MODELING LIGHTNING IN GIANT PLANETARY ATMOSPHERES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yury Salavatovich Aglyamov

May 2023

MODELING LIGHTNING IN GIANT PLANETARY ATMOSPHERES

Yury Salavatovich Aglyamov, Ph. D.

Cornell University 2023

Lightning is observed on all four Solar System giant planets; these observations can illuminate atmospheric structure in unique ways, but this relation requires interpretation. I present a computationally light, one-dimensional model of atmospheric convection and electrification and apply it to the Solar System giant planets. On Jupiter and Saturn, lightning observations are well-matched by the model; on Uranus and Neptune, existing observations are limited to the Voyager 2 flybys, and perspectives for future measurements are described instead. While quantitative results are imprecise, a number of strong qualitative results are demonstrated. On Jupiter, a dissolved antifreeze – presumably ammonia – in the water clouds is necessary to explain lightning observations. The vertically spread lightning distribution of Jupiter is replicated, and predicted to be matched on Saturn, Uranus, and Neptune, though shifted deeper on those planets. The best observable for constraining Jovian and Saturnian water abundance through lightning is the altitude of the deepest flashes. Anhydrous ammonia clouds are not predicted to substantially affect lightning on Jupiter and Saturn, but may play a large role in Uranian and Neptunian lightning, depending on the strength of the triboelectric effect under these conditions. From the model and comparisons to Jupiter, the Uranian and Neptunian lightning sferics observed by Voyager 2 are either deep lightning attenuated by the atmosphere between it and the spacecraft, or horizontally limited shallow lightning; a future Uranus Orbiter could resolve this question by attempting to detect an optical signal.

BIOGRAPHICAL SKETCH

Yury Salavatovich Aglyamov was born in Serpukhov, Russia on the 26th of September, 1995. His family, including his then-infant younger brother Fedor, moved to the United States in 2001, following the trajectory of the research group his father worked in, first to Ann Arbor, MI and then to Austin, TX. Yury was interested in math and science from elementary school, and participated in academic competitions at both Kealing Middle School and LASA High. In particular, his high school years were dominated by the Science Olympiad competition, and it was in it that he first started to focus on geoscience, initially because the school team lacked students for those subjects. After over a decade in the heart of Texas, Yury insisted on living somewhere with cooler summers, but that still left a wide range; he went to Caltech for college in 2013, initially unsure about his major, but after his first college-level geology class he immediately committed to the Geoplanetary Sciences department, majoring in geophysics and doing summer research in related areas. His interest in planetary science specifically, especially the giant planets, grew over that time. In 2017, not long after the Juno mission arrived at Jupiter, Yury was accepted to the graduate program at Cornell, where he joined Professor Jonathan Lunine's group, which would later be named the Planetary Origins, Evolution, and Life Research Group. Of several initial project ideas around Jupiter's atmosphere, the lightning direction proved the most fruitful, leading to the results described in this dissertation. After graduating in 2023, Yury intends to continue working on planetary atmosphere topics at the University of Michigan-Ann Arbor.

To my parents – thank you so much, for everything.

ACKNOWLEDGMENTS

First of all, a mountain of thanks to Jonathan Lunine – for being a phenomenal advisor, for his guidance and incredible patience. Thanks also to the rest of POELRG – especially Ishan Mishra and Ngoc Truong. We came to Cornell together, and together we've made our path. I'm grateful also to the other members of my committee, Geoff Abers, Alex Hayes, and Saul Teukolsky, as well as to my collaborators and mentors in the Juno mission; in addition to their valuable input, my research rests on their work in the most direct causal sense. To Andy Ingersoll and Dave Stevenson, Steve Levin and Scott Bolton, Sushil Atreya and Cheng Li, Heidi Becker and Tristan Guillot, and everyone else in the project that I've had briefer opportunities to discuss Jupiter with.

The department atmosphere felt like home from my initial grad school visit, and despite the difficulties of Covid, I'm thankful that it's held together. For keeping me connected, thanks to AGN and especially to the giant 2017 intake. The image that comes to mind is an elephant in a horse race… well, one way or another, we've shoved our way through. For keeping me sane, thanks especially to the board game regulars – Larry, Will, Bo, and Gabe.

And, above all, thanks to my family – Mom and Dad, Fedor, my grandparents, and the extended family.

This work was performed under a Juno subcontract to Cornell University from the Southwest Research Institute; at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

		Units
b	radius of a given cloud particle	m
b_0	approximate radius of particles in a given bin,= arithmetic mean of its bounds	m
C	aerodynamic drag coefficient of a particle, assumed constant	--
c_P	specific heat capacity of dry air, taken as 14.5 kJ K^{-1} kg^{-1} for Jupiter	kJ $K^{-1}kg^{-1}$
E	electric field at a given pressure level	N C^{-1}
E_i	collision efficiency = probability that two colliding cloud particles will merge	--
E_q	charging efficiency = probability that two colliding particles will transfer charge	--
f	*mass* fraction of condensate *vapor* in air, 0.00625 for solar water abundance on Jupiter	--
f_S	*mass* fraction of condensate *vapor* in the air that corresponds to saturation vapor pressure	--
g	gravitational acceleration, assumed constant, 24.79 m s^{-2} for Jupiter	m s^{-2}
J	vertical current density at a given pressure level	C s^{-1} m^{-2}
k	slope of the particle density distribution as a function of radius	m^{-5}
L	latent heat of vaporization of the condensate (water), assumed constant at 2.257 MJ kg^{-1}	MJ kg^{-1}
M	total mass density of cloud particles within the size range of a bin	kg m^{-3}
N	total number density of cloud particles within the size range of a bin	m^{-3}
n_0	particle density distribution as a function of radius at the center of a bin	m^{-4}
P	pressure level under consideration	Pascals
R	universal gas constant 8.314 J K^{-1} mol^{-1}	J $K^{-1}mol^{-1}$
r	radius of the entraining plume	m
T	temperature of the ascending air parcel	K
T_f	temperature of the dry air surrounding the plume	K
t_L	time since the last lightning discharge	s
V_m	molar volume of gas at temperature and pressure under consideration	m^3 mol^{-1}
w	upward velocity of the entraining plume at a given point	m s^{-1}
w_i	terminal velocity for a particle of a given size relative to its surroundings	m s^{-1}
w_{rel}	average relative velocity of particles in two size bins	m s^{-1}

ε	ratio of the molar mass of the condensate (water) to that of dry air, 8.2 for Jupiter.	--
ϵ	relative electric permittivity	--
μ	molar mass of dry air, assumed constant, 0.0022 kg mol^{-1} for Jupiter	kg mol^{-1}
φ	fraction of the entraining plume replaced with dry air per unit pressure rise	Pa^{-1}

CHAPTER 1

INTRODUCTION

Lightning has been detected at each of the Solar System giant planets, starting with Voyager 1 at Jupiter, which achieved detection both with its plasma wave instrument (Gurnett *et al*. 1979) and its optical camera (Cook *et al*. 1979). Each subsequent spacecraft in the Jupiter system has detected lightning (Lanzerotti *et al*. 1996, Rinnert *et al*. 1998, Brown *et al*. 2018), allowing measurements of lightning energy and depth to be made (Borucki *et al*. 1982, Borucki & Williams 1986, Little *et al*. 1999, Becker *et al*. 2020). Electrostatic discharges, not initially recognized as lightning but later shown to be such, were also detected by Voyager 1's radio science instrument at Saturn (Burns *et al*. 1983, Aplin *et al*. 2020); optical detection was complicated by Saturn's ringshine, but was eventually achieved with Cassini (Dyudina *et al*. 2010, Dyudina *et al*. 2013). The only spacecraft visits to Uranus and Neptune, the Voyager 2 flybys, also detected lightning sferics there (Zarka & Pedersen 1986, Zarka *et al*. 2004). Previous modeling work has indicated that the lightning observed on Jupiter and Saturn comes from the water cloud deck, although the situation on the ice giants is less clear (Yair *et al*. (1995ab), Gibbard *et al*. (1995)) .

Lightning's energetic nature makes it one of the few phenomena we can observe below giant planetary cloud tops. As such, in addition to its direct significance, it can serve as an indirect probe for deep atmospheric properties and processes (Ingersoll *et al*. 2000, Wong *et al*. 2008). This, however, requires modeling to be linked with the observable properties of lightning. One such observable, unexpectedly found by the Juno mission, is Jovian 'shallow lightning' at pressure levels of 1-2 bars, well above the water cloudbase (Becker *et al*. 2020).

This shallow lightning is nevertheless consistent with being generated by the water cloud, as can be shown by a simple scaling argument. The electrical energy available for lightning can be approximated by estimating that particles accumulate charge linearly (at a constant rate in time) by a non-inductive mechanism, implying the electric field is quadratic in time until it reaches the breakdown value; the electric current is proportional to the charge, as particles of different size have different terminal velocities. The breakdown field can be approximated as linear in pressure (Gibbard 1996). Available power per unit volume is proportional to PtQ' where P represents pressure (or, equivalently, breakdown field), t the time to reach breakdown field, and Q' the rate of charging due to particle collisions. Since t goes as $\sqrt{(P/Q')}$, power goes as $P^{1.5}Q'^{0.5}$. Q' is proportional to n^2vq', where n is the number of particles per unit volume, v the terminal velocity difference, and q' the microscopic charging rate. The particle size profile will change with altitude, but generally, the largest particles will be formed in strong updrafts; thus, if convection is vigorous enough to reach the 1-2 bar pressure level, particle size differences should be more pronounced there than in deeper levels. Since q' for ice was experimentally found by Keith and Saunders (1989) to be proportional to $v^{2.5}$, and terminal velocity in an ideal gas (for fixed particle shape) is proportional to $\sqrt{(T/P)}$, the available power scales as $P^{0.625}T^{0.875}n$. The temperature profile mapped by the Galileo probe (Seiff *et al*. 1998) found temperatures near 200 K at the 1.5-bar level, and 250 K at the 4-bar level (the level of free convection is unlikely to be deeper than this, so deeper lightning activity is expected to rely on precipitation from above). At these temperatures, the vapor pressure of water is near 0.1 Pa, compared to 60 Pa at 250 K (Wexler 1977); however, vigorous convection should carry a substantial fraction of water condensed at lower altitudes to this level. Multiplying these factors, the lower limit of lightning abundance at the 1-2 bar pressure level can be expected to be 600 times less than at the 4-bar

pressure level, though a much higher abundance is expected; given the far smaller flash energies seen by the SRU than previous observations, this would imply a comparable or greater flash rate. Alternatively, if cloud particles are carried up without being lost, so that their volume density remains constant, the electrical power should be within a factor of 2-3 at the higher level, allowing for a much greater flash rate.

Below, the broad strokes of this approach – depth profiles, time-averaging over short timescales, and non-inductive charging of cloud particles – are combined with the Gibbard (1996) model to form the Giant-planetary Entraining Plume Electrification model, a one-dimensional model describing the ascent and electrification of a single plume in a planetary atmosphere, hereafter GEPE (Aglyamov 2023b). Jupiter's atmosphere has been simulated by both time-dependent 2-D lightning models (Yair *et al*., 1995a, Yair *et al*. 1998) and 3-D models of moist convection driven by water (Palotai *et al*. 2014, Li and Chen, 2019). 3-D lightning models have also been published for Earth, such as that of Fierro *et al*. (2013). These models provide important information on the nature of moist convective storm formation. A 1-D model such as GEPE has substantial limitations, most significantly that it cannot include wind shear and particle collisions therein, as well as being unable to accurately simulate hail. However, 3-D models are computationally expensive, and only more accurate given sufficiently constrained parameters: for example, 3-D lightning models on Earth can simulate a particular storm type, but this involves knowledge about the 3-D temperature and velocity structure of a typical storm cell that we simply don't have for planetary weather systems. GEPE serves a different purpose; it is able to span a wider parameter space at lower computational expense, without requiring specifics about individual weather systems, and thus provide a bridge between observations and

processes, as well as between simple intuitions of stacked cloud decks and a turbulent reality.

Chapter 2 discusses the specifics of model construction. Chapter 3 describes its application to Jupiter, and in particular the problems of shallow lightning and deep atmospheric composition mentioned above, describing what conclusions can be drawn about Jupiter from our limited lightning observations; this corresponds to the work that has been published as Aglyamov *et al*. (2021). Chapter 4 extends the model to non-ideal gases and other giant planets; the work described there has been submitted as Aglyamov *et al*. (2023a). Chapter 5 describes an extension of the model to allow multiple cloud species, with application to ammonia and ammonium hydrosulfide; this work has not yet been published separately. Chapter 6 is a brief conclusion. An appendix containing example computer code for the most recent version of GEPE is included.

CHAPTER 2

NUMERICAL MODEL

2.1 Introduction

This chapter describes in detail the functioning of the Giant-planetary Entraining Plume Model (Aglyamov 2023b), while subsequent chapters describe its results. This one-dimensional model builds on the work of Gibbard (1996), with an entraining plume scheme originally proposed by Stoker (1986) and a charging model described in Gibbard *et al*. (1997). GEPE started as an extension of the Gibbard (1996) model to account for the phase transition between liquid and solid water and the need for liquid water in the system to produce lightning on Earth (Saunders 2008), with the goal of testing whether lightning rates could constrain Jovian water abundance. In the process, several bigger changes were made, especially regarding the electrification model. This led to version 0.4 of the model, as described in section 2.2, which was used for predicting lightning depth distribution on Jupiter; the results were published as Aglyamov *et al*. (2021) and are discussed in Chapter 3. Further work extended the model to other giant planets (version 0.5, section 2.3 and chapter 4) and to non-aqueous clouds (version 0.6, section 2.4 and chapter 5). GEPE successively executes a model of convection and particle growth, a model of precipitation, and a model of charging and electric field growth; the overall control flow of the model, as of version 0.5/0.6, is summarized in Figure 2.1, and sample code for version 0.6 is provided in the Appendix.

2.2 Model version 0.4

2.2.1 Convection model

We begin with a parcel of moist air at 10 bars for Jupiter and 1 bar for the Earth, surrounded by dry air of the appropriate composition for each body. So long as the moist air is no more buoyant than the dry air around it, both are assumed to follow an adiabatic profile, a dry adiabat except in the case when the moist air has a partial pressure of water exceeding the saturation vapor pressure calculated as per Lowe (1977), in which case water condenses, releasing its latent heat, and the temperature changes with pressure as per a moist adiabat. In calculating the heat capacity of Jovian air, only hydrogen and helium in the observed Jovian ratio (von Zahn et al. 1998) and water vapor (if any) are considered, as the contribution of other jovian gases is negligible.

Once the moist parcel is buoyant, convection is initiated, and the parcel gains a positive upward velocity that initially increases, as well as being able to keep in suspension particles whose terminal velocity is lower than the parcel's upward vertical velocity, allowing condensate to accumulate and particles to grow. The parcel temperature is assumed to change adiabatically, and the faster the plume rises, the faster it will incorporate dry air from the surrounding, downwelling region. A larger entrainment rate is mathematically equivalent to a smaller plume radius, and leads to more dry air from the surrounding environment introduced into the column. The entrained air follows the environmental temperature profile. In the buoyant region it will be denser than the plume despite having a lower molecular weight, making buoyant convection more difficult. Further, positive buoyancy is required to accelerate

initially stationary dry air to the plume's upward velocity. The condensate mass decreases buoyancy so long as it remains suspended in the plume, but this is a smaller effect.

Like Gibbard (1996), we use an entrainment rate

$$\phi = 0.2\frac{RT}{P\mu rg} \tag{2.1}$$

as the fraction of mass added per unit pressure. As such,

$$dw = -w\phi dP \tag{2.2}$$

$$df = -f\phi dP \tag{2.3}$$

and the concentration of cloud particles in each size bin is likewise decreased; the plume radius increases, but is capped to be no more than a factor of √2 greater than the initial radius to leave room for downdrafts. That is, for a dry adiabat with entrainment we take

$$\frac{dT}{dP} = \frac{RT_v}{\mu P c_P} - \phi\left(T - T_f\right) \tag{2.4}$$

and for a moist adiabat with entrainment

$$\frac{dT}{dP} = \frac{\left[\frac{dT}{dP}\right]_{dry} + \left[\frac{dT}{dP}\right]_{dry}\frac{Lf_S\mu}{RT_v} + \left(T - T_f\right)\phi\frac{Lf_S\mu}{RT_v} - \frac{Lf_S\phi}{c_P}}{1 + \frac{L^2 f_S \epsilon\mu}{c_P RT^2}} \tag{2.5}$$

where T_v is the virtual temperature defined by

$$T_v = T\frac{1+\frac{f}{\epsilon}}{1+f} \tag{2.6}.$$

For an adiabat outside the plume, either below the level of free convection or in the regions of descending dry air, which are assumed to surround the plume, the terms with entrainment rate are zero. Note that, while it is possible that the external environment might not be characterized by a dry adiabat, what we care about is the

temperature contrast created by the different temperature profiles within and outside the plume, which affects the buoyancy. Given that the entrainment is determined by the cross-sectional size of the plume, the distribution of which is not known for Jupiter, introducing a temperature profile different from dry adiabatic would simply add another parameter that is at the moment poorly constrained. We felt it simpler to make the most idealized assumption for the tropospheric environmental temperature gradient. However, inclusion of a stratosphere is important as the reversal of the environmental temperature gradient at the tropopause strongly brakes the ascent of buoyant plumes. As the stratospheric temperature profile does not approximate an adiabat, the Jovian stratospheric environmental temperature profile at pressures less than 0.25 bars is assumed to be equal to that found by the Galileo probe, as per Seiff *et al*. (1998), and the Earth stratosphere to the ISA Standard Atmosphere.

Particles are assumed to immediately reach their terminal velocity, given by

$$w_i^2 = \frac{8 b \rho g R T}{3 C \mu P} \qquad (2.7).$$

Particles in bins whose terminal velocity exceeds the upward velocity of the plume are precipitated immediately and removed from the plume, and any remaining particles left over in the plume if it reaches the top of the atmosphere are precipitated out at that point. The quantity and size distribution of these precipitating particles is recorded at each pressure level, to be included into the electrification model for deeper levels.

From the velocity and the hydrostatic pressure gradient, the time it takes the parcel to ascend through the pressure step of 10 pascals can be calculated and used for particle growth calculation. Both liquid water droplets and ice particles in the cloud are treated as being spherical with a constant density of 1000 kg m^{-3} for liquid and 400 kg m^{-3} for ice (graupel). They are divided into 31 bins according to their liquid-

equivalent radius (from which their actual radius, mass, and volume immediately follow), with each bin's upper boundary being a radius √2 times its lower boundary. The smallest bin stretches from a radius of 10 microns to a radius of 14.1 microns, with the largest bin's upper bound being a radius of approximately 46.3 centimeters (not reached in practice, but estimated to be near the maximum size of Jovian hail). Particles initially condense with a flat distribution of radii in the smallest bin; changing the radius of this bin by a factor of 2 in either direction changes flash rates by less than 10%, demonstrating stability with respect to the choice of size bin. The arithmetic mean of these upper and lower bounds, b_0, is used as an approximate radius of particles within the bin in some of the calculations to follow. We describe the size distribution of particles by a piecewise linear function F(b) such that the number density of particles with radius between b_1 and b_2 in a region is

$$N = \int_{b_1}^{b_2} F(b)\,db \tag{2.8}$$

As such, F(b) has values with dimension m^{-4}. In a given size bin, it is assumed that F(b) is linear except if this would cause it to be negative,

$$F(b) = max\left(0, n_0 + k\left(b - b_0\right)\right) \tag{2.9}$$

definedby two parameters per bin, n_0 (with units m^{-4}) and k (with units m^{-5}); for empty bins $n_0 = k = 0$. In the largest non-empty size bin, we thus tend to have F(b) be piecewise linear within the bin, with a negative slope until F(b)=0 and constantly zero thereafter. It is assumed that all particles within a bin precipitate together, using the terminal velocity of a particle of radius b_0. The total volume (and from it the total mass M) of particles in a given bin can thus be found by integrating

$$\frac{4\pi}{3} b^3 F(b) \tag{2.10}$$

from the lower bound of the bin to either its upper bound or the x-intercept of F(b); the average mass can be found by dividing M by the integral of F(b) over that range, which is the number density N.

It is assumed that, whenever two cloud particles collide, they will stick together to form a single particle with probability equal to E_i. Calculating the frequency of such collisions requires knowing the typical relative velocities of cloud particles. It is assumed that the velocity is predominantly vertical, with the relative velocity of the cloud particles being the difference between the terminal velocity at the upper bound of the larger bin and the lower bound of the smaller bin. This ensures that particles from the same bin have a nonzero relative velocity and can merge with one another. As such, the number of collisions $n_{coll}(i,j)$ between a particle from bin i and a particle from bin j, $i \geq j$, per unit time and per unit volume is

$$n_{coll}(i,j) = \pi b_0 \left(\left[b_0(i) \right]^2 + \left[b_0(j) \right]^2 \right) w_{rel}(i,j) N(i) N(j) E_i \qquad (2.11).$$

Multiplying by the time step gives a number of collisions per unit volume.

Collisions are assumed to remove a number of particles from the smaller bin equal to the number of collisions, with average mass equal to the average mass of the smaller bin. This mass is in its entirety transferred to the larger bin. However, this increase will cause some of the particles in the larger bin to be transferred from bin i to bin i+1. For this calculation, it is necessary to make simplifying assumptions. We define the radius bx as the radius of a particle that, when colliding with a particle of radius $b_0(j)$, would produce a particle with radius equal to the upper bound of bin i. Then, it is assumed that exactly the particles with radius greater than bx are transferred to bin i+1 upon collision, so that their total mass and number are transferred up a bin, along with the proportionate amount of mass initially from bin j. In this fashion, the total mass

and number density are updated for each bin. Following this, n_0 and k are rederived from M and N, with the formula dependent on whether or not F(b) has an x-intercept within the bin, and particle growth is iterated for the duration of the ascent through the pressure step.

In an actual storm, there would be a gradient between fully liquid and fully frozen particles. Here, to make the calculation tenable, we assume that all particles at a given altitude are either liquid water or ice; the boundary between them is set at a specific temperature, as the freezing point of water does not change significantly with pressure in this pressure range. This freezing point is set at 273 K or lower, the latter to simulate the possible supercooling of water—a complex phenomenon involving multiple microphysical effects—down to 233K (e.g. Pruppacher & Klett 1997), or a more important depression (possible down to 173K) due to the dilution of ammonia into water (Guillot *et al*, 2020). The difference between liquid water and ice lies in the values of E_i and E_q, the charging efficiencies. These efficiencies hold different values for liquid and ice: that is, E_i(liquid) + E_q(liquid) = 1, and E_i(solid) + E_q(solid) = 1. Since ice is assumed to satisfy E_i=0 and thus E_q=1, as E_q(liquid) decreases, charging and lightning will increasingly be dominated by that occurring at temperatures below freezing and therefore shift upward. However this also implies that E_i(liquid) –a variable parameter-- increases, allowing for faster growth of particle size contrast and more lightning overall.

2.2.2 Electrification model

In the above fashion, at each pressure level, a particle size distribution is calculated, as well as the sum of the size distribution of precipitation coming from above. The rest

of the computations to determine electric field buildup do not alter this distribution; hence, electrostatic levitation (which might serve to increase the mean particle size at which fallout occurs) is not included. Furthermore, particle evaporation (virga) at altitudes above the 10-bar level and particle re-lofting (hail) are not considered. The following calculations are performed at each pressure level, but using a step size of 1 kilopascal.

