Atmospheres, 104, 18 535–18 553, https://doi.org/10.1029/1999JD900147, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/1999JD900147, 1999.

Kessler, E.: On the Distribution and Continuity of Water Substance in Atmospheric Circulations, pp. 1–84, American Meteorological Society, Boston, MA, https://doi.org/10.1007/978-1-935704-36-2_1, https://doi.org/10.1007/978-1-935704-36-2_1, 1969.

Khain, A. P., Ovtchinnikov, M., Pinsky, M., Pokrovsky, A., and Krugliak, H.: Notes on the state-of-the-art numerical modeling of cloud microphysics, Atmospheric Research, 55, 159 – 224, https://doi.org/http://dx.doi.org/10.1016/S0169-8095(00)00064-8, http://www.sciencedirect.com/science/article/pii/S0169809500000648, 2000.

Khain, A. P., Beheng, K. D., Heymsfield, A. J., Korolev, A. V., Krichak, S. O., Levin, Z., Pinsky, M., Phillips, V., Prabhakaran, T., Teller, A., van den Heever, S. C., and Yano, J.-I.: Representation of microphysical processes in cloud-resolving models: Spectral (bin) microphysics versus bulk parameterization, Reviews of Geophysics, 53, 247–322, https://doi.org/10.1002/2014RG000468, http://dx.doi.org/10.1002/2014RG000468, 2015.

Khairoutdinov, M. and Kogan, Y.: A New Cloud Physics Parameterization in a Large-Eddy Simulation Model of Marine Stratocumulus, Monthly Weather Review, 128, 229–243, https://doi.org/10.1175/1520-0493(2000)128<0229:ANCPPI>2.0.CO;2, http://dx.doi.org/10.1175/1520-0493(2000)128<0229:ANCPPI>2.0.CO;2, 2000.

Kogan, Y. L. and Martin, W. J.: Parameterization of Bulk Condensation in Numerical Cloud Models, Journal of the Atmospheric Sciences, 51, 1728–1739, https://doi.org/10.1175/1520-0469(1994)051<1728:POBCIN>2.0.CO;2, http://dx.doi.org/10.1175/1520-0469(1994)051<1728:POBCIN>2.0.CO;2, 1994.

Köhler, H.: The nucleus in and the growth of hygroscopic droplets, Trans. Faraday Soc., 32, 1152–1161, https://doi.org/10.1039/TF9363201152, http://dx.doi.org/10.1039/TF9363201152, 1936.

Lamb, D. and Verlinde, J.: Physics and Chemistry of Clouds, Cambridge University Press, Cambridge, 2011.

Langlois, W. E.: A rapidly convergent procedure for computing large-scale condensation in a dynamical weather model, Tellus, 25, 86–87, https://doi.org/10.1111/j.2153-3490.1973.tb01598.x, http://dx.doi.org/10.1111/j.2153-3490.1973.tb01598.x, 1973.

Le Dimet, F.-X., Navon, I. M., and Daescu, D. N.: Second-Order Information in Data Assimilation, Monthly Weather Review, 130, 629–648, https://doi.org/10.1175/1520-0493(2002)130<0629:SOIIDA>2.0.CO;2, https://doi.org/10.1175/1520-0493(2002)130<0629:SOIIDA>2.0.CO;2, 2002.

Le Maître, O. and Knio, O. M.: Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics, Scientific Computation, Springer Science & Business Media, https://doi.org/10.1007/978-90-481-3520-2, 2010.

Maxwell, J. C.: Diffusion, reprinted in W.D. Niven (Ed.), The Scientific Papers of James Clerk Maxwell, 2, 625–645, 1877.

McDonald, J. E.: The saturation adjustment in numerical modelling of fog, Journal of the Atmospheric Sciences, 20, 476–478, 1963.

Neidinger, R. D.: Introduction to Automatic Differentiation and MATLAB Object-Oriented Programming, SIAM Review, 52, 545 – 563, https://doi.org/10.1137/080743627, 2010.

Orlanski, I.: A Rational Subdivision of Scales for Atmospheric Processes, Bulletin of the American Meteorological Society, 56, 527–530, www.jstor.org/stable/26216020, 1975.

Porz, N., Hanke, M., Baumgartner, M., and Spichtinger, P.: A model for warm clouds with implicit droplet activation, avoiding saturation adjustment, Mathematics of Climate and Weather Forecasting, 4, 50–78, https://doi.org/10.1515/mcwf-2018-0003, https://doi.org/10.1515/mcwf-2018-0003, 2018.