Firstly, the precipitation coming from above is weighted by the fraction of time it spends within the range of the pressure step. The height of this step is calculated as before, and the particle fall velocity as terminal velocity minus plume velocity. Any precipitating particles with a terminal velocity slower than the plume velocity are considered uninvolved in lightning production (in practice they would either be re-lifted or precipitate outside the plume area), while the upward-moving plume particles are added later. For a set particle radius, the time spent in a certain height range would be equal to height over velocity, but this leads to infinite values for particles just large enough to have a terminal velocity equal to the plume velocity. As such, each contributing particle radius within the bin is also weighted by the difference between its terminal velocity and the updraft velocity. The natural (asymptotic) weighting is

$$f_{bin} = \frac{w_W}{w_i - w} \qquad (2.12),$$

where w_W is the updraft velocity at the point where the particle was dropped by the plume. Instead, we perform the calculation for particles slightly larger than the critical radius by treating the bin as if its lower bound was at the critical radius and taking an average velocity for this shrunken bin, leading to a weighting

$$f_{bin} = w_W \frac{b_{+.} - b_m}{\int_{b_m}^{b_m \sqrt{2}} w_i(b) - w db} \qquad (2.13).$$

This yields

$$f_{bin} = \frac{3b_{+.} - 3b_m}{b_m\left[3w - 2w_i\left(b_m\right)\right] - b_m\sqrt{2}\left[3w - 2w_i\left(b_m\sqrt{2}\right)\right]} \qquad (2.14),$$

where w_W is the plume velocity at the altitude of the precipitation's origin, w the plume velocity at the altitude of the electric field growth, b_+ is the bin's upper bound, and b_m is the greater of the bin's lower bound and the critical levitating radius. The factor of $\sqrt{2}$ is the factor between the bin boundaries, as near the singularities b_m approaches b_+ and we wish to lower the weight of such bins to reflect that most particles in them are being lofted in these conditions.

This creates a weighted precipitation size distribution. Because n_0 and k do not in general add linearly, we convert from n_0 and k to M and N, add the weighted precipitation size distribution to the suspended cloud particle size distribution, and generate an overall particle size distribution that is then converted back to n_0 and k. We assume this population is stable in time; in reality particles fall down and similar particles, which may have a charging history involving different conditions, fall from above.

Non-inductive charge transfer is assumed, with the empirically observed formula for the amount of charge transferred between colliding ice particles of Keith and Saunders (1989). It is assumed that a collision between two particles will lead to charge transfer with probability

$$E_q = 1 - E_i \qquad (2.15).$$

As a result, particles of equal size colliding will have no net effect. As such, $w_{rel}(i,j)$ is calculated simply as the difference between the terminal velocities $w_i(b_0(i))$ and $w_i(b_0(j))$, the arithmetic means of the bin bounds, and collision probability is

calculated as with particle growth. In this fashion, a rate of charge accumulation per particle is calculated for each bin; under the assumptions being made, this rate is constant in time. Since particles are therefore charged linearly, the electric current caused by faster fall velocities of larger particles also increases linearly in time, and therefore the electric field is quadratic in time,

$$E = \sum_i \left[N(i) \frac{dQ(i)}{dt} \frac{\left[w_i\left(b_0(i)\right) - w \right]}{\epsilon_0} \right] \frac{t_L^2}{2} \tag{2.16}$$

The current density is linear in time:

$$J = \sum_i \left[N(i) \frac{dQ(i)}{dt} \left[w_i\left(b_0(i)\right) - w \right] \right] t_L \tag{2.17}$$

When the electric field reaches its breakdown value, a lightning discharge occurs. This breakdown value is taken as ($5\ m^2\ C^{-1}$)P for hydrogen on Jupiter and ($3\ m^2\ C^{-1}$)P for nitrogen/oxygen on Earth from Gibbard (1996), based on a combination of theoretical modeling and observational data. The energy released per unit volume is equal to this breakdown field, times the total charge transferred per unit area. When lightning thus occurs, the charge is assumed to reset to zero at that location and begins building up as before. Thus, dividing the energy per volume by the time-to-discharge yields a rate of time-averaged energy production per unit volume. Integrating over pressure leads to a power per unit area. The calculated power is a rate per unit area of updraft; that is, it does not include downwelling regions of the storm, and it does not include sections of the atmosphere that do not contain a thunderstorm at a given time.

While the model as described in equations 2.1 through 2.17 is based on that in Gibbard (1996), there are substantial differences, the greatest being a different assumption of equilibration. In the Gibbard model, convection is modeled separately from particle growth and charging; as such, a plume is modeled without considering precipitation, and then particle growth and charging is assumed to happen in a parcel of air without

new influx of material. Since this would ultimately result in all particles growing large enough to precipitate, the particle growth must be limited to an arbitrary time interval; this provides an additional tunable parameter, but one that is artificial. We instead assume that water or ice particles grow in a parcel of air as it ascends, precipitating out once their terminal velocity exceeds the upward velocity of the plume, with the implicit assumption being that the overall vertical structure of the plume is stable in time, with air and water circulating through it. While the plume structure evolves during a thunderstorm, in broad terms (which a constant entrainment rate implies) this is a reasonable approximation: with model velocities it takes 1-2 hours for the plume to ascend to the stratosphere, a short time compared to storm lifetimes, which are on the order of 3-14 days (Hueso *et al*. 2002, Li *et al*. 2004).

The calculation for non-inductive particle charging was, as in Gibbard (1996), done separately, following the assumption of a stable plume. However, because we calculate the particle size distribution at each altitude, we are able to include both precipitation and cloud particles in calculating particle charging. We also included a phase change, altered the particle growth algorithm, particularly in calculating collision velocities, and targeted the calculation towards obtaining a flash rate per unit area rather than a time to first flash.

2.3 Model version 0.5

Several changes were made to apply the model to Saturn, Uranus, and Neptune, which also implied further recalculation of Jupiter results. Firstly, the higher pressures involved in the water clouds of these planets, especially for Uranus and Neptune, make the ideal gas equation of state used by Gibbard (1996) and Aglyamov *et al*.

(2021) a poor approximation. The upper portion of the water cloud in these bodies may be at roughly 100 bars pressure and room temperature. There the mean free path is of order a nanometer, only a few times the collision diameter for hydrogen molecules. Therefore, the gas can no longer be considered to be composed of point particles, and a correction must be made for the finite molecular volumes. To do so we use the van der Waals-based equation-of-state of Hemmes. Driessen, and Griessen(1985). This equation was chosen because it is relatively computationally inexpensive, accurate for the moderate temperatures and pressures (less than 1000K and 1000 bar) involved, and has a closed form that allows differentiation:

$$T=\left(P+\frac{a(P)}{V_m^{\alpha}}\right)\left(V_m-b(P)\right)\left(R^{-1}\right) \tag{2.18}$$

$$a(P)=P^{-0.8946}e^{19.599-e^{-18.608}P^{2.6013}} \tag{2.19}$$

$$b(P)=\begin{cases}\sum_{i=0}^{8} b_i \ln(P)^i & if\ P<100\,bars \\ P(100\,bars) & if\ P>100\,bars\end{cases} \tag{2.20}$$

$\alpha = 2.846$

$b_0 = 20.285$

$b_1 = -7.44171$

$b_2 = 7.318565$

$b_3 = -3.463717$

$b_4 = 0.87372903$

$b_5 = -0.12385414$

$b_6 = 0.0098570583$

$b_7 = -0.00041153723$

$b_8 = 0.00000702499$

Since this equation and its derivatives give temperature as a function of pressure and molar volume, the variable tracked with the ascending plume is now molar volume

instead of temperature, and temperature is calculated when needed directly from pressure and molar volume.

The van der Waals equation above was derived for hydrogen, and does not address secondary constituents such as helium and water vapor. To prevent complexities related to mixing, we assume they follow the same per-mole equation of state, with the only difference being molecular weight; as such, gas at a given temperature and pressure will have a set molar volume regardless of composition, and total density scales linearly with molar mass.

Secondly, multiple changes were made to prevent the arbitrary pressure level of the model base from affecting results, if all convective and lightning activity happens above. The temperature of the moist parcel is now taken to be equal to the surrounding air at the cloudbase and all points below, only diverging above the cloudbase. Additionally, evaporation of precipitation below the cloudbase is now included.

Evaporation is assumed to proceed according to Knudsen's equation (equation 2.22 below) and the evaporation coefficient of Kessler (1969). Evaporation proceeds when the vapor pressure of water is lower than the saturation vapor pressure by a deficit pressure P_x. Evaporation due to a vapor pressure deficit should proceed at the same rate as condensation due to a vapor pressure excess; such condensation has a theoretical maximum where every excess vapor molecule that impacts the droplet sticks to it. These impacts are generated by both random thermal motion and the falling velocity of the droplet, with the velocities of the two being randomly oriented relative to one another and therefore adding as a root sum of squares:

$$O = \sqrt{O_{thermal}^2 + O_{fall}^2} \tag{2.21}$$

$$O_{thermal} = \frac{P_x}{\sqrt{2\pi P V_m \mu_{water}}} \tag{2.22}$$

$$O_{fall} = \frac{P_x v_{fall}}{4 V_m P} \tag{2.23}$$

where O is the impact rate (m^{-2} s^{-1}), $O_{thermal}$ and O_{fall} are its thermal and mechanical components, P the overall pressure (Pa), P_x the deficit vapor pressure (Pa), V_m the molar volume (m^3), v_{fall} the droplet fall velocity (m/s), and μ_{water} the molar mass of water (kg).

Experimentally, the growth and evaporation of water droplets is substantially slower than Knudsen's equation would directly predict, due to a number of microphysical processes (i.e. evaporating cooling of the droplet surface). This requires an evaporation coefficient $\varepsilon<1$; we use the Kessler (1969) evaporation coefficient

$$\varepsilon = 1.527 * 10^{-4} r^{-0.4} \tag{2.24}$$

which depends on droplet radius (in meters). Specifically, the coefficient for each particle size bin is calculated from the droplet radius r_0 that is the arithmetic mean of the bin boundaries.

This can be used to calculate the rate of mass loss for a given droplet per unit surface area, and thereby the rate at which droplet radius decreases:

$$\frac{dm}{dt} = \frac{-\varepsilon O \mu_{water}}{4} \tag{2.25}$$

$$\frac{dr}{dt} = \frac{1}{\rho}\frac{dm}{dt} \tag{2.26}$$

where ρ is the droplet density.

The particle size distribution in each bin is still assumed to be linear in radius, as per equation 2.9. There are two direct effects of droplet evaporation on the particle size

distribution: droplets at the bottom end of a larger bin moving into a smaller bin, and the shrinkage of particles that stay within a given bin. For the former effect, the instantaneous change in droplet number depends on the number of droplets at the bottom end of the larger bin (radius r_{min})

$$\frac{dN}{dt} = \left(n_0 + k\left(r_{min} - r_0\right)\right)\frac{dr}{dt} \tag{2.27}$$

and the change in mass by

$$\frac{dM}{dt} = \left(\frac{4}{3}\pi r_{min}^3\right)\frac{dN}{dt} \tag{2.28}$$

This is an increase in particle number and mass for the smaller bin, and a decrease for the larger bin. The same equation is used for the smallest size bin for overall evaporation of particles.

For the latter effect, droplet number per bin remains constant, but the bin mass decreases for an integral of all droplets in the bin:

$$R_3 = \frac{1}{3}\left(r_{max}^3 - r_{min}^3\right) \tag{2.29}$$

$$R_4 = \frac{1}{4}\left(r_{max}^4 - r_{min}^4\right) \tag{2.30}$$

$$\frac{dM}{dt} = \left(R_3\left(n_0 - kr_0\right) + R_4 k\right)\frac{dm}{dt} \tag{2.31}$$

This is then applied to the precipitation falling from the cloudbase, using the terminal velocity of droplets to convert from a time-derivative to a space-derivative.

Once the particle size distribution is known at every pressure level, the electric field can be calculated. Here, however, the steady-state assumption of the previous version of the model must be abandoned in favor of a more realistic procedure. Previously, the model assumed that particle charges would grow linearly in time with collisional non-inductive charge transfer as per Keith and Saunders (1989), leading to quadratic growth of the electric field in time at a given pressure level. This is a valid

approximation if the time a droplet takes to fall through the atmospheric column is much longer than the time to generate a breakdown electric field. Within the cloud, this is generally true for conditions in the giant planets. However, most of this charging is due to collision of large precipitating particles with smaller cloud particles, and below the cloud base, where the smallest particles evaporate very quickly, charging is therefore extremely slow. Electric field growth is nonetheless still expected, as the droplets in question formed significantly higher up in the ascending plume and have already acquired a charge by the time they precipitate.

As such, we use two separate models of electric field growth. Above the cloudbase, the electric field is quadratic in time, as in the previous version of the model; below the cloudbase, however, particle charge is calculated for a given particle as it falls from its point of precipitation. At a given pressure level, charge is therefore constant in time and the electric field grows linearly with time. This difference leads to a visible discontinuity in graphs of flash rate at the cloud base, but not one so large as to be unacceptable.

An additional complication for Uranus and Neptune is electromagnetic levitation: unlike for Jupiter and Saturn, breakdown fields in the water cloud region are high enough that, before such fields are reached, the electrostatic force on falling particles from the field affects particle velocities. Eventually, in a 1-D model, this leads to a field that is constant in time, which Gibbard (1996) argues will prevent lightning in ice giant water clouds.

There are, however, reasons lightning may nonetheless be possible. Firstly, while it is true that the electric field becomes stable in time for a 1-D model, this does not mean

all particles are falling with the same velocity, but rather that the relative velocities and charges cancel out. This, however, is within the context of a constant particle size distribution in distinct bins; particle growth over time and precipitation from above and below would prevent an equilibrium. A full analysis of these effects would require a 2-D model of the plume to account for time evolution. Secondly, real storms are not axisymmetric, and horizontal winds are substantial. Within the context of significant moving charges, this implies electromagnetic as well as electrostatic effects, along with turbulence effects.

Since electric field growth for Uranus and Neptune water clouds is fast compared to fall times, it seems unlikely for a stable electric field to keep charged particles suspended for a full storm lifetime. Nevertheless, the effects of electrostatic levitation are likely to decrease electric field growth.

To address this issue, we further modify electric field growth. Electric field growth proceeds as above until reaching a value E_l where levitation becomes significant. Between E_l and the breakdown field (if the breakdown field is larger than E_l), by contrast, electric field grows as the square-root of time either above or below the cloud base. As such, electrical potential energy is built up linearly in time, with a constant power, from this point until the breakdown field is reached and lightning discharges.

2.4 Model version 0.6

With the inclusion of ammonia, two condensible species are present rather than one. Some ammonia is sequestered in the water cloud at a preset N:O ratio. Remaining

ammonia is assumed to condense out when its partial vapor pressure exceeds the saturation vapor pressure of Lange (1967):

$$P = 10^{3+\left(6.67956-\frac{1002.711}{T-25.215}\right)} \tag{2.32}$$

Ammonium hydrosulfide, NH_4SH, is believed to form clouds in Jupiter and Saturn, and possibly the ice giants as well, in between the water and ammonia cloudbases (Atreya and Wong 2005). It would actually be formed by the chemical reaction of gaseous ammonia and gaseous hydrogen sulfide, but following Zuchowski *et al.* (2009), is for the sake of tractability treated as condensing if its abundance exceeds an effective vapor pressure

$$P = 10^5 e^{120.678-\left(\frac{2915.7/T+1.76\ln(T)-0.00078T}{0.167}\right)}. \tag{2.33}$$

The size distribution of ammonia and ammonium hydrosulfide clouds is treated as piecewise linear, as with water clouds. Also like water clouds, ammonia cloud particles are assumed to initially condense out at a radius between 10 μm and 10√2 μm, with a constant size distribution in that range, but unlike water, at a density of 680 kg/m^3 if liquid and 350 kg/m^3 if solid. The density of solid ammonia is chosen to have roughly the same relation to its observed density of 820 kg/m^3 as the ratio of modeled graupel density (400 kg/m^3) to solid ice (920 kg/m^3), to represent that it, like water, is also likely to condense in a complex fashion rather than as a solid crystal. Similarly, ammonium hydrosulfide particles are assumed to have a density of 1200 kg/m^3 if liquid and, preserving the same ratio, 520 kg/m^3 if solid.

Ammonia and NH_4SH cloud particles grow in the same fashion as water cloud particles; in this work, the collision efficiencies were assumed to be equivalent to water due to lack of evidence otherwise. They do not merge with water cloud particles present in the plume, keeping two or three fully separate populations. As

with water, if their terminal velocity exceeds the upwelling plume velocity, they will precipitate out, and upon falling below their species' cloud base will gradually evaporate. Evaporation is assumed to proceed as with water, using the coefficient of Kessler (1969), due to lack of distinct data on evaporation kinetics of ammonia or NH_4SH, but using the vapor pressure deficit for the relevant species rather than water.

Electric charging proceeds by particle collisions, with the charges on ammonia, NH_4SH, and water particles treated separately. In addition to water-water collisions, two additional mechanisms of particle charging are present: ammonia-ammonia (or NH_4SH-NH_4SH) particle collisions, and at altitudes where multiple cloud types overlap, collisions between particles of different species.

Ammonia-ammonia collisions are assumed to proceed analogously to water-water collisions, with the smaller particle acquiring a positive charge and the larger particle a negative one, as per the experiments of Keith and Saunders (1989). In general, since ammonia is less polar than water, it may be expected that such charge transfer is smaller in magnitude for ammonia than water. We follow Gibbard (1996) in assuming that the charge transfer is proportional to material dielectric constant; that is, an ammonia-ammonia collision will transfer charge, in Coulombs,

$$Q = \frac{\epsilon_{NH_3}}{\epsilon_{H_2O}} \left(\frac{w_{rel}}{3} \right)^{2.5} \left(10^{-15} G \right) \tag{2.34}$$

where

$$G = 2.71 * 10^{-5} \left(r_G^{2.7} \right) if r_G < 111\,\mu m \tag{2.35}$$

$$G = 9.88 * 10^{-2} \left(r_G^{0.98} \right) if r_G > 111\,\mu m \tag{2.36}$$

where r_G is the radius of the smaller of the colliding particles in microns. We approximate ϵ as 25 for ammonia and 80 for water. The same approach is applied to

ammonium hydrosulfide, except that its dielectric constant is not experimentally known. It is approximated by interpolation: the dielectric constant at a given wavelength is equal to the square of the index of refraction at that wavelength, and so we assume that the overall (infinite-wavelength) dielectric constant varies as a power law of refractive index at short wavelengths, and use the values of Joiner and Steffes (1991) to estimate a dielectric constant of 74.

For ammonia-water collisions, a significantly greater quantity of charge transfer can be expected due to the difference in composition (rather than a merely geometric difference between differently sized particles); this is one expression of the triboelectric effect. As studied by Lee *et al*. (2018), such charging is related to hydrophilicity, with hydrophilic particles acquiring a positive charge and hydrophobic particles a negative one. Lesprit *et al*. (2020) showed a scaling with velocity of $w_{rel}^{1.5}$, instead of the $w_{rel}^{2.5}$ scaling of Keith and Saunders (1989) for chemically identical particles. This may be justified by the difference between geometry-caused charge transfer, where the geometry changes over the timespan of the collision, and chemistry-caused charge transfer, which continues throughout the time of contact.

A significant deficit of experimental data regarding the actual charging of ammonia ice remains, and quantifying hydrophilicity based on contact angle with water, as is done in Lee *et al*. (2018), is complicated because ammonia is soluble in water. For a first approximation, we assume that a 'chemical potential'-like term is proportional to one minus the cosine of the contact angle, and use the results of Ruzaikin *et al*. (2022) for the contact angles of ammonia and water with steel to estimate that the NH_3-H_2O hydrophilicity difference is 0.317 times that of the hydrophilic glass-hydrophobic glass difference in the Lee *et al*. experiment. We assume the same ratio holds for the

quantity of charge transferred by the triboelectric effect. This implies a charge transfer per particle collision, in Coulombs, of

$$Q=\left(\frac{w_{rel}}{3}\right)^{2.5}\frac{20.68}{w_{rel}}\left(10^{-15}G\right) \tag{2.37}$$

with G defined as per equations 2.35 and 2.36. For NH_4SH, due to lack of experimental data, we treat charging quantity as proportional to dielectric constant difference; instead of 20.68, the constant is thus 18.42 for NH_4SH-NH_3 collisions and 2.26 for NH_4SH-H_2O collisions, with positive charge transferred from ammonia to NH_4SH and from NH_4SH to water, while negative charge is transferred in the opposite direction.

Summing the charging rates of particles due to collisions with all other bins, now for both water and ammonia (and where necessary NH_4SH), give an overall charging rates per particle, which add up to zero when summed over all particles. The relative motion of particles then creates a current until a breakdown field is reached, as in previous versions of the model.

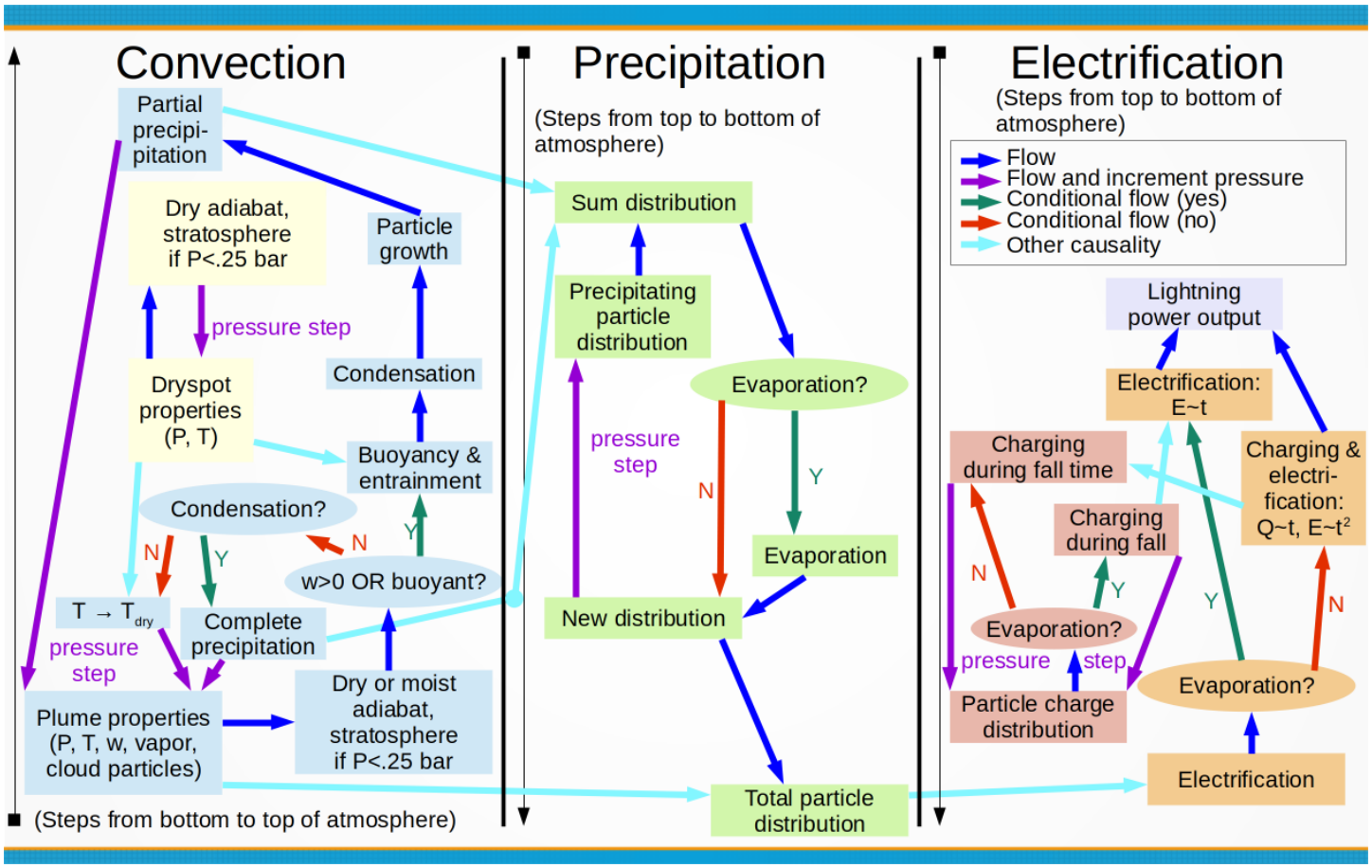


Figure 2.1: Flowchart of GEPE's overall structure, for version 0.5 or 0.6.

CHAPTER 3

JUPITER: LIGHTNING GENERATION, CONVECTION, AND CONSTRAINTS

3.1 Introduction

While Jovian lightning has been known since Voyager 1 (Cook *et al*. 1979, Gurnett *et al*. 1979), with substantial further observations by the Galileo mission (Lanzerotti 1996, Little *et al*. 1999) and modeling efforts such as Yair *et al*. (1995a), the Juno mission has been a massive step forward in observations thereof. In particular, the MWR (microwave radiometer) instrument has shed light on latitudinal lightning distribution (Brown *et al*. 2018), and the SRU (stellar reference unit) has been used to detect significantly shallower lightning flashes than previous work (Becker *et al*. 2020). At the same time, while Juno has provided constraints on Jovian water abundance (Li *et al*. 2020), a precise value remains elusive. Since lightning is one of the relatively few signals we receive from levels of Jupiter below the uppermost cloud deck, it can be of use as a tracer of convection and potentially a constraint on water abundance (Ingersoll *et al*. 2000, Wong *et al*. 2008), but such an approach requires modeling work to connect these atmospheric variables with observable flash rate. Lightning in the water cloud requires a sufficiently high water abundance; additionally, terrestrial lightning generally involves liquid water (Christian *et al*. 2003, Saunders 2008), and the ice-ice electrification observed on Earth has been in a storm system that includes the liquid water regime (Saunders 1993, Dye & Bansemer 2019). This chapter describes the application of GEPE to Jupiter for this purpose, concluding that existing lightning observations do not strongly constrain oxygen abundance.

However, they do imply substantial freezing point depression, likely due to ammonia antifreeze in the water cloud particles.

Section 3.2 describes the parameters used for GEPE 0.4 throughout the calculations. Section 3.3 then details the calibration of these parameters with Earth observations. Section 3.4 summarizes model results for Jupiter. Section 3.5 discusses how available observational data relates to model output, and section 3.6 discusses the resulting constraints on Jovian atmospheric conditions that can be derived from lightning.

3.2 Choice of parameters

The model involves a number of known parameters, such as the molecular mass of Jovian air and the heat capacities of the substances involved, the latter of which were assumed to be the NIST-JANAF values. Furthermore, values such as the size of each particle radius bin or the time step should not affect the calculation so long as they are sufficiently small. However, seven parameters represent genuine unknowns about the atmosphere of Jupiter.

The water abundance in Jupiter's atmosphere, below its clouds, is not well known due to spatial variation and model dependencies. Given the solar composition as per Asplund *et al*. (2009), if all oxygen were in the form of water, the water mass fraction in Jupiter's atmosphere would be approximately 0.00625; this is referred to as 1-solar water abundance. The Galileo probe measured a subsolar oxygen abundance, despite most other measured elements being enriched by a factor of 2-3 relative to the solar atmosphere, but this may have been an atypical value due to dynamical effects leading to subsidence and drying in the “5-micron hot spot” into which the probe fell (Orton *et*

al. 1998, Atreya *et al*. 1999). Juno's MWR has found a deep-water value at a latitude just north of the equator of between one and five times solar (Li *et al*., 2020). Within the Great Red Spot, high-resolution 5-μm spectroscopy constrains water to the 1.1-12x solar range (Bjoraker *et al*. 2018). A greater water concentration leads to earlier and more extensive water condensation, and therefore greater moist convection and more lightning, although other parameters such as the temperature profile are also known to have an effect (Yair *et al*. 1998). Since we as yet do not know the deep water abundance for regions far from the equator, four values considered for water mass fraction were 0.3-solar, 1-solar, 3-solar, and 10-solar, spanning the wide range of values consistent with the Galileo Probe measurement, modeling of the disequilibrium CO abundance, and tropospheric gravity waves radiating from Shoemaker-Levy 9 impacts (Wong *et al*. 2004, Wang *et al*. 2015, Ingersoll and Kanamori 1995). Given the number of other parameters that are uncertain, finer water abundance increments are not useful.

The temperature of Jupiter's atmosphere at 10 bars pressure was measured at 338 Kelvin by the Galileo probe (Seiff *et al*. 1998). Because the probe descended in a 5-micron hot spot atypical of most of Jupiter, we consider possible temperature values of either 330 K or 350 K, with most calculations done at the cooler temperature. This may be larger than the plausible range of 10-bar temperatures, but bounds the problem. Higher temperatures lead to water condensing higher up in the atmosphere, but also higher updraft velocities, with differing signs of effect on lightning; overall, the temperature profiles lead to similar conclusions. It is possible to have the plume begin at a higher temperature than its surroundings, but this is not necessary for convection because the difference between moist and dry adiabats creates a

temperature contrast between the plume and its surroundings with increasing altitude (under our assumption of a dry adiabatic environmental temperature structure).

The drag coefficient observed in Earth rain varies depending on raindrop size, taking its lowest value at a drop radius near 1 mm and rising for both smaller and larger sizes (Spilhaus 1948). Additional complications are added by the state of water, and the history of that state. We assume the drag coefficient of a sphere in turbulent flow, which is approximately 0.5, for all cloud and precipitation particles. Higher drag coefficients correspond to lower terminal velocities. While this enables larger particles to be supported by the plume, its main effect is to greatly slow particle growth and charging due to lower collision velocities, thus leading to lower flash rates.

The collision efficiency and the charging efficiency both affect the generation of lightning, and which of them is the greater constraint depends on conditions. Their actual values depend enormously on the physical state (shape, stickiness, etc) of the condensate particles being considered, which for Jupiter are unknown even in general. We optimistically assume that their sum is unity, that is, that every collision not resulting in particles merging transfers charge. Furthermore, we assume that ice has a collision efficiency of zero (and so a charging efficiency of unity), because ice particles are unlikely to stick upon collision; this explains the need for liquid water in lightning-generating systems. Then, we find that the model best fits observed behavior of Earth's atmosphere if liquid water has E_i=0.8 (and so E_q=0.2); see Table 3.1 and figure 3.1. Jovian calculations for E_i=0.5 (E_q=0.5) were also performed, and found to be qualitatively similar. A complete absence of charging in liquids would imply E_i=1 (E_q=0), but some liquid electrification is observed on Earth (Stetten et al. 2019), and liquid charging is experimentally observed during drop breakup or

inductive charging (Saunders 2008). Furthermore, the equilibrium liquid region will still have some solid precipitating particles, as well as melted precipitation that was charged while still solid. However, despite these effects, Earth electrification is dominated by solid-state charging; therefore, a small but nonzero liquid charging efficiency is a reasonable result. In the case of Jupiter, analyses of data from both the Galileo and Cassini spacecraft have found lightning at depths greater than 5 bars (Little *et al*. 1999, Dyudina *et al*. 2004), which have temperatures warmer than 273 K, and therefore liquid water for any amount of freezing point depression, for either the Galileo probe temperature profile or an adiabatic temperature profile; as such, a nonzero liquid charging efficiency is required to fit observations.

The radius of the entraining plume is set by the properties of convection. An overly small plume will entrain so much material that it stifles itself, whereas a very large plume will break up into smaller plumes (figure 3.1). In the model, the latter process does not happen, and so it is necessary to choose a plume size that is big enough to be convective but small enough not to escape the atmosphere. For Jupiter, a plume radius of 5 kilometers fits these conditions and is consistent with the results in Yair *et al*., (1995a). Hueso *et al*. (2002) consider storms of 25 km, but refer to this as an upper limit beyond which their storms break up. Since storms may consist of multiple upwelling plumes, our choice is not necessarily inconsistent with their upper limit. For Earth, typical thunderstorms have upwelling cores that are roughly a kilometer in diameter, though larger upwellings also occur (Stolzenberg 1998).

The energy of a single flash has a strictly inverse linear effect on the flash rate: the lightning power per unit area does not in this model depend on the flash energy, and thus, for example, doubling the flash energy will always halve the flash rate. As such,

the flash rate can be calculated for multiple values of flash energy without rerunning the model, and we use constant values of flash energy for both settings: Given typical total flash energies of $1.5*10^9$ J as per Maggio *et al*. (2008), the peak observed rate in updrafts corresponds to an electrical energy flux of 2 W/m^2; with the chosen parameters for collision efficiency, our model matches this well, predicting maximum values 0.1 to 4 W/m^2 depending on temperature. Less intense lightning flashes on Jupiter, as observed by Juno, would correspond to higher predicted flash rates. These values refer to the total electrical energy released by the flash, and not to the optical flash energy, which may be several orders of magnitude smaller.

3.3 Earth calibration

To calibrate the model, Earth cases were considered with a surface (sea level) temperature of 280, 290, 300 or 310 Kelvin, 90% humidity, and environmental temperature 5.5 to 14.5 K higher than plume temperature at sea level (to prevent immediate buoyancy of the vapor-containing plume), for various collision efficiencies with a plume radius of 1 kilometer, consistent with observations of terrestrial thunderstorms. Under these conditions, a buoyant plume is formed, with maximum upward velocities between 15 and 40 meters per second (figure 3.1).

For mesoscale convective complexes on Earth, the peak rate of cloud-to-ground discharges has been observed to be on the order of $1.5*10^{-3}$ flashes per square kilometer per second, which must be doubled when restricted to only updrafts; for the system as a whole, though, the maximum rate was observed at $3.3*10^{-6}$ flashes per square kilometer per second (Goodman and MacGorman 1986). For the entire Earth, including regions without cloud cover, the often-quoted value is 100 flashes per

second, corresponding to a flash rate of $2*10^{-7}$ flashes per square kilometer per second (Orville and Spencer 1979). More recent space-based observations have given values a factor of two lower (Christian *et al*., 2003). Given typical total flash energies of $1.5*10^{9}$ J, we find that our model is consistent within a factor of a few with peak flash rates observed for Earth within convective storms.

The rate and altitude of lightning flashes and temperature dependence for the model, as applied to Earth conditions, best fits observations for a water collision efficiency of 0.8 and a freezing point of 253 K. The former constraint is relatively weak, with collision efficiencies of 0.5 or 0.9 producing similar results. A collision efficiency of 1.0, however, produces lightning concentrated at overly low pressures. Flash rates peak near the 0.5-bar level, higher at lower freezing points; however, flashes are observed at a variety of altitudes, down to the ground. While the model does not specifically simulate cloud-to-ground flashes, flashes in the lowermost atmosphere represent charge transfer to the ground, which leads to cloud-to-ground flashes in practice. Greater freezing point depression causes flashes to increasingly occur in the stratosphere, and also greatly decreases the dependence of flash rate on temperature; for comparable levels of free convection, supercooling of 10-20 Kelvin leads to the flash rate varying by a factor of 4-10 per 10 K base temperature, consistent with observation. Observationally, some supercooled water droplets have been observed at temperatures as low as 233 K, but a lower upper limit is reasonable for the bulk. This offers a constraint for interpreting the Jovian results: freezing point depression greater than ~20 K cannot, within the model, be explained by supercooled water.

3.4 Jupiter results

When considering the flash rates of lightning to the depth of 10 bars in Jovian conditions, the flash rate is mainly determined by the water abundance and the freezing point (tables 3.2 through 3.7 and figures 3.2 through 3.7). With insufficient freezing point depression, no lightning occurs; otherwise, the flash rate rapidly climbs to a value largely dependent on water abundance alone. This value ranges from about 200 W/m^2 for 0.3x solar water to 15 000 W/m^2 for 10x solar water, and the minimal amount of freezing point depression drops from 60 K to 20 K over that range.

Lightning occurs over a wide range of pressure levels. A substantial fraction of lightning occurs in the 0-2 bar region, where even ammonia-containing water is likely to solidify (tables 3.6 and 3.7). However, when lightning occurs in the local absence of liquid, it then relies on advection of material from below, as particle growth does not take place without a liquid phase. For a freezing point above 253 K and 3x solar or lower water, no lightning at all is observed, due to the absence of liquid above the level of free convection. For parameter values near this boundary, electrical power is extremely low, and its value and distribution change very rapidly compared to the size of the parameter grid.

The electrical power rises slightly faster than linearly with water abundance, if the freezing point is sufficiently low. With greater water abundance, convection shifts deeper into the atmosphere, requiring less freezing point depression; but due to greater convective vigor, lightning rates increase throughout the atmosphere, and the depth distribution of available electrical power does not drastically change. Medium and deep lightning rates grow faster than shallow lightning rates because the level of free convection is deeper, but lightning in the cloud grows more quickly with water abundance than below the cloud, and for that reason medium and deep lightning rates

overall grow at similar speeds with increasing water abundance. Similarly, if the freezing point is not far below 273 K, a colder temperature profile leads to deeper condensation and greater lightning rates, particularly at deep pressure levels.

An extreme freezing point depression of more than 60 K—much greater than the 0-40K drop generally inferred for terrestrial clouds where NH_3 is absent-- decreases shallow lightning rates, as the condensate being liquid throughout the plume lowers the efficiency of electric charging at high levels, and little particle growth occurs so high up regardless. That is, shallow lightning becomes less frequent due to the lower charging efficiency of liquid. At these extremely low freezing points, the effect of 10-bar temperature on lightning rate lessens and in some cases reverses, because the colder profiles have less vigorous (albeit deeper) convection.

3.5 Comparison with observations

Galileo's observations of Jupiter's global flash rate were on the order of 10 strokes per second (Little *et al*, 1999), less than Earth over an area 100 times greater; Juno observed a higher flash rate but much optically dimmer flashes summing to a lower energy flux, implying that a different part of the lightning distribution was being sampled. This sets an absolute lower bound of 10^{-4} watts per square meter for the power flux assuming a flash energy of 10^{12}J as per Yair *et al*. (2008, 1995a). For individual storms, an areal lightning analysis for Galileo for three storms found a flash rate of 0.2 s^{-1} for each on average (Little *et al*. 1999). The sizes of the storms have large uncertainties ranging between 10^5 and $1.4*10^6$ km^2. The implied flash rate is thus between $1.4*10^{-7}$ and $2.0*10^{-6}$ s^{-1} km^{-2}, and likely closer to the lower part of this range. This rate is multiplied by four to account for the storm's non-rectangularity and

the portion of it taken up by downdrafts. This provides a lower bound between 0.9 and 12 W m^{-2} for flash energies as per Yair *et al*. (2008), far lower than the bulk of model predictions. The reason is that the model is designed to predict a peak flash rate, which requires higher resolution that can be achieved for Jupiter. However, the prevalence of lightning is sufficient to exclude parameter combinations that produce no lightning at all and provide additional evidence against 0.1x solar water.

The depth of lightning observed on Jupiter can be estimated, via a model of atmospheric scattering, from the half-width at half-maximum (HWHM) of the flash's spatial extent. Such estimates have been reached for several flashes by the Galileo Solid State Imager and the Juno Stellar Reference Unit (Little *et al*. 1999, Dyudina *et al*. 2002, Becker *et al*. 2020). Estimates were also made from Voyager 1 observations, but given the lower spatial/temporal resolution, doubts have been raised as to whether some of these recorded flashes were in reality overlapping sets of smaller flashes. While the Juno observed flash rate was 15 times higher than the Galileo flash rate, average Galileo flash energies were ~150 times higher, and measured at greater depths. The sign of this difference is expected: Galileo's measurements were made with longer exposure time but lower sensitivity than the Juno SRU, meaning that fewer but more intense lightning flashes should be observed, and the breakdown field increases with pressure, explaining the higher intensity in deep flashes. In SRU data the most intense Juno flashes were almost equal in energy to the weakest Galileo flashes, implying that the two energy ranges can be concatenated to find the overall rate; however, if very infrequent flashes more intense than the Galileo range were found to occur on Jupiter, these conclusions would be altered.

Quantitative differences suffer from small-number statistics. The Juno SRU observed four flashes with HWHM of 40 km or less, corresponding to a pressure level of 0-2 bars (hereafter shallow flashes), and two with HWHM between 40 and ~ 55 km, corresponding to a pressure level between 2 and 4 bars (hereafter medium-depth flashes). The latter were on average more energetic, with 55% of the energy coming from medium-depth flashes and 45% from shallow flashes. The Galileo observed energy flux (flashes per unit area times the average flash energy) was ten times higher, but only three flashes had depths measured, one (25% of the energy) a medium-depth flash, and two with half-widths above 60 km, expected to correspond to pressure levels of 4-10 bars (hereafter deep flashes). Dyudina *et al*. (2002) rederived depths and added more flashes, generally consistent with the Little *et al*. (1999) conclusion that Galileo observed deep flashes. Some of these flashes might occur below the cloud base; this is expected, as precipitation should carry charge differential down below cloud level, until the depth at which it evaporates. The precise depth at which this occurs is unknown and is unlikely to be exactly the 10 bars used in the model, but flash rates below the cloud base do not greatly increase with depth, so the effect of this uncertainty is limited.

As such, observations show that approximately 30% of flash energy is released at 0-4 bars, and that among those 0-4 bar flashes, 13% of the energy is released at 0-2 bars. This is reached based on a very small number of observations, but is most consistent with the model results for a freezing point near 213 K. For lightning at 0-4 bars pressure to output a minority of the energy, a minimum 40 K of freezing point depression - greater than the best fit for Earth - is required. While no clear conclusion on water abundance can be reached, 0.1x or 0.3x solar water require 80K or greater freezing point depression to satisfy this constraint for either temperature profile. The

upper limit on water abundance is global, as lightning will not preferentially form in the absence of clouds; the lower limit, by the same reasoning, is local, as lightning may only occur in limited wet regions of Jupiter. Further constraints would require continued measurements of the depth distribution of Jovian lightning, or else information on local flash rates.

3.6 Interpretation

The observation of lightning at altitudes above the 2-bar pressure level by Becker *et al*. (2020) is fully consistent with the model. Lightning power at these shallow levels is estimated to be 10-100 times smaller than the total lightning power down to 10 bars; however, as conditions in this region are more Earthlike, it is expected that such flashes would be lower-energy and therefore more common. This is precisely what we see in the combined Juno and Galileo data. While the water cloud base is far below this level, the water cloud tops are substantially higher, and lower breakdown fields are required to generate lightning at higher altitudes. The predicted distribution is smooth rather than bimodal, but a bimodal distribution seems unlikely unless lightning is being generated in the pure ammonia clouds (which are above the 0.7-bar level), a situation that is unlikely given that the condensate in those clouds is low-density solid. Both observation and model thus imply that lightning at Jupiter occurs in a broad altitude range, rather than at a singular pressure level.

Conversely, the observed shallow lightning seems to call for liquid water to occur at altitudes where the temperature is so far below the pure water freezing point that even supercooling cannot be invoked. Therefore, freezing point depression such as that caused by ammonia solvation in the liquid seems a plausible mechanism. But whether

ammonia is necessary to stabilize liquid water for lightning generation depends chiefly on the level of free convection. By the nature of both the model and the real Jovian atmosphere, particle growth and charging are both required for lightning generation, but they need not happen at the same time. If the plume rises from a level where liquid water is stable, allowing particle growth, ice particles will still charge one another in the upper reaches of the atmosphere to generate electric fields without the need for ammonia at the 1-2 bar level to keep the water liquid. However, if it is assumed that ice particles are 'non-sticky' and do not grow by collisions (or grow at very low efficiency), then this requires very vigorous convection ongoing below the ice line. While the base of the water cloud (the lifting condensation level) occurs at depths below the ice line for a subset of reasonable Jovian parameters, the level of free convection where the plume becomes buoyant is higher. This gap is further increased for Jupiter, relative to Earth, because water vapor has a lower molecular mass than nitrogen and is therefore buoyant on Earth, whereas in Jupiter's hydrogen-helium atmosphere the compositionally driven buoyancy is negative. In our model, the level of free convection typically occurs near 4 bars. In what follows, we do not consider additional speculative sources of convection such as deep alkali metal moist convection (Lunine *et al*. 1989) or mechanically forced convection for which we have no evidence at this time. A quantitative estimate of the effect this has can be gathered from varying the freezing point. Earth observations, as described above, require 10-20 K supercooling (Rutlege and Hobbs 1984, Pruppacher and Klett 1997). For Jupiter, given a 330 K base temperature, the water freezing point must be lowered, at the warmest, to 253 K for 3-solar water, to 233 K for 1-solar water, or to 213 K for 0.3-solar water to generate observable amounts of lightning; for energetic lightning to be mostly deep, even greater freezing point depression is required. These depressions are at the limits of supercooling in the terrestrial context, implying that water cloud

particles on Jupiter have substantial dissolved ammonia. The great latitudinal variability in lightning rates may indicate that Jupiter is in a region of the phase space where lightning rates vary extremely rapidly with changing parameters; this would imply that the true ammonia abundance corresponds to between 20 and 40 Kelvin freezing point depression in addition to supercooling, that is, water cloud droplets that contain 10-25 percent ammonia by weight relative to the total condensate (ammonia plus water). More ammonia is needed for lower deep-water abundances, but even for subsolar water abundance, lightning can be generated if enough ammonia to create liquid is present, and even for supersolar water abundance some ammonia is required.

In order for ammonia to depress the freezing point sufficiently, there must be enough of it relative to water to maintain a significant liquid fraction. In fact, equilibrium thermodynamics predict that, in Jupiter conditions, ammonia and water can mix to form a liquid at pressures between 1.1 and 1.5 bar and temperatures between 173K and 188K, as shown by Guillot *et al*. (2020). The consequent growth of these water-ammonia particles (‘mushballs’) and their fall through the cloud should lead to the presence of partially liquid particles in a wider range of pressure levels. The distribution of these particles in terms of sizes and pressure levels remains an open question. A model run with a separate liquid region at 173-188 Kelvin in addition to freezing point depression, as would naively follow from Guillot *et al*.'s equilibrium calculations, has been tested, with lightning being generated at lower water abundances than with freezing point depression alone, but the qualitative conclusions about the water abundance remaining the same. Full analysis of the mushball model will require modeling time-dependent processes in condensate formation, followed ultimately by three-dimensional convective modeling of mixed ammonia-water

storms. Potentially interesting charge reversal effects associated with ammonium ions may also be a further avenue for exploration (Workman and Reynolds, 1950).

Table 3.1: *Earth, electrical energy flux (W m^{-2}) vs. Temperature, Liquid Collision Efficiency, and Liquid Solution Freezing Point*

Freezing point, efficiency	280 K, 285.5 K	290 K, 298.5 K	300 K, 311.5 K	310 K, 324.5 K
273K, 0.5	0	0	0.3	2.8
263K, 0.5	0	0.1	1.2	4.6
253K, 0.5	0.1	0.5	2.0	5.4
243K, 0.5	0.4	0.7	2.3	5.6
233K, 0.5	0.5	0.9	2.4	5.7
273K, 0.8	0	0	0.4	4.7
263K, 0.8	0	0.2	2.1	7.1
253K, 0.8	0.2	0.7	3.0	7.5
243K, 0.8	0.4	1.0	3.3	7.1
233K, 0.8	0.5	1.1	3.0	7.0
273K, 1.0	0	0	0.5	5.5
263K, 1.0	0	0.2	2.3	6.9
253K, 1.0	0.1	0.7	2.8	5.7
243K, 1.0	0.3	0.8	2.1	3.9
233K, 1.0	0.3	0.6	1.3	2.4
	[a] First temperature is plume temperature at 1 bar; second temperature is environment temperature at 1 bar. Initial temperature contrast must be negative on Earth because water vapor has a lower molecular weight than nitrogen and oxygen, and is thus innately buoyant. Values of temperature contrast chosen to ensure level of free convection is at approximately 0.7 bar for all temperatures.			