Rauser, F., Riehme, J., Leppkes, K., Korn, P., and Naumann, U.: On the use of discrete adjoints in goal error estimation for shallow water equations, Procedia Computer Science, 1, 107 – 115, https://doi.org/https://doi.org/10.1016/j.procs.2010.04.013, http://www.sciencedirect.com/science/article/pii/S1877050910000141, iCCS 2010, 2010.

Rogers, R. and Yau, M.: A Short Course in Cloud Physics, International Series in Natural Philosophy, Butterworth-Heinemann, third edition edn., 1989.

Rosemeier, J., Baumgartner, M., and Spichtinger, P.: Intercomparison of Warm-Rain Bulk Microphysics Schemes using Asymptotics, Mathematics of Climate and Weather Forecasting, 4, 104–124, https://doi.org/10.1515/mcwf-2018-0005, https://doi.org/10.1515/mcwf-2018-0005, 2018.

Sagebaum, M., Albring, T., and Gauger, N. R.: High-Performance Derivative Computations using CoDiPack, arXiv preprint arXiv:1709.07229, 2017a.

Sagebaum, M., Özkaya, E., Gauger, N. R., Backhaus, J., Frey, C., Mann, S., and Nagel, M.: Efficient Algorithmic Differentiation Techniques for Turbo-machinery Design, AIAA 2017-3998, 2017b.

Sagebaum, M., Albring, T., Demidov, D., Möller, M., van der Weide, E., and Lam, M.: SciCompKL/CoDiPack: Version 1.8.1, https://doi.org/10.5281/zenodo.3460682, 2019.

Sandu, A.: On the Properties of Runge-Kutta Discrete Adjoints, in: Computational Science – ICCS 2006, edited by Alexandrov, V. N., van Albada, G. D., Sloot, P. M. A., and Dongarra, J., pp. 550–557, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

Sandu, A., Daescu, D. N., and Carmichael, G. R.: Direct and adjoint sensitivity analysis of chemical kinetic systems with KPP: Part I—theory and software tools, Atmospheric Environment, 37, 5083 – 5096, https://doi.org/https://doi.org/10.1016/j.atmosenv.2003.08.019, http://www.sciencedirect.com/science/article/pii/S1352231003006721, 2003.

Soong, S.-T. and Ogura, Y.: A Comparison Between Axisymmetric and Slab-Symmetric Cumulus Cloud Models, Journal of the Atmospheric Sciences, 30, 879–893, https://doi.org/10.1175/1520-0469(1973)030<0879:ACBAAS>2.0.CO;2, http://dx.doi.org/10.1175/1520-0469(1973)030<0879:ACBAAS>2.0.CO;2, 1973.

Sullivan, T. J.: Introduction to Uncertainty Quantification, vol. 63 of *Texts in Applied Mathematics*, Springer-Verlag, Cham Heidelberg New York Dordrecht London, https://doi.org/10.1007/978-3-319-23395-6, 2015.

van Oldenborgh, G. J., Burgers, G., Venzke, S., Eckert, C., and Giering, R.: Tracking Down the ENSO Delayed Oscillator with an Adjoint OGCM, Monthly Weather Review, 127, 1477–1496, https://doi.org/10.1175/1520-0493(1999)127<1477:TDTEDO>2.0.CO;2, https://doi.org/10.1175/1520-0493(1999)127<1477:TDTEDO>2.0.CO;2, 1999.

Walther, A.: Automatic differentiation of explicit Runge-Kutta methods for optimal control, Computational Optimization and Applications, 36, 83–108, https://doi.org/10.1007/s10589-006-0397-3, https://doi.org/10.1007/s10589-006-0397-3, 2007.

Xiao, Q., Kuo, Y.-H., Ma, Z., Huang, W., Huang, X.-Y., Zhang, X., Barker, D. M., Michalakes, J., and Dudhia, J.: Application of an Adiabatic WRF Adjoint to the Investigation of the May 2004 McMurdo, Antarctica, Severe Wind Event, Monthly Weather Review, 136, 3696–3713, https://doi.org/10.1175/2008MWR2235.1, https://doi.org/10.1175/2008MWR2235.1, 2008.

Zhang, X., Huang, X.-Y., and Pan, N.: Development of the Upgraded Tangent Linear and Adjoint of the Weather Research and Forecasting (WRF) Model, Journal of Atmospheric and Oceanic Technology, 30, 1180–1188, https://doi.org/10.1175/JTECH-D-12-00213.1, https://doi.org/10.1175/JTECH-D-12-00213.1, 2013.