Table 3.2: *Electrical energy flux above 10 bars (W m^{-2}) vs. Water Abundance and Liquid Solution Freezing Point. Liquid collision efficiency is 0.8, temperature at 10 bars is 330 K.*

Freezing point	0.1 x	0.3x	1x	3x	10x
273K	0	0	0	0	3367.6
253K	0	0	0	1206.7	13384.4
233K	0	1.0	322.3	3465.6	14045.6
213K	2.1	59.6	854.1	4070.3	16060.7
193K	21.3	213.9	1170.8	4401.4	16641.9
173K	26.9	241.3	1268.1	4577.7	17038.1
153K	27.9	250.1	1269.9	4593.3	17100.5

Table 3.3: *Electrical energy flux above 10 bars (W m^{-2}) vs. Water Abundance and Liquid Solution Freezing Point. Liquid collision efficiency is 0.5, temperature at 10 bars is 330 K.*

Freezing point	0.1 x	0.3x	1x	3x	10x
273K	0	0	0	0	1988.0
253K	0	0	0	754.1	9781.8
233K	0	1.0	207.6	2061.7	14699.9
213K	1.9	39.3	472.6	3046.5	19049.9
193K	17.0	110.2	1055.0	4855.5	20393.6
173K	29.0	211.0	1212.5	5049.6	21190.4
153K	32.7	220.1	1298.9	5216.8	21250.3

Table 3.4: *Electrical energy flux above 10 bars (W m^{-2}) vs. Water Abundance and Liquid Solution Freezing Point. Liquid collision efficiency is 1.0, temperature at 10 bars is 330 K.*

Freezing point	0.1 x	0.3x	1x	3x	10x
273K	0	0	0	0	4065.6
253K	0	0	0	1282.1	7361.3
233K	0	0.6	242.5	1094.8	2740.0
213K	0.6	22.8	103.3	354.7	690.1
193K	0.3	3.1	16.2	65.9	165.9
173K	0.1	0.5	3.6	8.8	19.8
153K	0.01	0.1	0.5	0.9	2.1

Table 3.5: *Electrical energy flux above 10 bars (W m^{-2}) vs. Water Abundance and Liquid Solution Freezing Point. Liquid collision efficiency is 0.8, temperature at 10 bars is 350 K.*

Freezing point	0.1 x	0.3x	1x	3x	10x
273K	0	0	0	0	1602.1
253K	0	0	0	586.9	11602.3
233K	0	0.005	180.1	2810.0	13030.6
213K	0.7	40.8	781.1	3425.2	14242.3
193K	4.5	174.6	1079.1	4125.9	15766.4
173K	8.8	200.7	1125.1	4169.9	15796.5
153K	11.5	203.5	1149.0	4203.2	15905.0

Table 3.6: *Shallow Lightning Fractions (percent) vs. Water Abundance and Liquid Solution Freezing Point* [a]

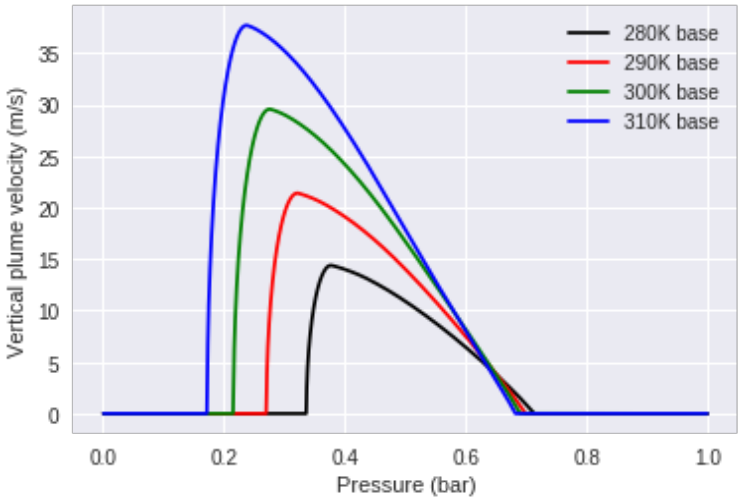
Freezing point	[b]0.1 x	0.3 x	1x	3x	10x
273K	--,--[c]	--,--	--,--	--,--	13.1, 58.2
253K	--,--	--,--	--,--	16.6, 88.9	15.1, 56.2
233K	--,--	34.5,57.4	32.9, 73.9	22.5, 46.1	18.2, 40.1
213K	50.4,38.0	53.3,49.0	25.7, 40.2	17.7, 37.5	11.9, 35.8
193K	20.7,24.9	12.3,32.5	8.5, 35.6	8.3, 35.5	6.6, 35.5
173K	18.7,24.4	11.1,32.3	6.8, 35.3	6.0, 35.0	4.2, 34.4
153K	18.6,24.4	10.7,32.1	6.6, 35.3	5.6, 34.9	3.8, 34.1

[a] For 330 K temperature at 10 bar Liquid collision efficiency is 0.8. First number is % of lightning from 0-2 bar relative to total in 0-4 bar. Second number is % of lightning from 0-4 bar relative to total in 0-10bar. [b]0.3x is 0.3 times solar water, etc. [c]- - indicates no lightning

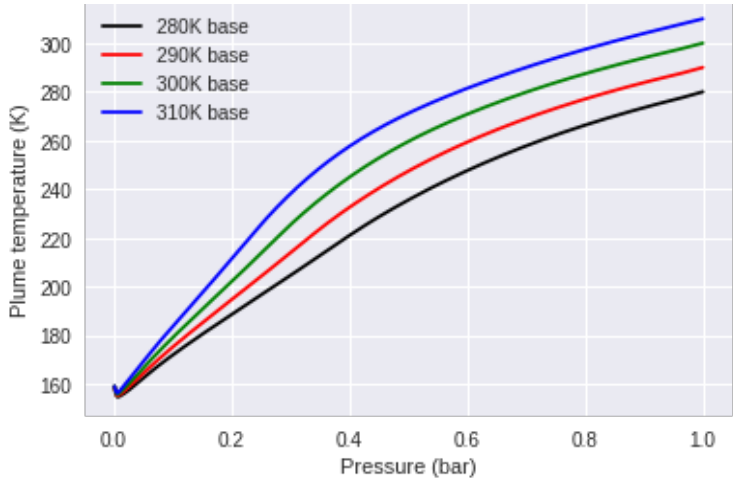
Table 3.7: *Shallow Lightning Fractions (percent) vs. Water Abundance and Liquid Solution Freezing Point* [a]

Freezing point	[b]0.1 x	0.3 x	1x	3x	10x
273K	--,--[c]	--,--	--,--	--,--	15.1,90.4
253K	--,--	--,--	--,--	29.8,89.1	22.8,61.6
233K	--,--	0.0,15.1	58.8,73.1	37.4,47.6	24.9,49.2
213K	13.8,20.0	68.3,45.1	31.7,36.4	24.9,38.9	15.0,46.4
193K	38.5,26.0	34.4,29.1	16.0,31.8	14.9,36.5	8.3,45.1
173K	34.1,24.6	33.6,28.8	14.5,31.4	13.6,36.1	7.5,44.9
153K	31.5,23.9	33.5,28.8	14.0,31.2	13.1,36.0	7.1,44.7

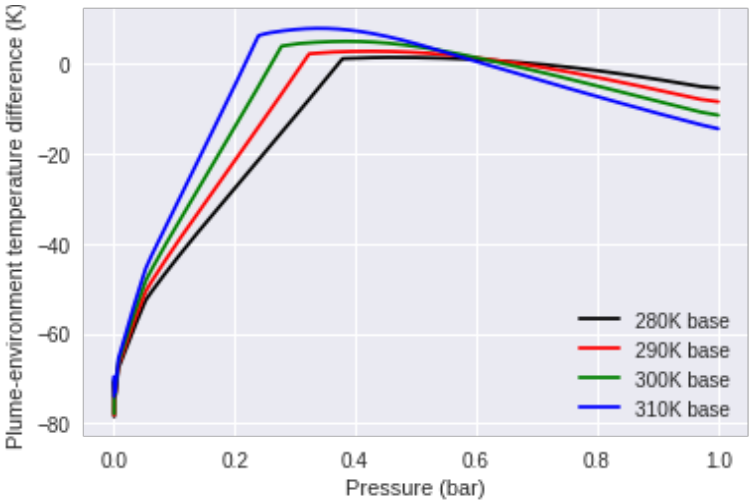
[a] For 350 K temperature at 10 bar. Liquid collision efficiency is 0.8.. First number is % of lightning from 0-2 bar relative to total in 0-4 bar. Second number is % of lightning from 0-4 bar relative to total in 0-10bar. [b]0.3x is 0.3 times solar water, etc. [c]- - indicates no lightning


280K base
290K base
300K base
310K base
Vertical plume velocity (m/s)
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0
Pressure (bar)

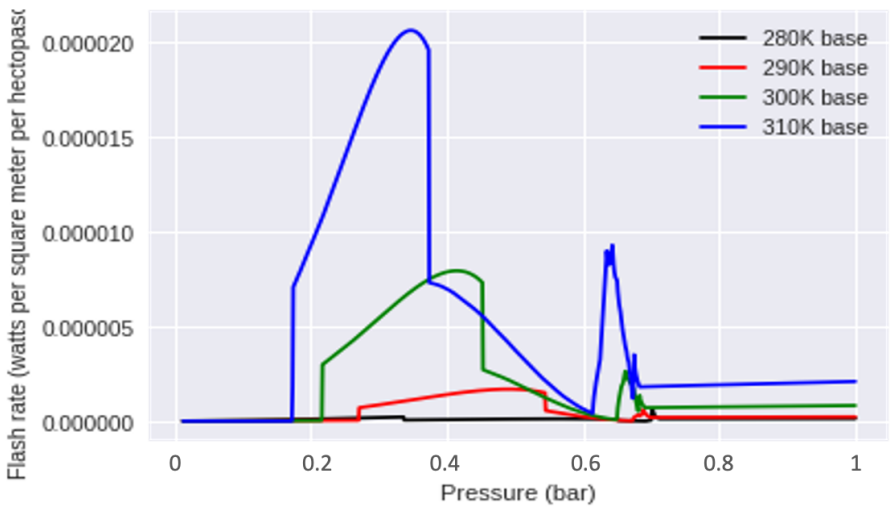
(a)


280K base
290K base
300K base
310K base
Plume temperature (K)
160
180
200
220
240
260
280
300
0.0
0.2
0.4
0.6
0.8
1.0
Pressure (bar)

(b)

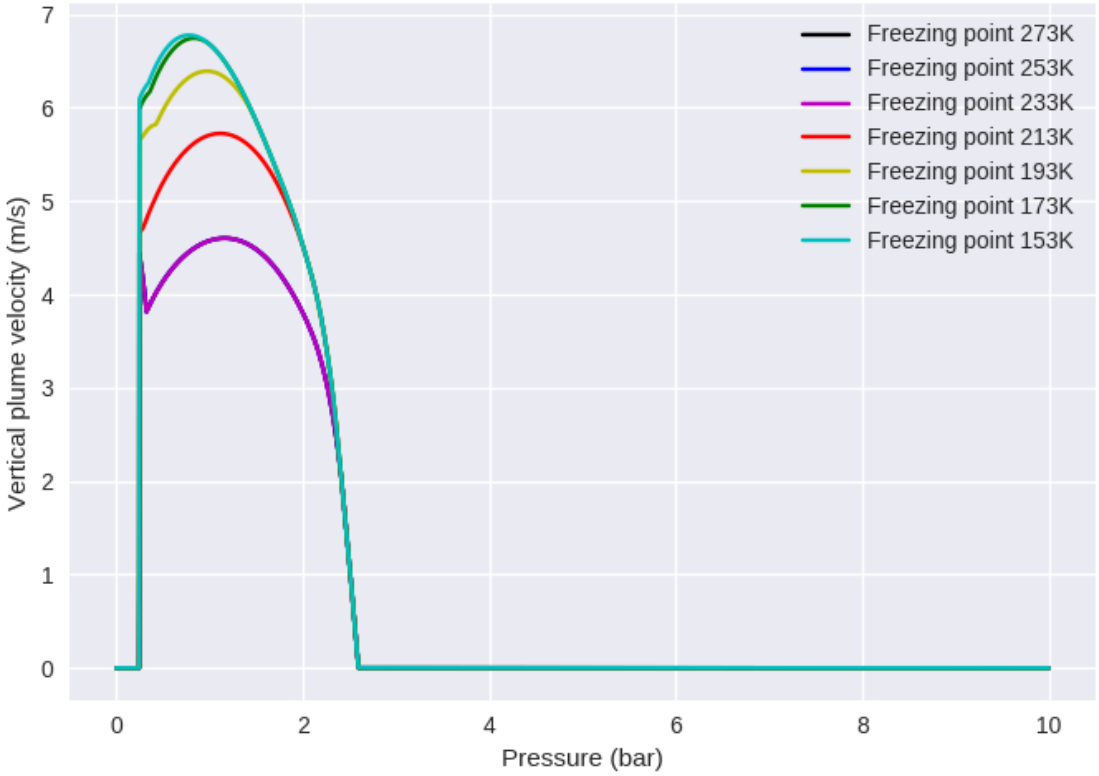

Plume-environment temperature difference (K)
0
−20
−40
−60
−80
280K base
290K base
300K base
310K base
0.0
0.2
0.4
0.6
0.8
1.0
Pressure (bar)

(c)

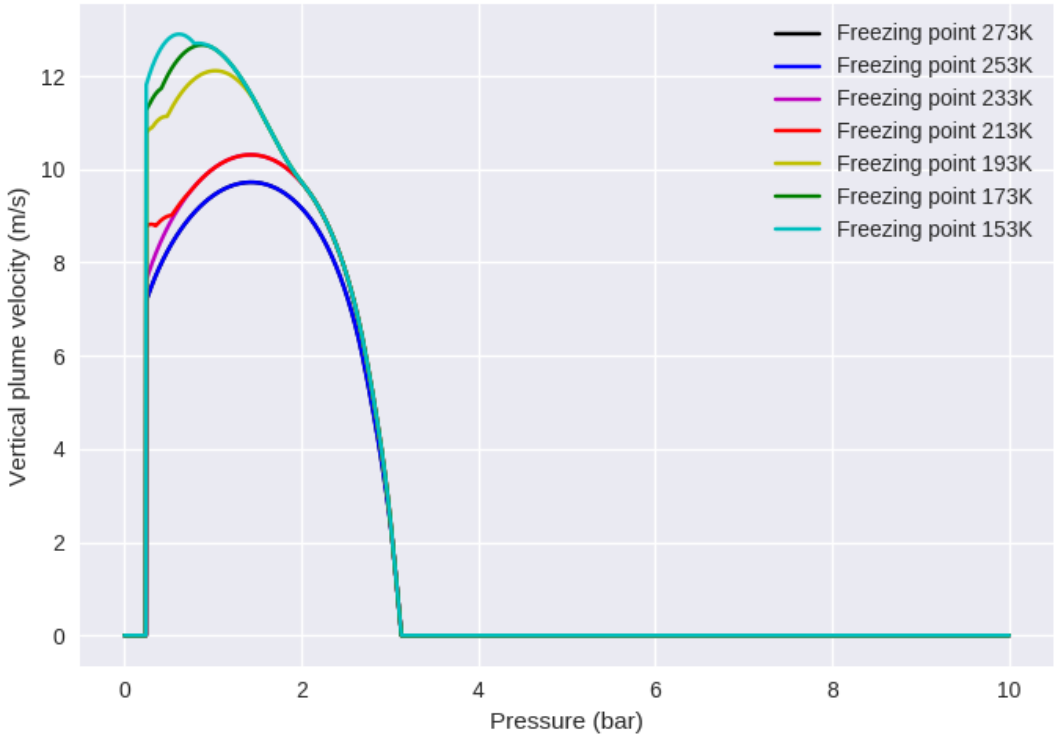


(d)

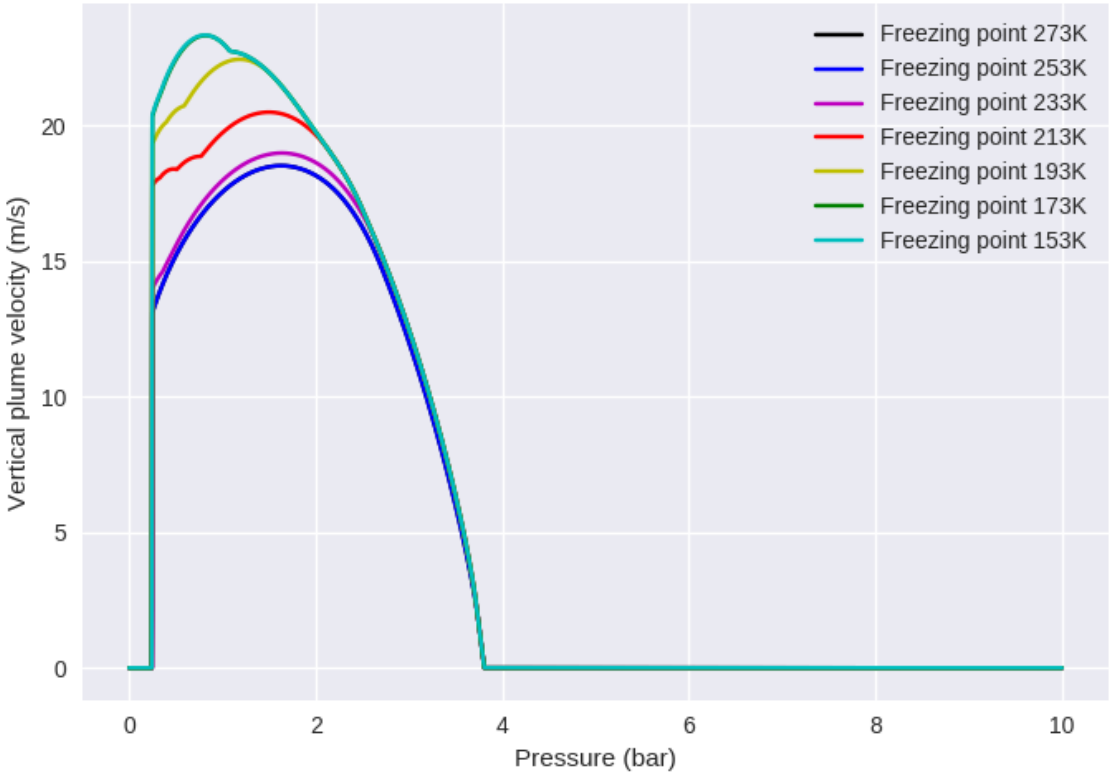
Figure 3.1: Earth profiles of (top to bottom) (a) updraft velocity, (b) plume temperature, (c)plume-environment temperature difference, and (d) flash rate for freezing point 253 K and liquid collision efficiency 0.8.



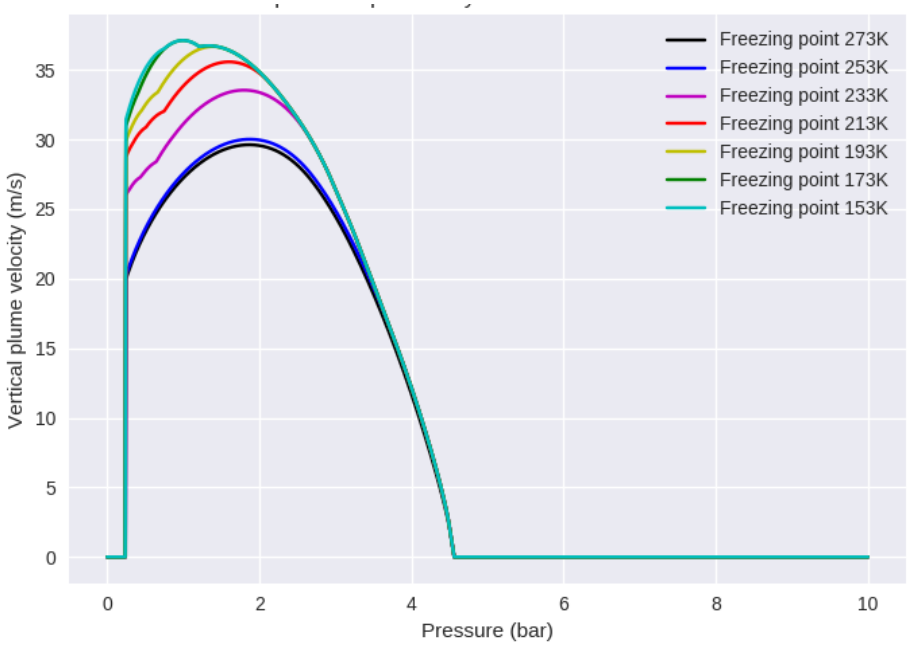
(a)


Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Vertical plume velocity (m/s)
Pressure (bar)
0
2
4
6
8
10
12

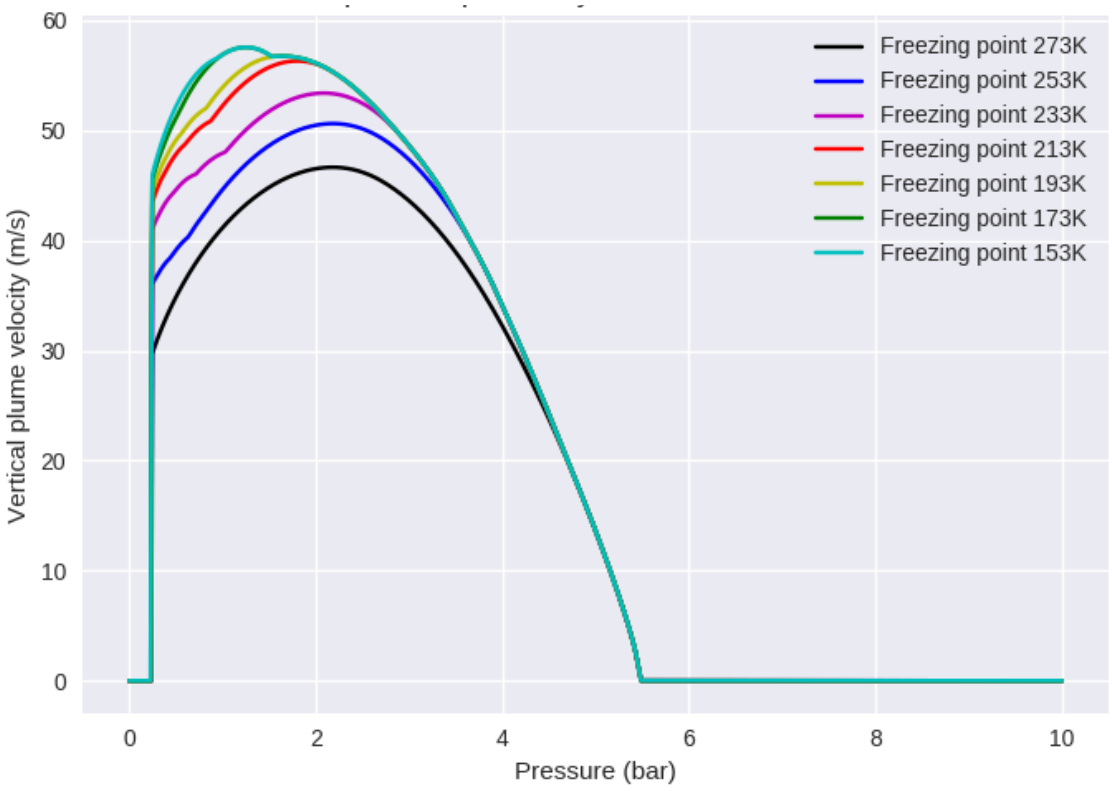
(b)


Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Vertical plume velocity (m/s)
Pressure (bar)
0
5
10
15
20

(c)

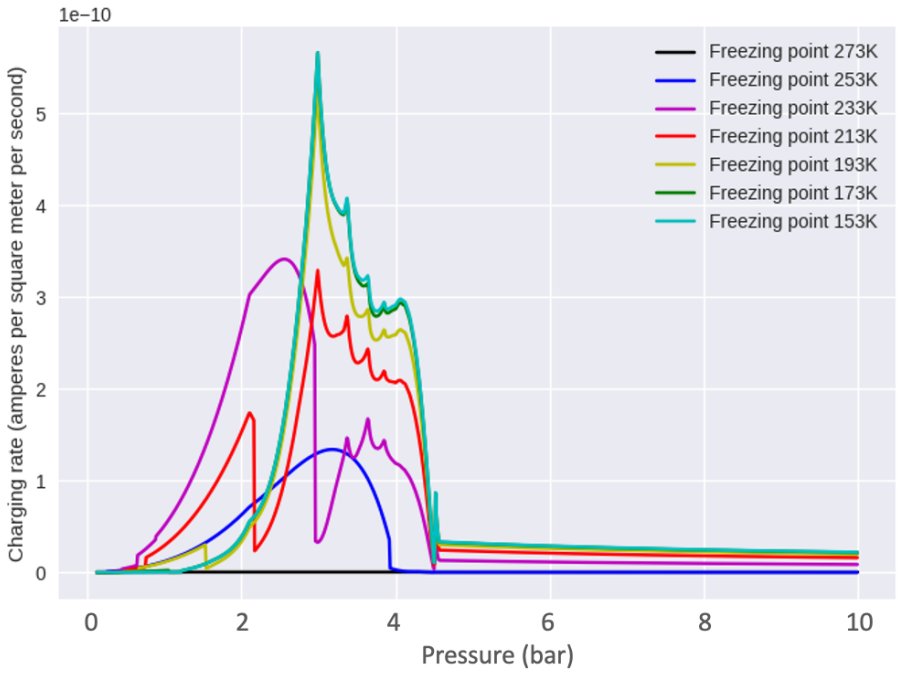

Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Vertical plume velocity (m/s)
0
5
10
15
20
25
30
35
0
2
4
6
8
10
Pressure (bar)

(d)

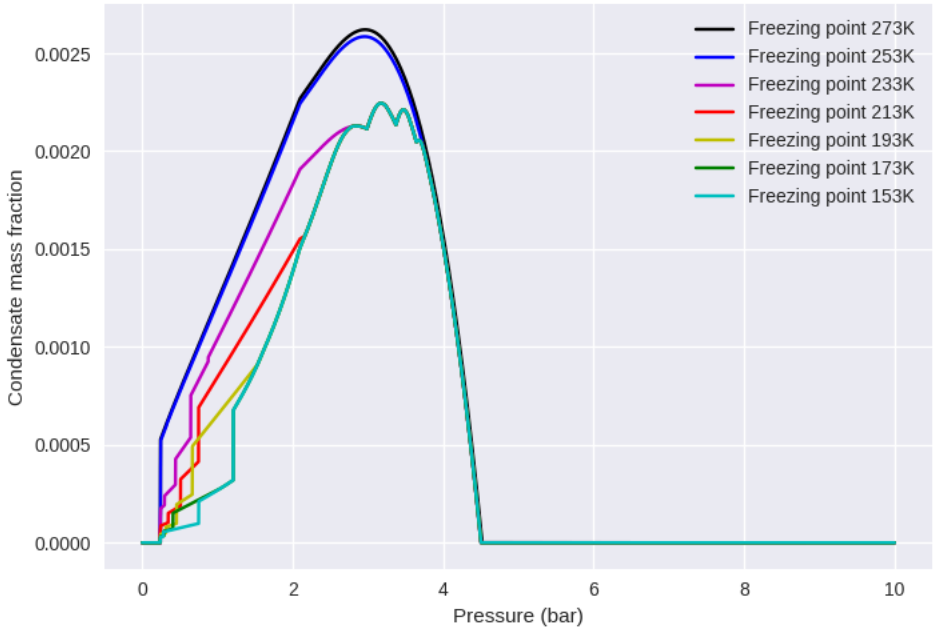


(e)

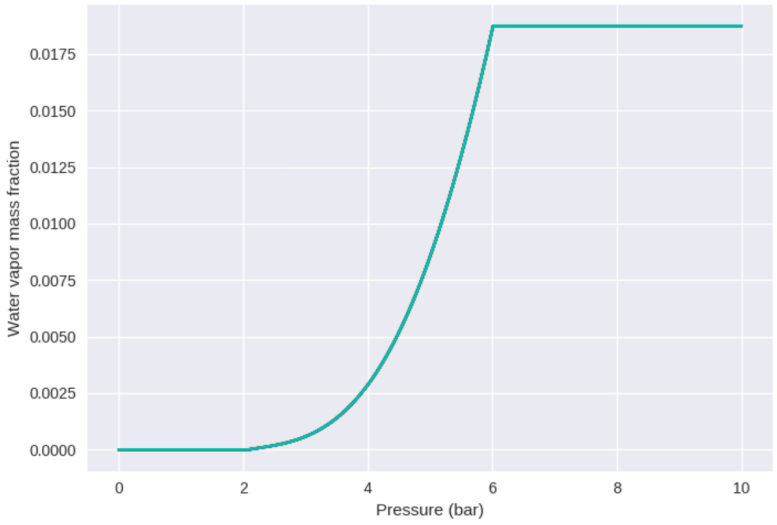
Figure 3.2: Velocity profiles for different choices of liquid solution freezing point; from top to bottom, (a) 0.1x, (b) 0.3x, (c) 1x, (d) 3x, and (e) 10x solar water. (Environment temperature 330 K at 10 bars.)


1e−10
Charging rate (amperes per square meter per second)
Pressure (bar)
Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K

(a)

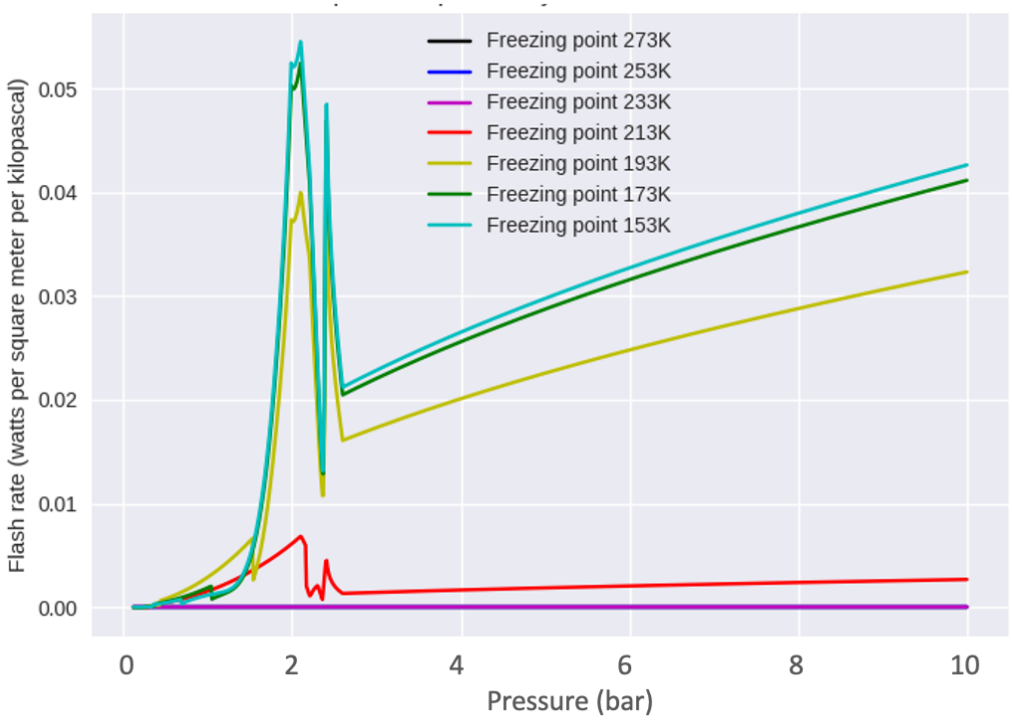

Condensate mass fraction
Pressure (bar)
Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K

(b)

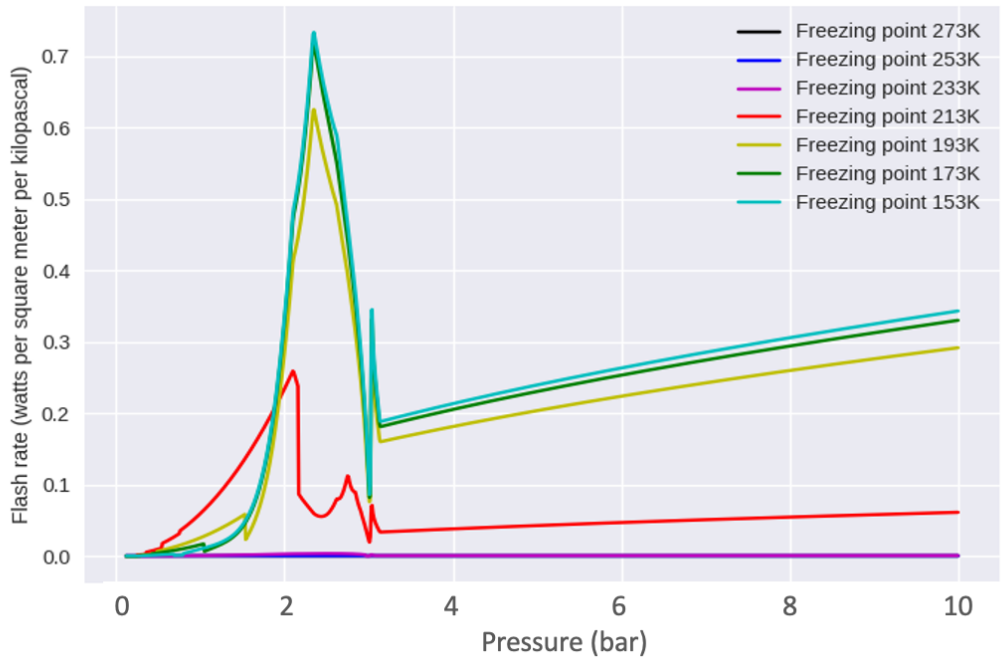


(c)

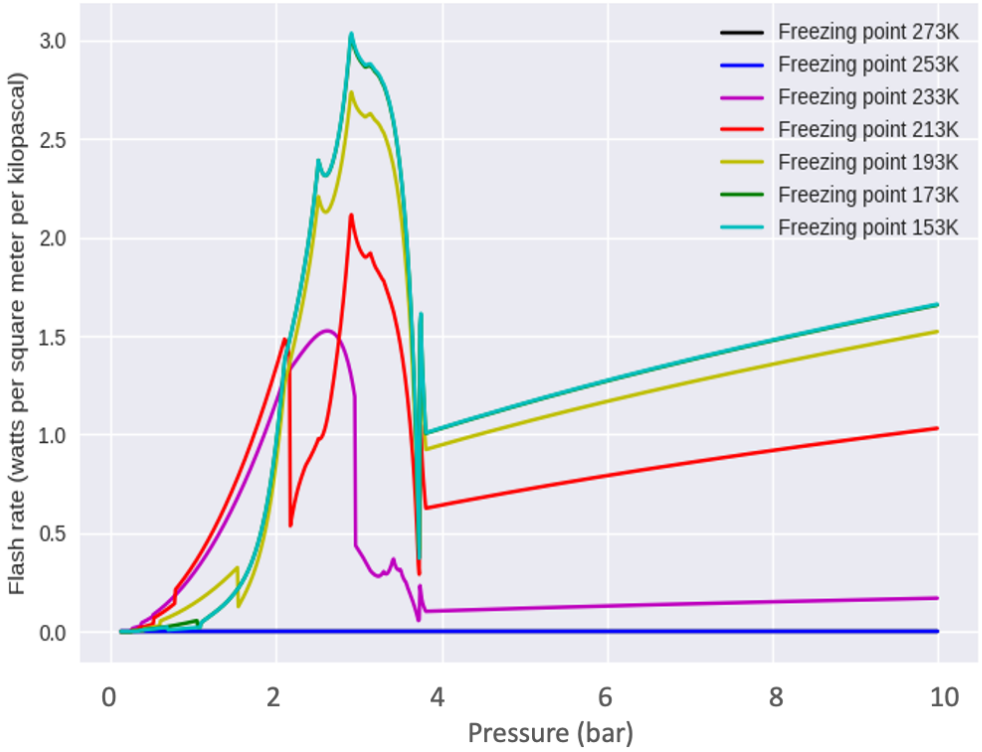
Figure 3.3: (a) Charging rate, (b) Condensate mass fraction, and (c) water vapor mass fraction . (Environmental temperature 330 K at 10 bars.)



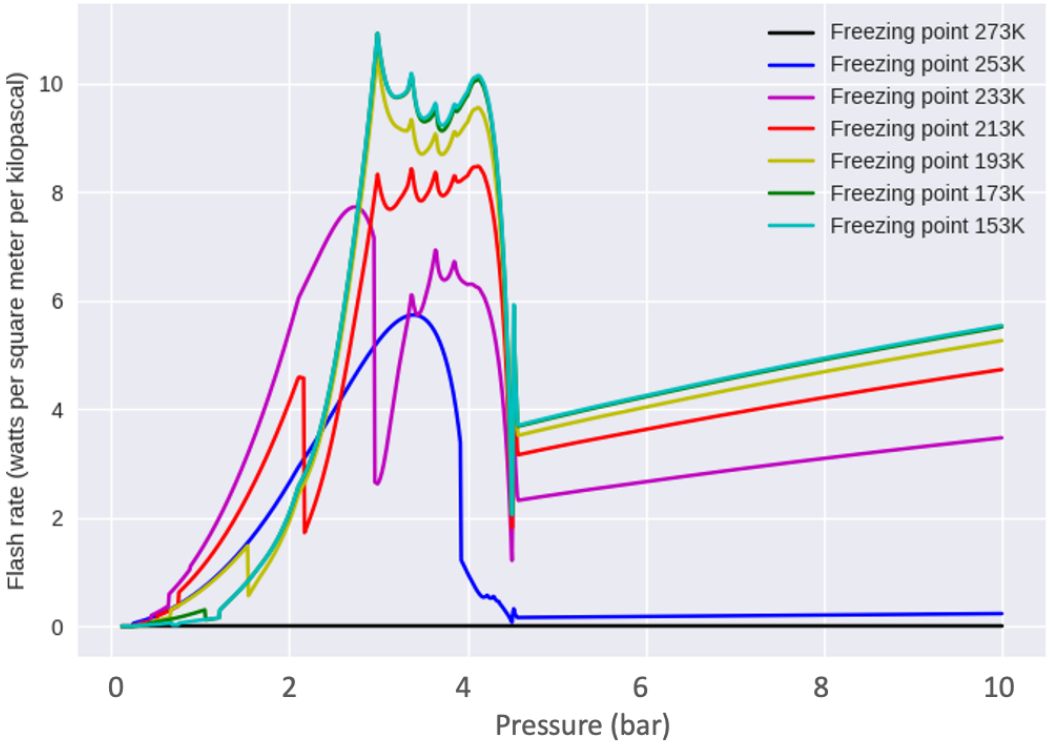
(a)


Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Flash rate (watts per square meter per kilopascal)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
2
4
6
8
10
Pressure (bar)

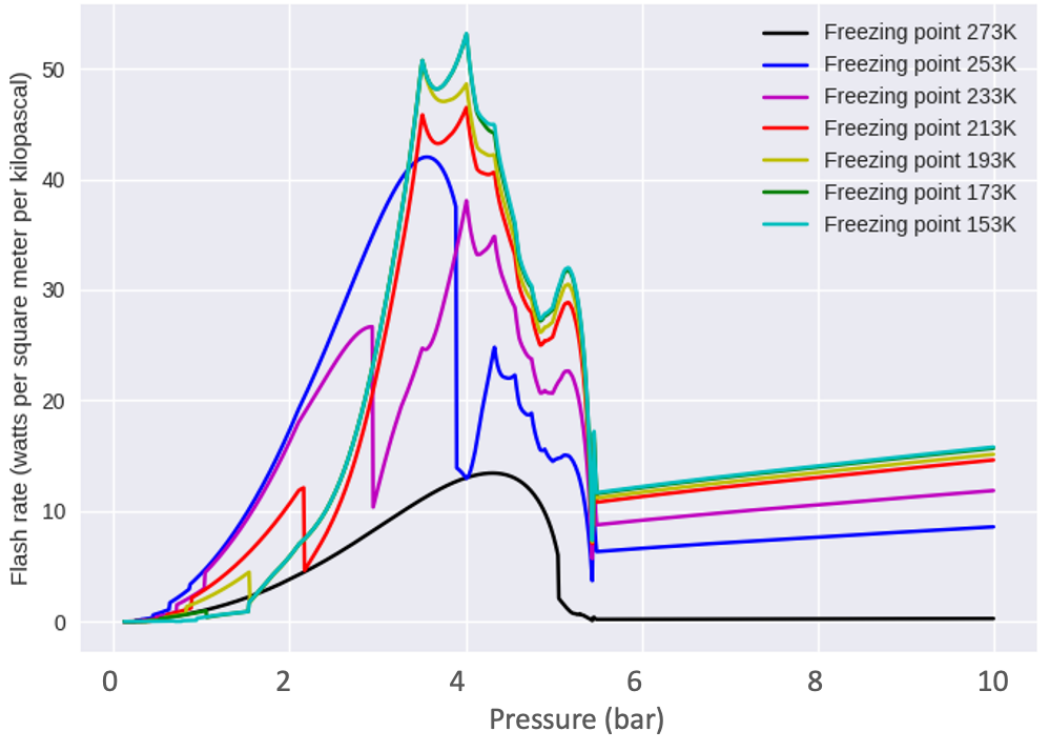
(b)


Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Flash rate (watts per square meter per kilopascal)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
2
4
6
8
10
Pressure (bar)

(c)

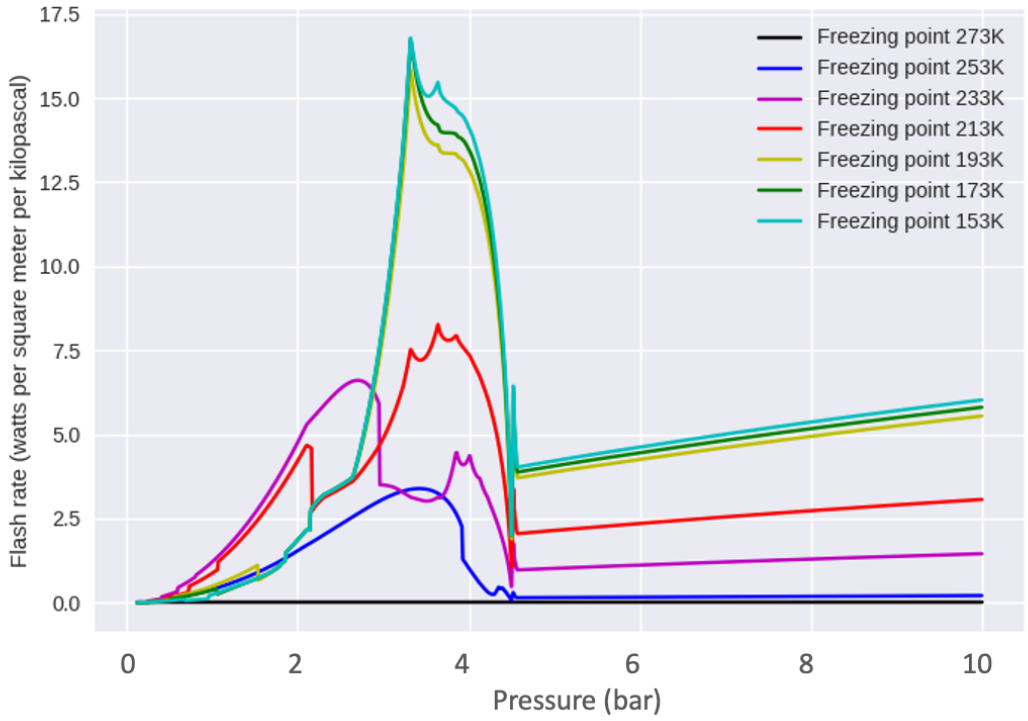

Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Flash rate (watts per square meter per kilopascal)
0
2
4
6
8
10
Pressure (bar)
0
2
4
6
8
10

(d)

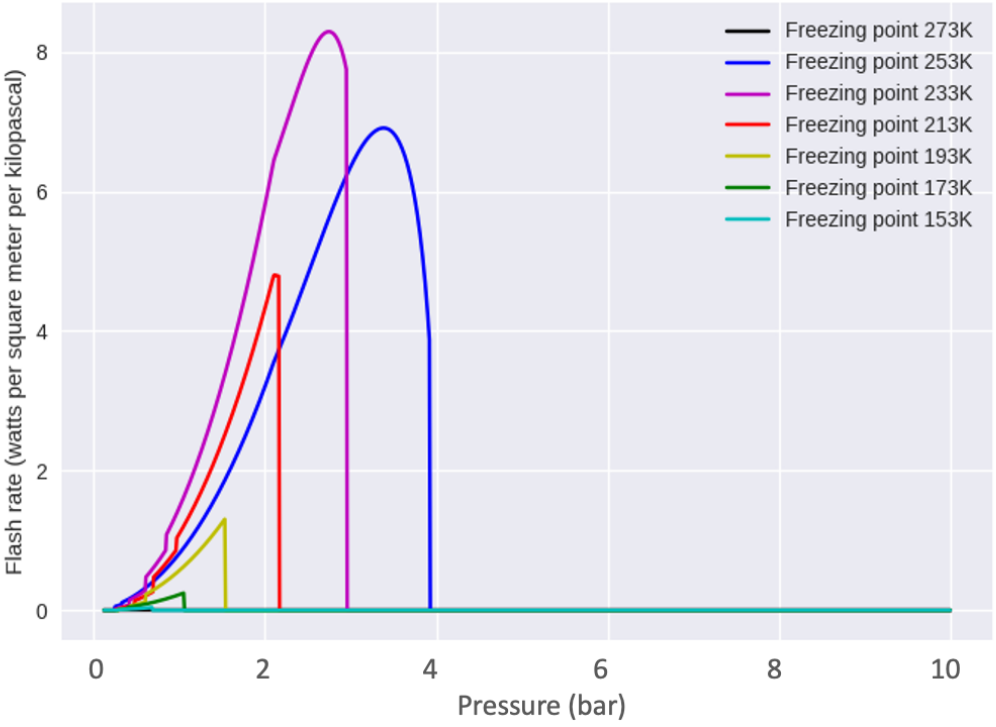


(e)

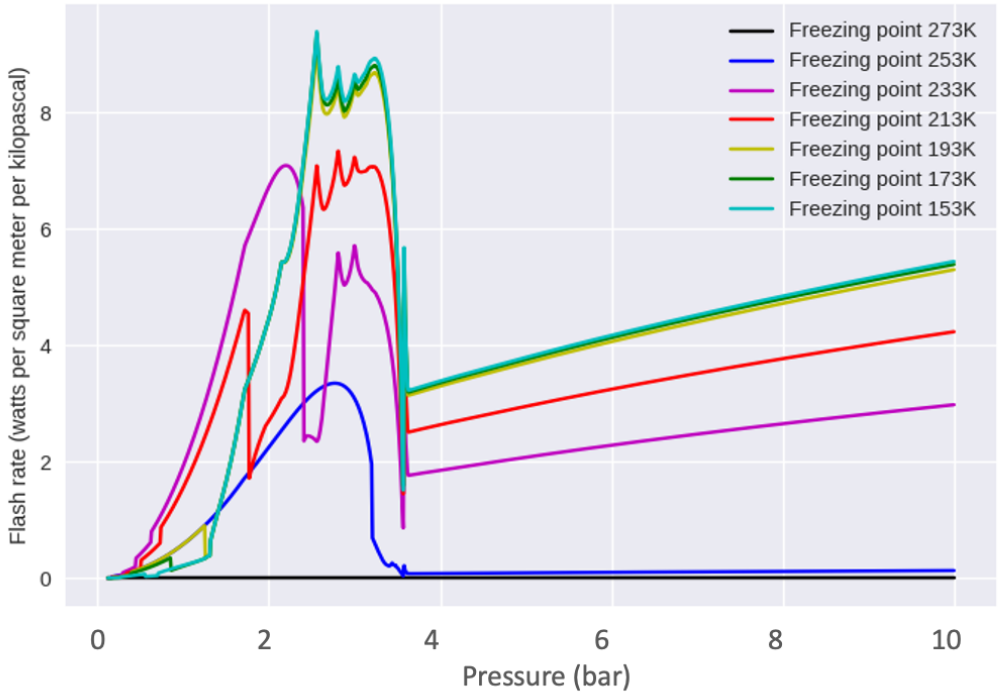
Figure 3.4: Flash rate profiles for different choices of liquid solution freezing point; from top to bottom, (a) 0.1x, (b) 0.3x, (c) 1x, (d) 3x, and (e) 10x solar water. (Environment temperature 330 K at 10 bars.)


Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Flash rate (watts per square meter per kilopascal)
Pressure (bar)
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
2
4
6
8
10

(a)

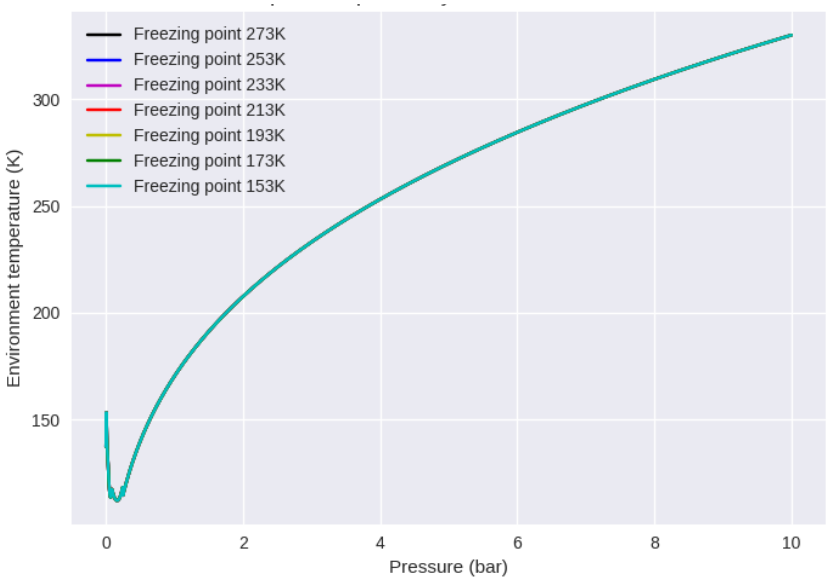

Freezing point 273K
Freezing point 253K
Freezing point 233K
Freezing point 213K
Freezing point 193K
Freezing point 173K
Freezing point 153K
Flash rate (watts per square meter per kilopascal)
Pressure (bar)
0
2
4
6
8
0
2
4
6
8
10

(b)

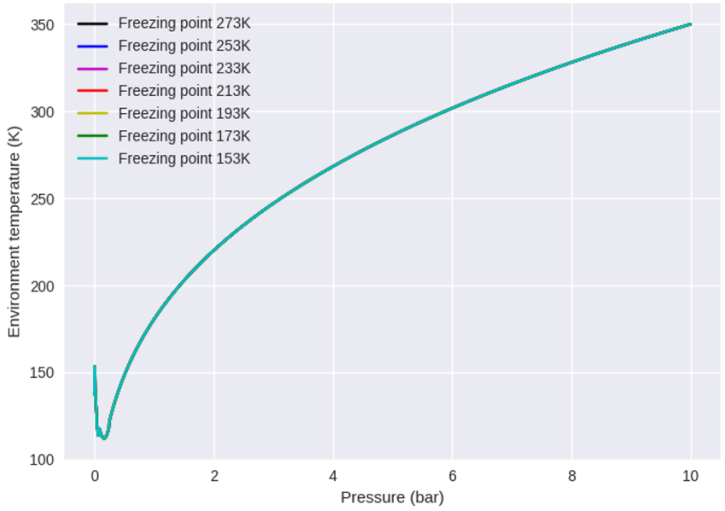


(c)

Figure 3.5: Flash rate profiles for alternate choices of temperature profile and liquid collision efficiency; from top to bottom, (a) 330K and 0.5, (b) 330K and 1.0, (c) 350K and 0.8. (3x solar water.)



(a)



(b)

Figure 3.6: Environment temperature profile; (a) 330 K at 10 bars profile, (b) 350 K at 10 bars profile

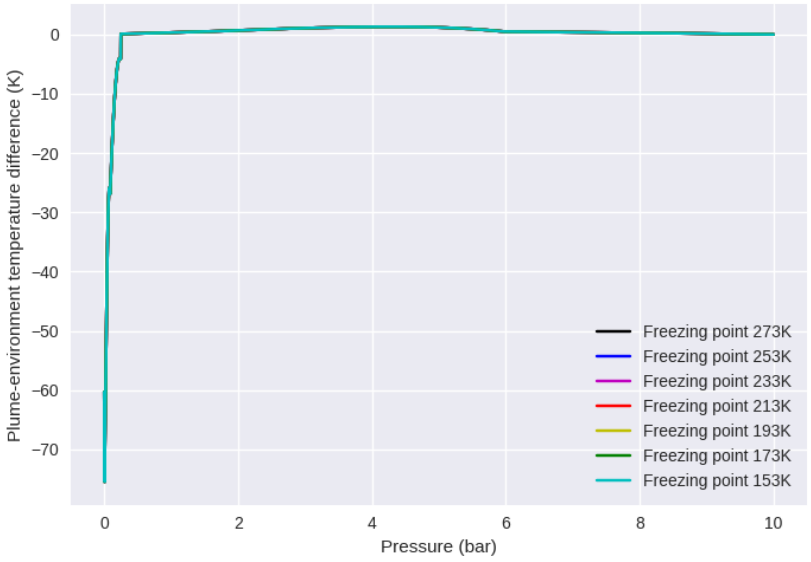


Figure 3.7: Profile of plume-environment temperature difference. (3x solar water, environment temperature 330 K at 10 bars.)

CHAPTER 4

LIGHTNING IN NON-IDEAL GASES

4.1 Introduction

Lightning has been detected on Saturn, Uranus, and Neptune by various spacecraft (Aplin *et al*. 2020), but there is not the same wealth of observational data as for Jupiter, nor a mission like Juno that is actively collecting new data. Saturnian lightning was detected first in the radio and later, despite major complications from ringshine, in optical wavelengths (Dyudina *et al*. 2010), but observations of Uranus and Neptune – including their lightning – are limited to the Voyager 2 flybys, which detected radio sferics but did not see optical lightning. Applying GEPE to these planets therefore is useful not only in the context of, as for Jupiter, constraining atmospheric properties but also in the context of future mission design, priorities for lightning science, and predictions about what may be observed in the future. Furthermore, the modifications to GEPE for this purpose – in particular, the addition of simulated evaporation of raindrops below the cloudbase – affect Jupiter results, specifically by relating the depth of the deepest lightning flashes to water abundance. Potential lightning depth profiles are discussed for Saturn, including a link to its observed phenomenon of infrequent giant storms, as well as for the ice giants. Overall, the model parallels observations quite well for Saturn, like for Jupiter, whereas for the ice giants the Voyager 2 observations are mysterious for reasons that are not model-specific.

Section 4.2 describes the parameters used in GEPE 0.5 for this work. Section 4.3 describes the output for the Jupiter recalculation, while section 4.4 does so for Saturn,

Uranus, and Neptune. Section 4.5 discusses the results for Jupiter, and section 4.6 does so for the other giant planets.

4.2 Choice of parameters

The model includes a number of parameters, such as molar masses, which are somewhat reliably known for Saturn but not for Uranus and Neptune. Additionally, several constants are retained from Chapter 3: the drag coefficient is 0.5, colliding water droplets will stick together 80% of the time and transfer charge the remaining 20%, and colliding ice particles will never stick together and always transfer charge. These parameter values were set in Chapter 3 as matching terrestrial thunderstorm lightning rates and altitudes when the model is applied to Earth's atmosphere; the changes made in this version of the model have minimal relevance for Earth conditions (pressures are low enough to use the ideal gas equation of state, while rain evaporation is limited by the presence of a solid surface), and the calibration therefore retains its relevance here.

Plume radius is set to 5 kilometers, as for Jupiter. The stratospheric temperature profiles for Saturn, Uranus, and Neptune of Lindal (1991) are used at pressures of less than 0.25 bars. The remaining free parameters are water abundance and temperature at the model base, along with the choice of the pressure level at which to set the said base, and the water freezing point, which is depressed relative to 273 K due to supercooling and dissolved ammonia; Earth calibration is best fit by a freezing point depression of 10-20 Kelvin, due to supercooled liquid.

Unlike Jupiter and Saturn, Uranus and Neptune have sufficient enrichment of methane to substantially affect the molar weight of the dry atmosphere. The methane enrichment is known only approximately from observation (Atreya *et al*. 2020); we use the composition of the non-water atmosphere of Wiktorowicz and Ingersoll (2006), with a molar weight of 5.685 grams per mole.

The maximum pressure level was set at a depth so that the bulk of precipitation evaporates before reaching it. At Saturn, 30 bars was chosen; for all water abundances considered, no precipitation reached further than 28 bars. For Uranus and Neptune, the situation is more difficult: there is substantial uncertainty in deep atmospheric composition, and due to the high water abundance, the cloudbase is deep. This not only strains the limits of the Van der Waals equation of state, but it also requires a large pressure step size, decreasing accuracy. Furthermore, for particularly cold and high-water temperature profiles, there may be no conventional cloudbase, but rather an 'ocean surface' below which the single phase is continuous with condensed water rather than hydrogen gas, although this is unlikely (Wiktorowicz and Ingersoll 2005). We set the model base for both ice giants at 2000 bars, which for most choices of the water abundance is well below the cloudbase, and do not consider temperature-composition combinations with a deeper cloud base. Evaporating precipitation can fall well below the cloud base, at most roughly 100 bars, but compared to the total depths and uncertainties involved here this is small, and for the combinations we consider, all precipitation evaporates before reaching the model bottom.

For the oxygen abundance of Saturn, we use the constraint of Visscher and Fegley (2005) of 1.9 to 6.1 times solar abundance and extend the uncertainty intervals, considering oxygen abundances of 1.5x solar, 2.5x solar, 4.5x solar, and 7.5x solar.

The last of these values is in line with the observed Saturnian carbon abundance from methane. For Uranus and Neptune, uncertainties are much higher; from densities they may be composed primarily of ices, but ices can be replaced with a mixture of rock and hydrogen without a change in density. In reality, Uranus and Neptune are expected to have all three, but the proportions are unclear. For both, we consider abundances of 32x solar, 60x solar,80x solar, 100x solar, and 128x solar, in terms of mass fraction (corresponding to enrichments of approximately 70x solar, 160x solar, 240x solar, 340x solar, and 560x solar if expressed in terms of O:H ratio). This corresponds to water mass fractions (out of hydrogen, helium, and water) of 20%, 37.5%, 50%, 62.5%, and 80%, covering the range of estimates; for comparison, Wiktorowicz and Ingersoll (2006) reach an estimate of 53.8% water by mass, while Moses et al. (2020) estimate an O:H ratio that is <45x solar for Uranus and 250x solar for Neptune.

The freezing point of water is set in 20-Kelvin intervals between 273 K (the freezing point of pure water) and 153 K (the ammonia-water eutectic is at 177 K, with this temperature adding supercooling on the upper edge of the best fit for Earth, to capture the widest range of plausible freezing points).

Temperature choices were based on Leconte *et al*. (2016), for conditions without a stable radiative layer (because such a layer is not present in the model). For Saturn, temperatures of 360K and 380K at 30 bars were considered; the latter more closely matched observed upper tropospheric temperatures. For Uranus and Neptune, temperatures were selected so that the upper atmosphere would match observations and Leconte *et al*. (2016); this led to temperatures at 2000 bars of 960K for Uranus and 940K for Neptune.

The electric field E_l at which electrostatic levitation becomes significant is set based on a charging model that includes levitation, run for the particle distributions at 40 bars, 100 bars, 250 bars, and the cloudbase for various water abundances. For both Uranus and Neptune, the electric field stabilized between $5*10^6$ newtons per coulomb and $1.5*10^7$ N/C in most cases, so E_l was set to 10^7 N/C for both.

The model's sensitivity to oxygen abundance and water freezing point (that is, physically, ammonia content) is the main focus of the results below. Of the other parameters, the primary effect of changing base temperature is to raise or lower the cloudbase, and thus compress or stretch the entire flash rate profile, but these changes are generally smooth. Changing the percentage of time that water and/or ice particles stick together upon collision, as opposed to transferring charge, can have a substantial effect on model predictions, the liquid coefficients especially. The most likely source of error is plume radius: it controls entrainment of dry air and therefore overall vigor of convection, and is not set by calibration with Earth. However, while changing plume radius has a substantial effect on the flash rate, it achieves this primarily by altering the strength of convection, and constraints are thus given by observations (the plume must be large enough to cause convection, but not so large as to escape the atmosphere).

4.3 Jupiter comparison and results

We tested the model by running it for Jupiter and comparing with earlier results. While the altered equation of state does not produce substantively different flash rates for Jupiter, the addition of evaporation does produce a substantial effect . As seen in

Figure 4.1, flash rates rapidly decrease below the cloudbase. For water abundances up to 3x solar, the precipitation evaporates fully before reaching the 10-bar pressure level. For a 330K temperature at the 10-bar pressure level, flash rates go to zero deeper than 4.5 bars for 0.3x solar water, 6.5 bars for 1x solar water, and 9 bars for 3x solar water, regardless of ammonia admixture. The fraction of lightning power at pressures of 0-4 bars depends on the amount of freezing point depression, but ranges from 99% to 100% for 0.3x solar water, from 70% to 100% for 1x solar water, from 36% to 100% for 3x solar water, and from 12% to 100% for 10x solar water. These ratios are comparable to earlier results for 3x solar and 10x solar water, but for 1x solar or subsolar water they imply substantially less deep lightning than concluded in Chapter 3 due to evaporation below the cloudbase.

A single point where all liquid instantaneously freezes is of course a simplification; several Jupiter model runs were performed with a gradual phase transition, with a linear change in density and charging efficiency over a range of temperatures. The results showed behavior similar to that at a single freezing point, intermediate between the center and cold endpoint of the temperature range. For instance, at 1x solar water abundance and gradual freezing between 273K and 193K, total flash rates are 374 W/m^2, close to the 388 W/m^2 for sharp freezing at 233K, and similar behavior is observed for higher water abundances. At 0.1x and 0.3x solar water, however, the gradual phase transition enables higher lightning rates than a sharp transition at 233K (though still less than a sharp transition at 193K) by enabling some liquid to persist into the upper atmosphere where convection is occurring and thus allowing substantial particle growth.

4.4 Saturn, Uranus, and Neptune results

Entraining plumes form on Saturn as well as Jupiter, but have notably lower velocities for the same water abundance. For 1x solar water, a plume on Jupiter acquires a maximum velocity of 20-27 m/s, and for 3x solar water this reaches 35-42 m/s, while for 0.3x solar water it is reduced to 10-14 m/s. A plume on Saturn for 1.5x solar water reaches a maximum velocity of 10-14 m/s, and for 7.5x solar water 24-28 m/s. That is, the convection on Jupiter is as vigorous (as measured by upwelling velocity) as the equivalent would be on Saturn with five to seven times higher water abundance. Nevertheless, lightning is generated on Saturn at rates that equal or surpass those on Jupiter for similar water abundance. Saturnian flash rates are shown in Table 4.1 and Figure 4.2. For a 233K freezing point, overall power output goes from 5000 W/m^2 for 1.5x solar water to 43000 W/m^2 for 7.5x solar water. This is greater than the predicted flash rates for the same water abundance on Jupiter, especially for lower water abundances. Furthermore, as shown in Table 4.2, when compared to similar water abundances for Jupiter, less freezing point depression is required to generate lightning. This is because, while Saturn's atmosphere is overall colder than Jupiter's, the cloudbase is defined by the point where the saturation vapor pressure curve intersects a scaled adiabat (representing the atmospheric water vapor pressure), with the former being steeper as a function of pressure; shifting the adiabat to lower temperatures moves the intersection to both higher temperatures and pressures s. Thus, for a given water mass fraction, Saturn’s cloudbase and convection occur at higher temperatures than on Jupiter, allowing substantial particle growth even at near-solar water abundance. Nevertheless, for a 273K freezing point without any freezing point depression or supercooling, no lightning occurs for 1.5x solar water and only limited lightning (190 W/m^2) at 2.5x solar water, while other parameter combinations generate from 2500 W/m^2 to 43000 W/m^2.

For 1.5x solar water, Saturn's lightning power is concentrated at pressure levels of 3 to 12 bars. As seen in Table 4.1, higher water abundance deepens the cloudbase, and therefore also the deep end of this distribution; for 7.5x solar water, substantial lightning extends down to 20-21 bars. On the shallow end of this distribution, lightning power at 0-3 bars is a small fraction of total lightning power for any water abundance and freezing point, with a maximum of 66 W/m^2 for 7.5x solar water and a 233K freezing point. Lightning power at 3-6 bars, however, is substantial; outside of the two scenarios with minimal lightning mentioned above, it ranges from 280 W/m^2 to 2700 W/m^2. In both of these ranges, lightning power changes non-monotonically and relatively little with water abundance. Meanwhile, this 'relatively shallow' lightning power decreases with increased ammonia fraction (decreasing freezing point). This effect is not prominent on Jupiter because freezing point depression is required for substantial particle growth, but on Saturn the cloudbase is deep enough that particle growth occurs regardless. Two effects then favor decreased lightning for high ammonia content on Saturn. First, ice has a higher charging coefficient than water, leading to more lightning for the same particle size distribution. Second, because upwelling velocities on Saturn are lower than on Jupiter, particle growth enabled by freezing point depression largely leads to particles large enough to precipitate; as such, greater freezing point depression decreases the total cloud particle content in the shallow atmosphere.

On Uranus and Neptune, unlike Jupiter and Saturn, freezing point depression has only limited effect because the bulk of both convection and lightning happens well below the pressure level corresponding to 273 K. As such, only the endmember freezing points of 273K (pure water) and 153K (ammonia -water eutectic with supercooling)

were considered, since any other ammonia-water mixture would have a freezing point between those bounds. The resulting convection behavior is shown in Figure 4.3: large water content leads to buoyancy being achieved only a large distance above the cloudbase, but potentially very strong convection where it does occur, with upwelling speeds of more than 250 m/s in the extreme case, although this assumes that a temperature contrast is built up all the way from the cloudbase rather than evolving as multiple vertical cells. Even for 32x solar water, this leads to extreme flash rates - for Neptunian gravity and a 940K basal temperature, total lightning power is approximately 2.5 megawatts per square meter, and this increases with water abundance. As shown in Table 4.3 and Figure 4.4, much of this is generated at depths of over 100 bars, but substantial lightning is generated at shallower depths as well. Hundreds of kilowatts in lightning power per square meter can be released at pressures of 0-30 bars, much more than on Jupiter or Saturn. Since lightning power cannot exceed overall power from convection, such strongly convective regions would be rare.

4.5 Jupiter discussion

Table 4.4 lists lightning observations on the giant planets. On Jupiter, Galileo detected deep lightning flashes to at least the 8-bar pressure level (Little *et al*. 1999, Dyudina *et al*. 2002), which would imply at least a 3x solar water abundance at these locations. Combining the Galileo deep lightning and Juno shallow lightning rates while accounting for the difference in exposure time implies, from a very small sample size, that roughly 30% of visible lightning energy is released at 0-4 bars, and 70% at 4+ bars; this would correspond to a water abundance of 3-6 times solar. This provides some tenuous evidence for 3x solar or higher water abundance in Jovian

storms, though this is consistent with lower abundances elsewhere on the planet, since ‘wetter’ regions will have more convective activity and lightning. For comparison, observations with the Juno microwave radiometer have produced 1-5x solar water in a narrow equatorial region of Jupiter (Li *et al*., 2020), with work to derive a “global” water abundance continuing.

It must, however, be noted that this model does not consider hail, formed by precipitation being caught in repeated updrafts and continuing to grow. Similarly, mushballs (Guillot *et al*. 2020) are not incorporated into the model. It is certain that hailstones larger than those produced in the model will occur on Jupiter; for them to cause deeper lightning, however, they must make up a significant fraction of Jupiter's total precipitation flux. The model also does not consider collisional ice splintering, which may significantly affect the ice particle size distribution (Sullivan *et al*. 2018, Qu *et al*. 2020); this process would lead to a greater amount of smaller particles, and therefore to faster evaporation.

An additional caveat is provided by the fact that only one of the flashes Galileo detected was confirmed by Dyudina *et al*. (2002) to be at a depth of 8+ bars; the other five flashes are consistent with pressures of 2-6 bars, although they may also be deeper. Lightning within this pressure range is predicted even for solar water abundance. As such, it is possible that the deepest flash was generated by some other process, or alternatively that it occurred in conditions where the cloud tops were elevated unusually far into the stratosphere and thereby caused a higher observed depth from the topmost scattering layer rather than actually being unusually deep. It is also possible that the deepest flash was actually a combination of several nearby flashes over the 6.4-second exposure time.

4.6 Saturn, Uranus, and Neptune discussion

Unlike on Jupiter, lightning on Saturn was not observed optically until Cassini, and even then with difficulty (Dyudina *et al*., 2010). Radio emissions from lightning have been observed before and since, but they too indicate less frequent lightning than on Jupiter. Our numerical model predicts similar lightning power on Saturn as on Jupiter, if the atmosphere is convecting. Observations of Saturn, however, are consistent with intermittent convection, with large storms sweeping through every several decades but otherwise a less feature-dense atmosphere than Jupiter. Furthermore, atmospheric modeling also supports intermittent convection (Li and Ingersoll, 2015), and lightning observations show that, at any given time, there is usually only one location on Saturn - one storm - that exhibits lightning flashes (Fischer *et al*., 2011). As such, it is entirely consistent with observations that when lightning occurs on Saturn, it does so with equal or greater intensity to Jupiter. Indeed, during the 2011 giant storm, Dyudina *et al*. (2013) detected flash rates 20x greater than Little *et al*. (1999) found for Jupiter storms, although a comparison per unit area is difficult to make because of the different structures of these storms.

The optical flash observations of Dyudina *et al*. (2010) and Dyudina *et al*. (2013) mostly occurred at similar depths, somewhere between 125 to 250 km below the cloud tops with model-based uncertainty. If the cloudtop estimate from Dyudina *et al*. (2013) of 1.2 bars is used, this corresponds to a pressure somewhere between 6 and 20 bars, in line with model predictions. Cassini was unable to detect lightning flashes with optical energy less than $2*10^8$ J on the nightside or $1*10^9$ J on the dayside, and found that the frequency of flashes increased with decreasing energy up to this

detection limit, implying additional small flashes below the detection limit; since the breakdown field increases with density, this may be a shallower population of flashes. The flash with the lowest energy observed also had a very sharp brightness distribution, with a corresponding depth of 62-125 km below the cloud tops, 3-6 bars; Dyudina *et al*. (2013) note this observation as questionable, as it could have been a gamma ray strike, but if real it provides evidence towards such a shallow lower-energy lightning population. The direction of this difference would parallel the Juno observations on Jupiter. Unlike Jupiter, however, we predict that this distribution does not continue further to lightning at the 1-2 bar level.

Ammonia or other antifreeze is not required to cause lightning on Saturn; its presence is consistent with observations, but unlike Jupiter, lightning is not evidence that Saturn's water clouds have an ammonia admixture. For Uranus and Neptune, an ammonia admixture has even smaller influence, but does decrease the flash rate at shallow levels at temperatures below 273K, due to ice being substantially more effective than water at charging.

The massive flash rates predicted for Uranus and Neptune rely on an assumption of simple convection that may break down for the massive depths involved in these planets' water clouds. Additionally, electrification of dense molecular hydrogen may not follow the low-pressure trend of a linearly increasing breakdown field. If massive plumes spanning the entire water cloud do exist, they may be expected to be stable over a fairly long span of time given the distances involved; as such, while the time it takes to generate lightning is longer for these higher pressures, lightning should eventually occur. An additional assumption is that the background temperature

profile is a dry adiabat; a subadiabatic profile is possible if there is additional mixing, and a superadiabatic profile has also been suggested (Guillot 1995).

Voyager 2 detected radio evidence of lightning at both Uranus and Neptune, although fewer and weaker signals than for Saturn, but no optical signal (Aplin *et al*. 2020). Lightning sufficiently deep in the water cloud would not be optically detectable. The maximum depth to which lightning can be detected optically is unclear, but Dyudina *et al*. (2013) suggest a 1:1 correspondence of optical and radio detections on Saturn is consistent with observations, implying that all lightning on Saturn is shallow enough for optical detection (given suitable observing conditions). In that case, Uranian and Neptunian lightning down to at least 12-20 bars should be optically detectable. For sufficiently vigorous convection, substantial flash rates are predicted. However, the weak radio signals are inconsistent with such a flash profile, which would effectively be a distribution similar to Saturn but with 10x higher flash rate. The weakness of the signals implies that either the flashes are happening at a very shallow pressure level (in the ammonia clouds), they are of small horizontal extent for unclear reasons, or their depth is sufficient to attenuate the radio signal. Additionally, Voyager 2 had clearer detection of lightning on Uranus than on Neptune. For the same temperature and composition, Uranus's lower gravity tends to lead to lower lightning rates than Neptune in the 0-200 bar region and similar lightning rates below. Uranus is also believed to have a somewhat warmer deep atmosphere than Neptune; this effect raises the cloudbase and intensifies convection, leading to higher 'shallow' (in this case 0-200 bar) flash rates but substantially lower deep flash rates, and lower flash rates overall.

These distributions suggest two possible origins of ice giant lightning, which could be distinguished by optical observation. If the sferics detected by Voyager 2 came from

the shallow or middle atmosphere, even at the 20-bar pressure level, a future Uranus orbiter may be able to detect optical lightning. The convective systems causing this lightning would be those of the ammonia cloud, though there may be ice and ammonia-water particles involved in electric field generation. If, by contrast, they came from the deep atmosphere and the water cloud, it is likely that the observed signal strength is detected after substantial attenuation. In this case, Neptune's few observed sferics are more attenuated because of Neptune's colder temperature profile and a deeper water cloud, whereas Uranian lightning is easier to observe because it is shallower. Nevertheless, both ice giants would be exhibiting lightning in an extreme high-pressure regime quite unlike anything seen on Earth, Jupiter, or Saturn.

The plausibility of such attenuation depends on the absorptivity of ice giant atmospheres, which is not well-understood. The sferics on Uranus and Neptune were detected mostly in the HF band, at frequencies of 0.9 to 40 megahertz for Uranus and 20 to 30 megahertz for Neptune. Uranian discharges were measured as being approximately 10x (10 dB) weaker than Saturnian, and Neptunian 45x (16 dB) weaker than Uranian (Zarka and Pedersen 1986, Aplin *et al*. 2020). The absorptivity of a giant planet atmosphere in this radio wavelength range is expected to be dominated by water and ammonia, both in vapor and in cloud form. The absorption behavior of water under these high-pressure conditions is not directly known, although in general, in the low-frequency limit, water absorption should be proportional to the square of the frequency. Segelstein (1981) estimates an absorption coefficient of liquid water, at 30 MHz, of 0.0087 m-1, corresponding to a 10x decline in intensity (10 decibels signal loss) over 265 meters. For ice, absorption at radio wavelengths is strongly dependent on temperature, but signal loss in the Antarctic (at ~10 MHz) is approximately 10 decibels in 400-500 meters (Macgregor *et al*., 2007), with estimates for colder Europa

ice centering around 10dB loss in 1000-3000 meters (Moore 2000). These can be converted to water pressure for an equivalent mass of vapor, with absorption of 10 dB in 250-500 meters water equating to 25-50 bars water, which for a 50% water (by mass) atmosphere is 50-100 bars of total pressure, comparable to the depths and depth contrasts predicted for Uranus and Neptune.

For water vapor, one must extrapolate from higher-frequency data. For example, the equation for water vapor self-interaction of Rosenkranz (1998) predicts very high attenuation; for 10% (by mass) water vapor at 100 bars pressure and 400K, for example, the projected attenuation is approximately 900 dB per meter gas, effectively totally opaque. More recent experiments predict greater transparency, but there remains substantial uncertainty about its degree, as the experimental data is at lower pressures and water contents than the Uranian and Neptunian clouds and therefore requires extrapolation. The equations of Karpowicz and Steffes (2011) can be extrapolated to 10dB absorption over 47 km under the above conditions, substantial but not total, while follow-up modeling by Bellotti *et al*. (2016), despite mostly using the same dataset, yields only 10dB absorption over 2600 km, functionally transparent at the 100-bar level. Janssen *et al*. (2017) predict even less absorption, with water and ammonia becoming near-transparent in the radio at pressures above 100 bars, and estimate that the Juno MWR's channel 1 (600 MHz) should see down to depths in the thousands of bars. Further complicating affairs, lightning emissions are absorbed and/or scattered by cloud particles as well as by gas; in the optical wavelengths, this is used to quantify lightning on Earth (Goodman *et al*. 2013, Peterson 2019) and on the giant planets (Little *et al*. 1999, Dyudina *et al*. 2010, Dyudina *et al*. 2013, Becker *et al*. 2020). Radio waves are not as strongly absorbed, but Earth-based radio observations at 20 MHz can be substantially impeded by rain, despite its small size

compared to the wavelengths involved (Mondal and Bhattacharya 2015), showing that cloud absorption is also likely to be significant for Uranus and Neptune. As such, the possibility that Voyager 2 detected strongly attenuated deep lightning at the ice giants remains plausible, but uncertain. A future lightning campaign at Uranus with simultaneous optical and microwave/radio observations could address this ambiguity.

Table 4.1: *Saturn flash rates (W m^{-2}) for 253K freezing point*

	1.5x solar water	2.5x solar water	4.5x solar water	7.5x solar water
0-3 bars	22	27	16	51
3-6 bars	490	1159	2147	2519
6-9 bars	970	1832	4364	6603
9-12 bars	1988	4629	2275	8543
12-15 bars	0	2893	1582	3012
15-18 bars	0	0	190	5419
18-21 bars	0	0	0	626
21-24 bars	0	0	0	0

Table 4.2: *Comparison of total flash rate (W m^{-2}) for the same water abundance, Jupiter vs Saturn*

Water freezing point & abundance	Jupiter	Saturn
273K, 1x solar	0	0
253K, 1x solar	0	569
213K, 1x solar	594	2338
153K, 1x solar	725	1379
273K, 3x solar	0	1137
253K, 3x solar	2780	14566
213K, 3x solar	4442	7200
153K, 3x solar	5184	7174

Table 4.3: *Uranus and Neptune flash rates (W m^{-2}) for 273K freezing point*

Planet & water abundance	0-30 bars	30-100 bars	100-200 bars	200 bars and below
Uranus, 32x solar	205 395	981 721	585 355	174 457
Uranus, 60x solar	395 191	2 081 595	539 638	462 742
Uranus, 80x solar	471 238	2 770 892	98 373	267 172
Uranus, 100x solar	658 631	2 840 410	93 699	551 845
Uranus, 128x solar	1 499 758	3 707 309	4 659 038	112 433 514
Neptune, 32x solar	272 712	1 104 474	907 497	207 545
Neptune, 60x solar	536 232	2 263 682	1 039 991	575 677
Neptune, 80x solar	689 999	3 176 756	330 609	407 906
Neptune, 100x solar	511 446	3 506 926	28 046	131 839
Neptune, 128x solar	1 669 778	2 455 844	2 959 815	54 575 369

Table 4.4: *Selected giant planetary lightning observations, including all those used for comparison with model output*

Planet	**Spacecraft**	**Instrument**	**Wavelength**	**Depth**	**Reference**
Jupiter	Voyager 1&2	ISS	~500 nm	~5 bar	Cook *et al*. (1979)
Jupiter	Voyager 1&2	PWS	~100 km		Gurnett *et al*. (1979)
Jupiter	Galileo probe	LRD	~100 km		Rinnert *et al*. (1998)
Jupiter	Galileo	SSI	~500 nm	6-8 bar	Little *et al*. (1999)
Jupiter	Juno	SRU	~500 nm	2-4 bar	Becker *et al*. (2020)
Jupiter	Juno	MWR	50 cm		Brown *et al*. (2018)
Saturn	Voyager 1&2	PRA	~10 m		Burns et al. (1983)
Saturn	Cassini	ISS	~500 nm	6-20 bar	Dyudina *et al*. (2013)
Uranus	Voyager 2	PRA	~20 m		Zarka & Pedersen (1986)
Neptune	Voyager 2	PRA	~10 m		Kaiser *et al*. (1991)

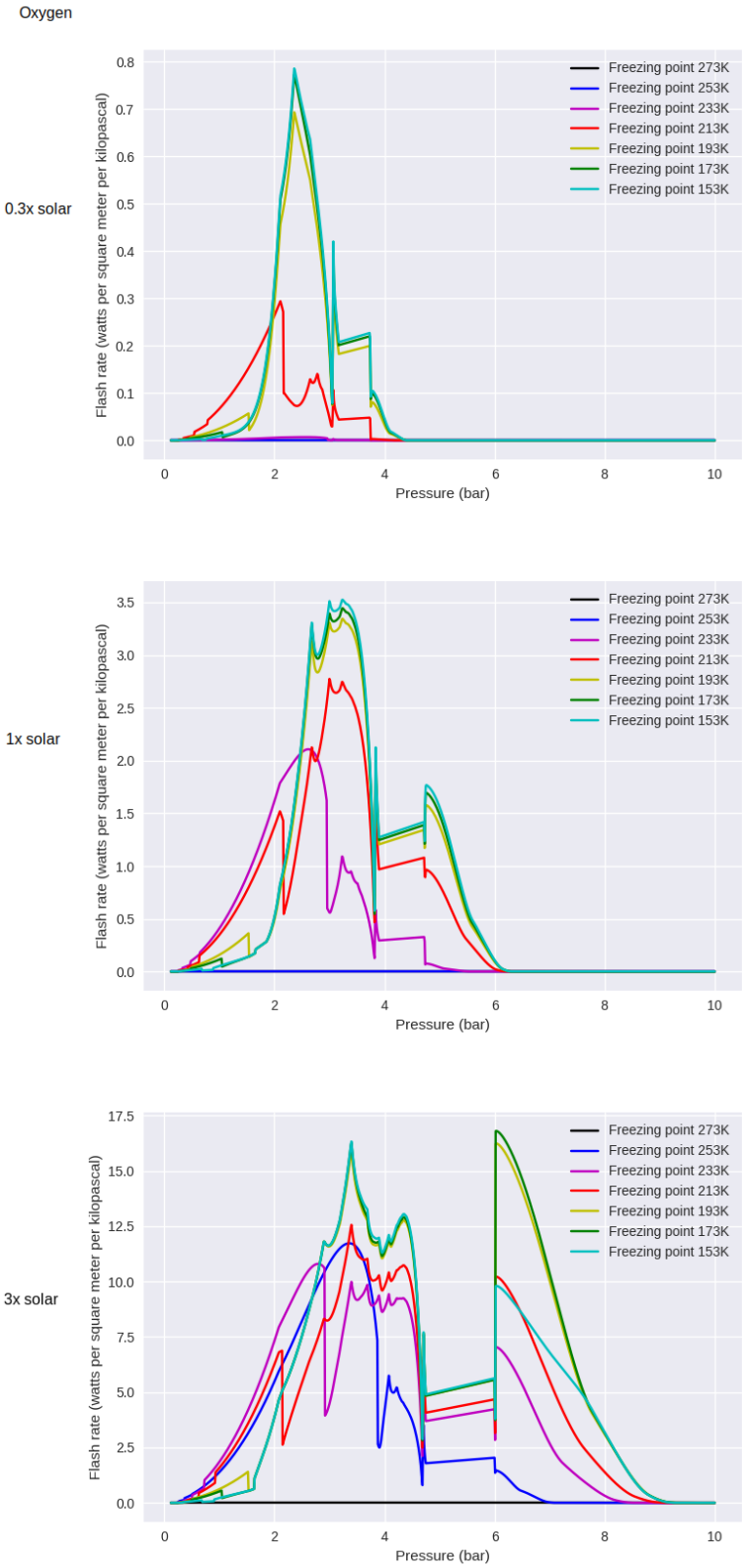


Figure 4.1: Modeled flash rates for Jupiter, for water abundances given in left column.

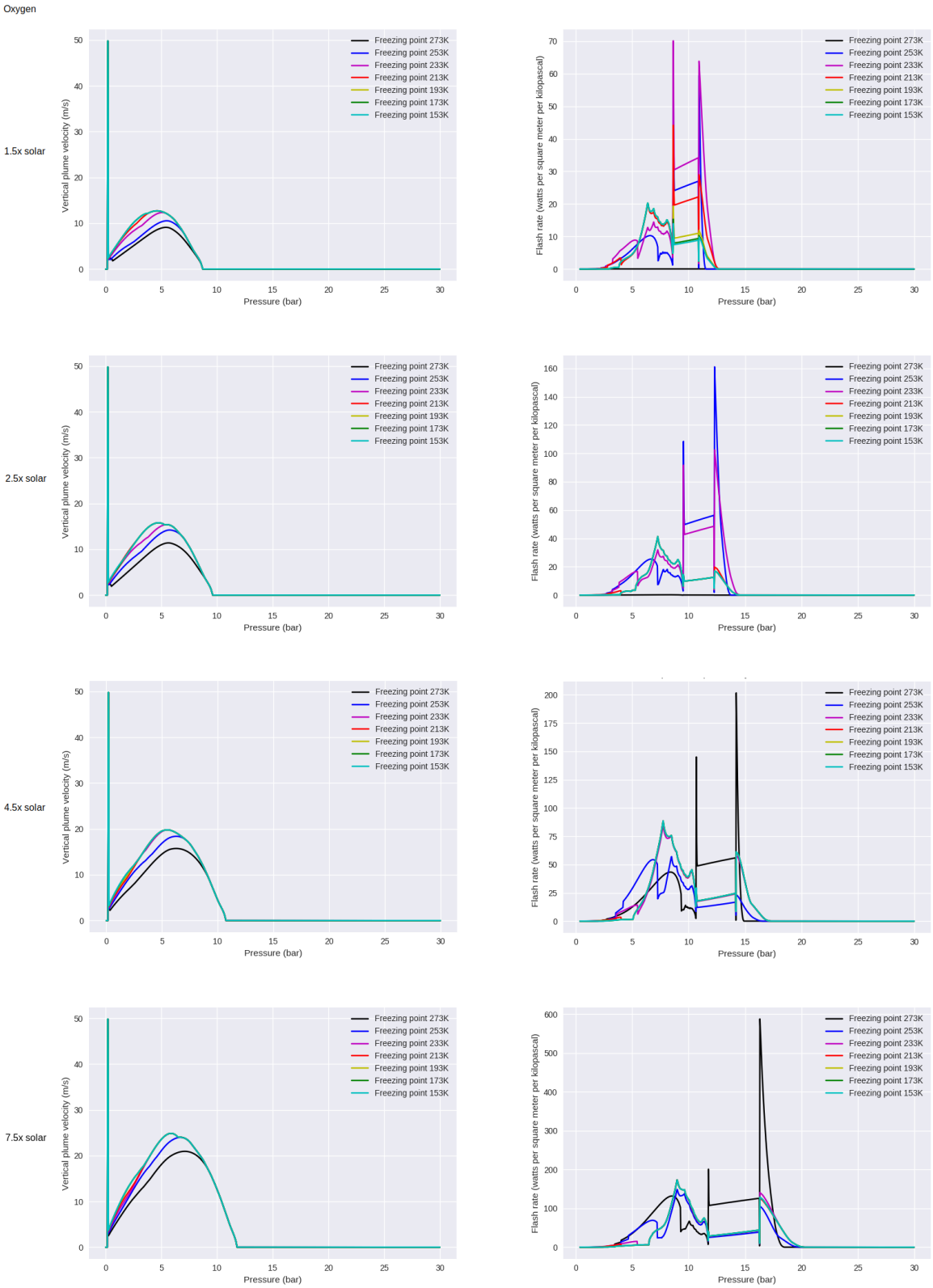


Figure 4.2: Saturn model results for (left) plume velocity and (right) flash rate, for water abundances stated in left column.

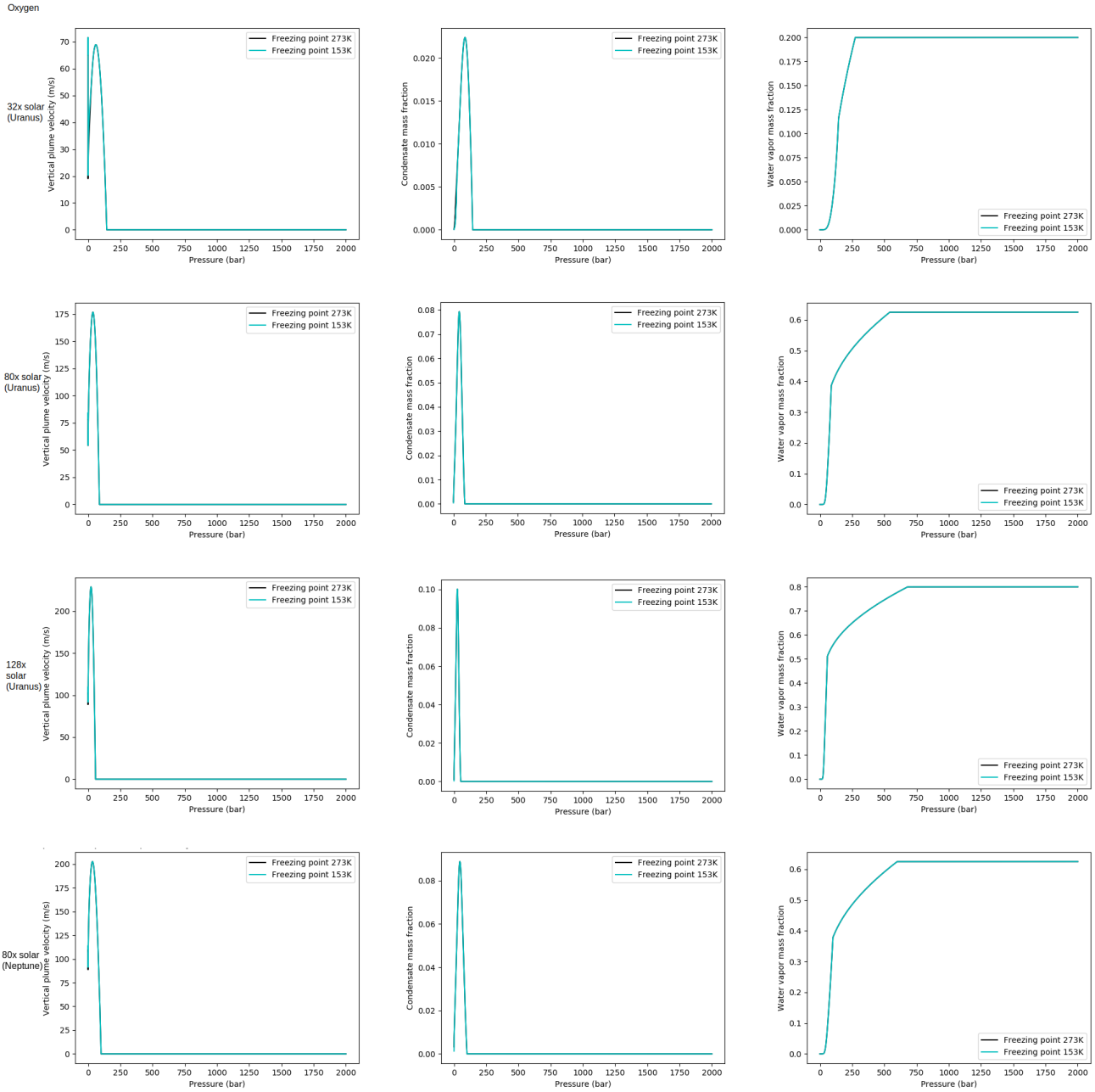


Figure 4.3: Model results for Uranus and Neptune for (left) plume velocity, (center) mass fraction cloud suspended in plume, and (right) vapor mass fraction, for water abundances and planet as stated in left column.

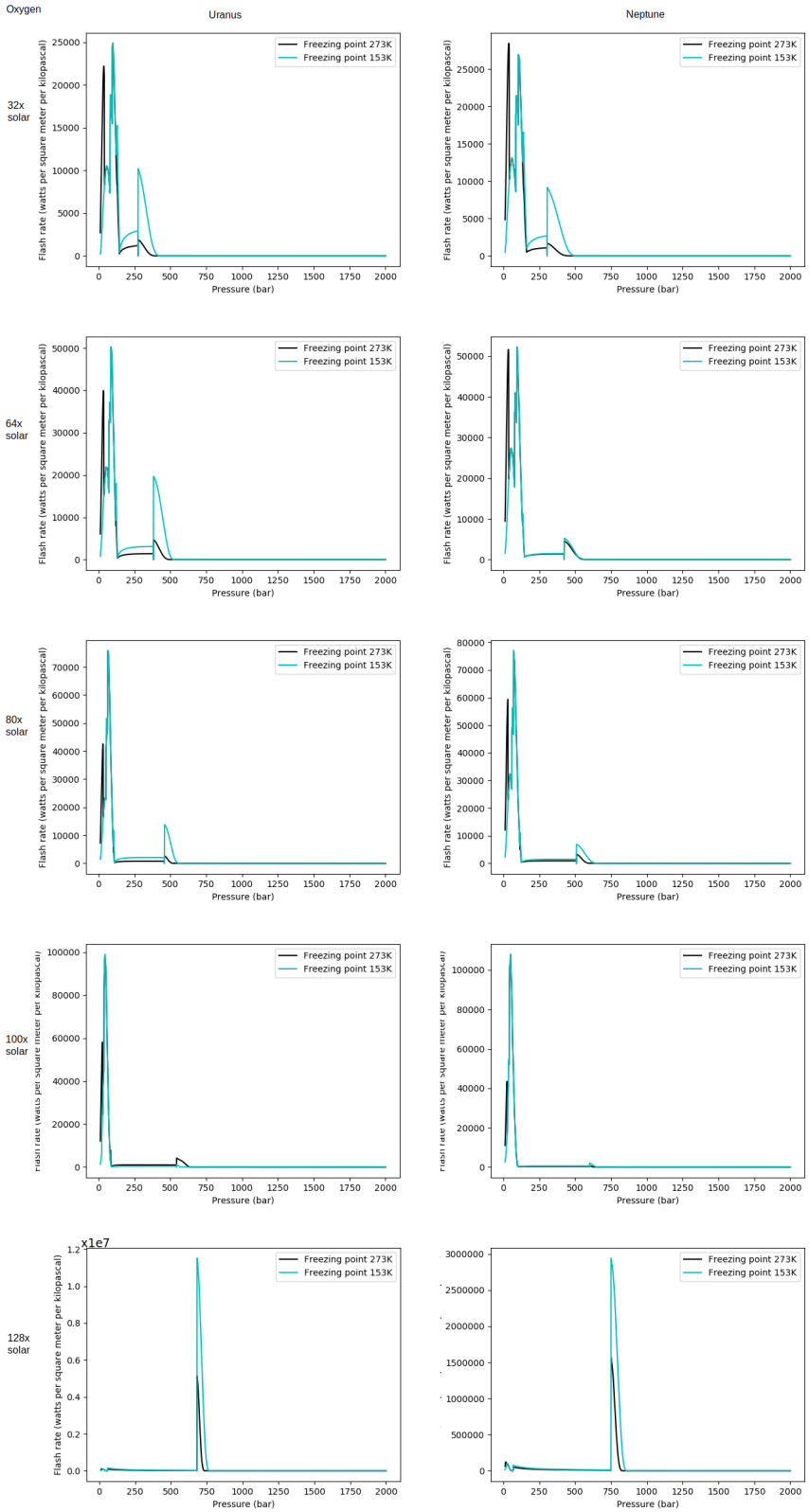


Figure 4.4: Modeled flash rates for (left) Uranus and (right) Neptune, for water abundances stated in left column

CHAPTER 5

EFFECTS OF AMMONIA CLOUDS

5.1 Introduction

The detection of shallow Jovian lightning by Becker *et al*. (2020) was unexpected, and its proximity in altitude to the presumed Jovian ammonia cloud led to questions of whether the lightning might be caused by ammonia. The water cloud is significantly denser, and previous modeling efforts (Yair *et al*. 1995ab, Gibbard *et al*. 1995) have concluded that observed Jovian lightning originates there; as shown in Chapter 3, this is entirely consistent with the shallow detections. Nevertheless, ammonia is a polar molecule with behavior analogous to water in a number of ways, and thus the possibility of electrification in the ammonia cloud is worth testing. Furthermore, on Uranus and Neptune, all heavy elements are believed to be enriched substantially compared to Jupiter and Saturn, with thicker clouds, and Gibbard (1996) concluded that the Neptunian ammonia clouds are favorable for lightning generation. At the same time, if moist convection is ongoing (as in a thunderstorm) the two cloud species cannot be treated as separate systems.

Section 5.2 describes the parameters chosen for GEPE 0.6 to study the effects of the ammonia cloud deck on lightning for Jupiter, Saturn, Uranus and Neptune, along with exploratory work for the effect of ammonium hydrosulfide on Jupiter. Section 5.3 describes the results throughout their atmospheric columns, with an anhydrous ammonia cloud having minimal effect for the gas giants but substantially raising lightning rates for the ice giants. Section 5.4 describes the conclusions and implications for future observations.

5.2 Choice of parameters

A number of adjustable model parameters are kept the same throughout the calculations. As before, based on Earth calibration, it is assumed that that colliding solid particles will always transfer charge, while colliding liquid particles will merge 80% of the time and transfer charge the remaining 20%. The drag coefficient is set to 0.5, and for Uranus and Neptune, where breakdown fields in the deeper part of the convective region are high, the threshold where electrostatic levitation becomes significant is set at a field strength of 107 N C^{-1}. Plume radius is set to 5 kilometers, and a stratospheric temperature profile is set at pressures of 0.25 bars or less. Ammonia liquid and solid charging efficiencies are set to be equivalent to water, and the total magnitude of that charging is as described in section 2.4. Changing these rates would directly change overall flash rate by a proportional factor; since, on all four giant planets, the ammonia cloud is solid and thus does not undergo particle growth in this model, ammonia-caused lightning is triboelectric, and therefore the (substantial) uncertainty in the triboelectric charging rate can be directly translated to an equivalent uncertainty in additional flash rate from ammonia.

The temperature profile is by default retained from Chapters 3 and 4: Jupiter is initialized with a temperature of 330K at 10 bars, Saturn with 380K at 30 bars, Uranus with 960K at 2000 bars, and Neptune with 940K at 2000 bars, so as to match up with the stratospheric temperature profile at 0.25 bars. For Saturn, additional runs were performed with a colder temperature of 360K at 30 bars.

The molar mass of 'dry air' is set for Jupiter and Saturn at 2.2 g mol^{-1}, corresponding to hydrogen with a slightly subsolar helium enrichment, as is observed for both planets (von Zahn *et al*. 1998, Koskinen and Guerlet 2018). For Uranus and Neptune, the atmosphere has a significant enrichment in carbon, and thus the dry molar mass is much higher, but remains poorly constrained. We use the composition of Wiktorowicz and Ingersoll (2006) while excluding water and ammonia, yielding an average molar mass of 5.116 g mol^{-1}, for dry air on both ice giants.

The molar masses of ammonia and water themselves are known. However, as we are considering a water cloud that sequesters ammonia, we count a corresponding fraction of ammonia with the water vapor, lowering the effective water molar mass; this parameter is equivalent to the mass fraction of ammonia in water. We consider cases where the water cloud is 0%, 10%, 20%, or 35% ammonia by mass, with the last of those corresponding to the eutectic. If the N:O ratio in the overall system is too low for a eutectic-composition water cloud, we do not consider the physically impossible compositions, but do consider the case where all nitrogen is sequestered into the water cloud. The freezing point depressions due to ammonia are – for the above weight percentages – approximately 0K, 10K, 30K, and 100K. We lower the effective freezing point an additional 10K to account for supercooling, and linearly interpolate between these data points for a water cloud with an intermediate composition (which can only happen in the case where all ammonia is sequestered). Thus, the water cloud composition parameter also determines water freezing point. As discussed in Aglyamov *et al*. (2021), the presence of ammonia in the water cloud is required to explain lightning observations on Jupiter, and also substantially enhances lightning rates on Saturn.

Water abundances, which obviously have a central influence on lightning rates, were set to the same grid of values as in Aglyamov *et al*. (2023). This includes 0.3x, 1x, 3x, and 10x solar oxygen for Jupiter; 1x, 1.5x, 2.5x, 3x, 4.5x, 7.5x, and 10x solar oxygen for Saturn; and for Uranus and Neptune, expressed as mass-fractions of water (out of water plus dry air), 32x, 60x, 80x, 100x, and 128x, equivalent to mass ratios of water to dry air of 1:4, 3:5, 1:1, 5:3, and 4:1. Water abundance remains observationally uncertain on all of these planets, although for Saturn the extreme values of 1x and 10x are both unlikely and were included only to enable direct comparison with Jupiter.

Nitrogen abundance on Jupiter was taken as 3x solar, as per the Galileo probe measurements. On Saturn the uncertainty for nitrogen is greater than for Jupiter, but based on Fletcher *et al*. (2011), we also used an enrichment of 3x solar, while also testing a value of 6x solar. For Uranus and Neptune, we used the Wiktorowicz and Ingersoll (2006) enrichment of 48x solar, meaning a 1:24 mass ratio of NH_3 to dry air. For the NH_4SH analysis on Jupiter, sulfur abundance was taken to be 3x solar, also in line with Galileo probe observations.

5.3 Results

For Jupiter, substantial lightning is not observed as a result of ammonia clouds. As in previous work, ammonia primarily affects lightning rates by acting as an antifreeze within the water clouds (Figure 5.1e). Overall flash rates instead decrease with the presence of ammonia, if it is not mixed into the aqueous clouds. This is due to the molecular weight effect: ammonia vapor is denser than the hydrogen/helium mixture of giant planetary ‘dry air’, and so its presence adds negative buoyancy to the plume, reducing upwelling velocity (Figure 5.1a and 5.1d) and the quantity and size of water

cloud particles produced and lofted. Additionally, as seen in Figure 5.1b and 5.1c, the ammonia vapor concentration begins decreasing below the ammonia cloudbase due to entrainment mixing the upwelling plume with surrounding dry air; as a result, the ammonia enrichment in the plume is decreased, leading to even fewer ammonia particles condensing. In sum, the ammonia cloud is simply too thin to meaningfully change the lightning depth distribution on Jupiter, and the flash rates.

The results for Saturn are mostly qualitatively similar (Table 5.1). The presence of ammonia not dissolved in the water cloud acts to slightly suppress lightning rates for oxygen abundance up to 2.5x solar. At 4.5x solar oxygen, however, flash rates are similar for the case with and without nitrogen, and for 7.5x solar oxygen, flash rates are higher with ammonia present, although still higher for 3x solar nitrogen than 6x solar nitrogen. While ammonia vapor still makes the plume less buoyant, leading to decreased upwelling velocity, under these conditions this effect is smaller relative to water abundance, and can be outbalanced by triboelectric charging accelerating electrification due to ammonia precipitation colliding with water cloud particles. A similar effect is observed on Jupiter for 10x solar oxygen abundance, although such a strongly elevated water abundance is more likely on Saturn than on Jupiter. Like on Jupiter, minimal lightning is observed in the region of the ammonia cloud; in addition to the cloud being thin, ammonia is solid at the temperatures involved, restricting it from particle growth. The colder temperature profile with a base temperature of 360K, meanwhile, sets both the ammonia and water cloudbases lower, and increases lightning rates by a factor of between 1.5 and 2, regardless of whether ammonia is involved (Figure 5.2).

For Uranus and Neptune, the situation is quite different (Tables 5.2 and 5.3). The temperature profile of the atmosphere is far cooler than Jupiter and Saturn, and water abundance far higher, leading to deep clouds and the potential for extremely strong convection and extremely high flash rates. The presence of ammonia increases overall flash rates by a factor of 3-4 at oxygen abundances of 80x solar (50% water by mass) or below; this increase is seen both near the ammonia cloud at 0-10 bars and at deep pressure levels >200 bar, but is small in the region of active convection. Neptunian flash rates are higher than Uranian rates at equal water abundance, due to its colder temperature profile.

Lightning is therefore predicted at the ammonia cloud level, but is caused by an interaction of lofted water ice particles and the ammonia cloud itself. Due to the low temperatures, evaporation is slow enough that ammonia grains can fall deep into the water cloud region.

Ammonia does not increase flash rates for a 128x solar (80% by mass) water abundance. On Uranus and Neptune, due to the extremely high water abundance, there is a huge amount of potential energy available for convection, but the molecular weight effect is also extremely effective in preventing it. In the model, with a lifted moist parcel, the result is very vigorous convection far above the cloud base, the gap increasing with water abundance (Figure 5.3). This is especially true for a >50% water atmosphere, and leads to the additional molecular weight effect of ammonia being especially effective in inhibiting convection.

Unlike ammonia, ammonium hydrosulfide is found to be capable of producing substantial flash rates on Jupiter in its clouds. In particular, for solar or subsolar water

abundances, it can produce substantial lightning due to the triboelectric effect even in the absence of liquid water and, thus, without ammonia being dissolved in the water cloud (Figure 5.4). The quantity of lightning is, however, smaller than that in the water cloud, and for 3x or 10x solar water, the hydrosulfide cloud has negligible effect except in cases where it's the only source of lightning.

5.4 Discussion

The modeled results depend directly on the amount of triboelectric charging, which is estimated through a rather long chain of extrapolations for ammonia and an even longer one for ammonium hydrosulfide. If ammonia-water collisions have an order of magnitude less charge transfer, anhydrous ammonia clouds do not play a substantial role in electrification in any of the Solar System giant planets; if they have an order of magnitude more, they become the primary lightning mechanism on Uranus and Neptune. Furthermore, the model assumes cloud particles of two set compositions, but this is unlikely to be a true depiction of ammonia and water clouds on giant planets. Indeed, it's possible that some ammonia-water collisions are the means by which ammonia is sequestered in water clouds in the first place, as thermodynamically the expectation is for Jovian water clouds to mostly condense as pure water (Guillot *et al*. 2020). Similar considerations apply to the interaction of ammonium hydrosulfide with ammonia. These dynamics would, however, require a time-variable model to represent in full, as they are necessarily tied to hail particles that are repeatedly lifted and dropped in a thunderstorm, a non-equilibrium process.

For Jupiter and Saturn, the presence of a separate ammonia cloud has minimal effect on lightning rates, either at cloud level or lower due to precipitation. While an

ammonia cloud does form, its mass is simply insufficient to meaningfully affect electrification. The negative result confirms the Chapter 3 conclusion that SRU observations of Jovian shallow lightning (Becker *et al*. 2020) imply ammonia admixture in the water clouds, rather than lightning locally generated in the ammonia clouds. By contrast, the ammonium hydrosulfide cloud, being located at deeper levels and thus containing a higher overall density of material, may provide an alternative method for the generation of Jovian lightning without ammonia admixture. If oxygen abundance is sufficiently high to generate water-driven lightning, though, the effects of ammonium hydrosulfide are comparatively minor. However, the electrical properties of ammonium hydrosulfide remain extremely poorly known, and further constraints are required to make quantitative conclusions.

For Uranus and Neptune, the calculated flash rates in the upper atmosphere are still sufficiently high to intersect oddly with Voyager 2 observations of lightning sferics significantly weaker than those on Saturn (Zarka and Pedersen 1986, Zarka *et al*. 2004). If the sources of these signals are shallow, rather than deep and attenuated, they must be small in horizontal scale, which could in principle be related to the distinct triboelectric charging mechanism. The overall high flash rates, however, are most likely explained by intermittent convection – Saturn's flashes are as energetic as Jupiter's, but rarer, and average electrical power output is capped by overall planetary heat flow. The molecular weight effect, while relevant, is probably not as dominant as in the model because moist and dry regions are not perfectly separate throughout the cloud column; a parcel that's negatively buoyant will only be mechanically lifted so far, in practice. It's thus likely that, if water mass fraction does exceed 50%, multiple convective layers can form throughout the water cloud deck.

Nevertheless, two main conclusions are unaffected by this. Firstly, triboelectric charging from condensates of different composition can provide a large fraction of lightning on the ice giants, including a majority of shallow lightning. This may be increased further by the presence of NH_4SH and/or H_2S clouds, neither of which is modeled here; in particular, H_2S has low polarity, and while Gibbard (1996) found that this makes it a poor candidate for self-charging, the situation is much more favorable for triboelectric charging with more polar condensates.

Secondly, we emphasize that convection bringing up cloud material that has condensed at greater depths plays a crucial role in determining lightning rates, as well as the state of the atmosphere in general. The condensation of water and ammonia is predicted to happen at practically non-overlapping altitudes (Atreya & Wong 2005), but nevertheless convective mixing can bring substantial water cloud mass up to these shallow levels, such that the shallow cloud may contain more water than ammonia.

Table 5.1: *Flash rates for Saturn (W m^{-2}) depending on oxygen abundance, nitrogen abundance, and mass fraction ammonia in the water clouds*

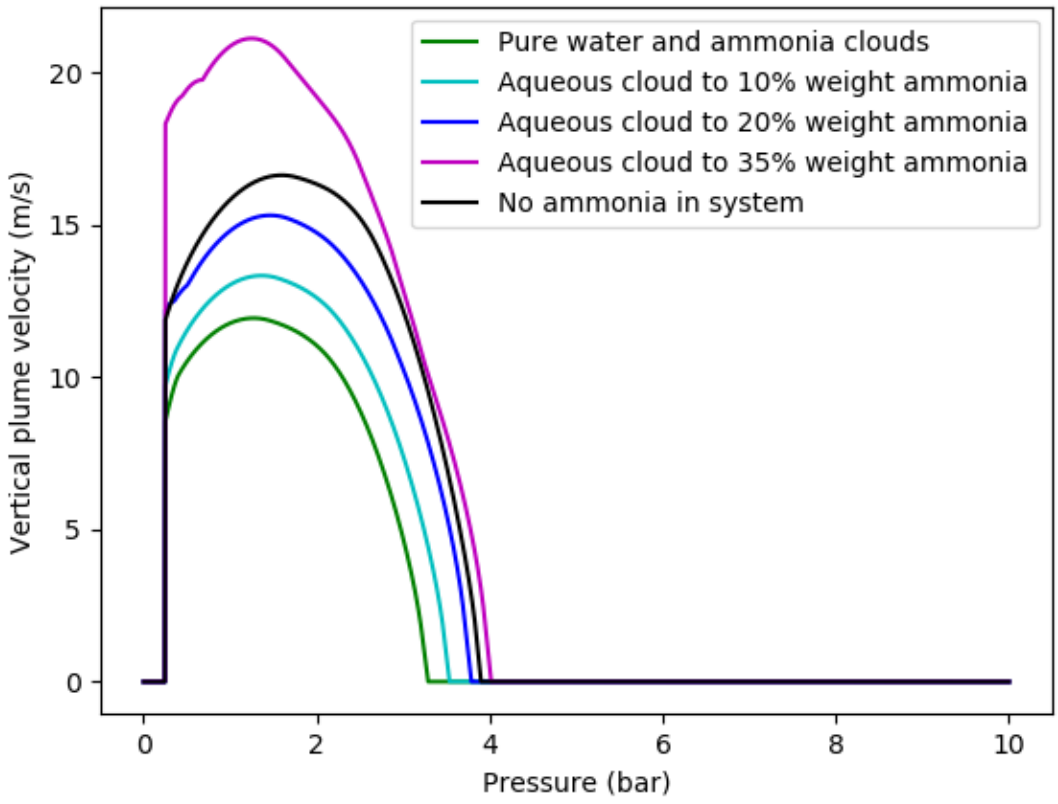
Pressure (bar)	O 1.5x, N 0x	O 2.5x, N 0x	O 4.5x, N 0x	O 7.5x, N 0x	O 1.5x, N 3x, 0%	O 1.5x, N 3x, 10%	O 1.5x, N 3x, 20%	O 2.5x, N 3x, 0%	O 2.5x, N 3x, 10%	O 4.5x, N 3x, 0%	O 7.5x, N 3x, 0%	O 1.5x, N 6x, 0%	O 1.5x, N 6x, 10%	O 1.5x, N 6x, 20%	O 7.5x, N 6x, 0%
0-3	1	6	6	9	0	4	5	4	7	12	11	0	0	5	12
3-6	8	138	426	774	0	63	144	67	235	352	812	0	0	79	813
6-9	16	301	961	2128	0	78	342	131	380	811	2109	0	0	147	1975
9-12	0	94	506	1834	0	24	337	13	264	525	2089	0	0	111	1725
12-15	0	9	264	705	0	0	1	1	87	423	1878	0	0	0	1475
15-18	0	0	0	846	0	0	0	0	0	0	1865	0	0	0	1426
18-21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.2: *Flash rates for Uranus (W m^{-2}) depending on oxygen abundance and nitrogen abundance, without ammonia in the water clouds*

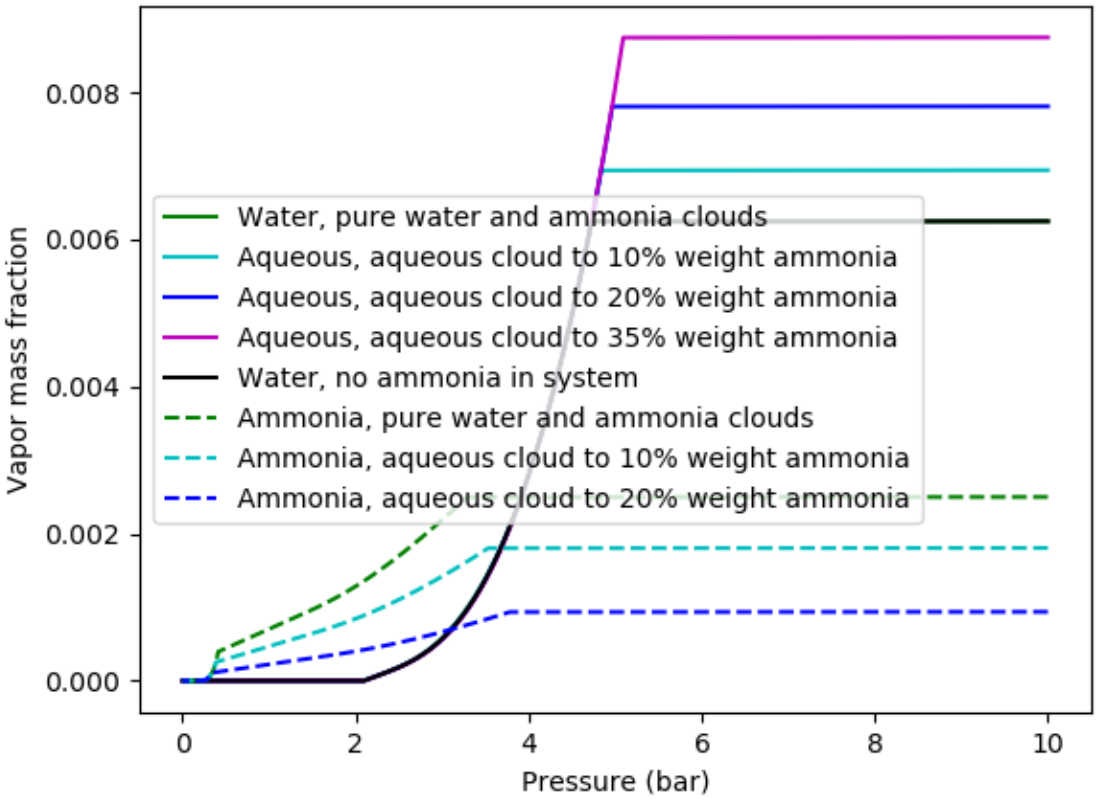
Pressure (bar)	O 32x, N 0x	O 32x, N 48x	O 60x, N 0x	O 60x, N 48x	O 80x, N 0x	O 80x, N 48x	O 100x, N 0x	O 100x, N 48x	O 128x, N 0x	O 128x, N 48x
0-5	62	199	128	257	178	738	995	829	4830	7006
5-10	742	676	1117	1547	3661	5380	18 954	22 575	38 489	44 731
10-15	2374	2679	6039	8001	21 932	31 668	64 681	71 500	121 160	136 425
15-20	6403	9290	23 277	30 137	74 716	94 141	139 312	154 355	176 741	165 118
20-30	40 366	47 100	112 951	204 756	384 570	431 696	437 956	433 443	419 099	404 870
30-40	49 377	64 698	157 798	221 669	312 383	314 179	479 499	535 103	190 368	100 956
40-1000	15 058 643	62 537 580	25 156 949	93 637 819	6 109 149	20 033 094	1 977 590	3 792 983	95 348	103 755

Table 5.3: *Flash rates for Neptune (W m-2) depending on oxygen abundance and nitrogen abundance, without ammonia in the water clouds*

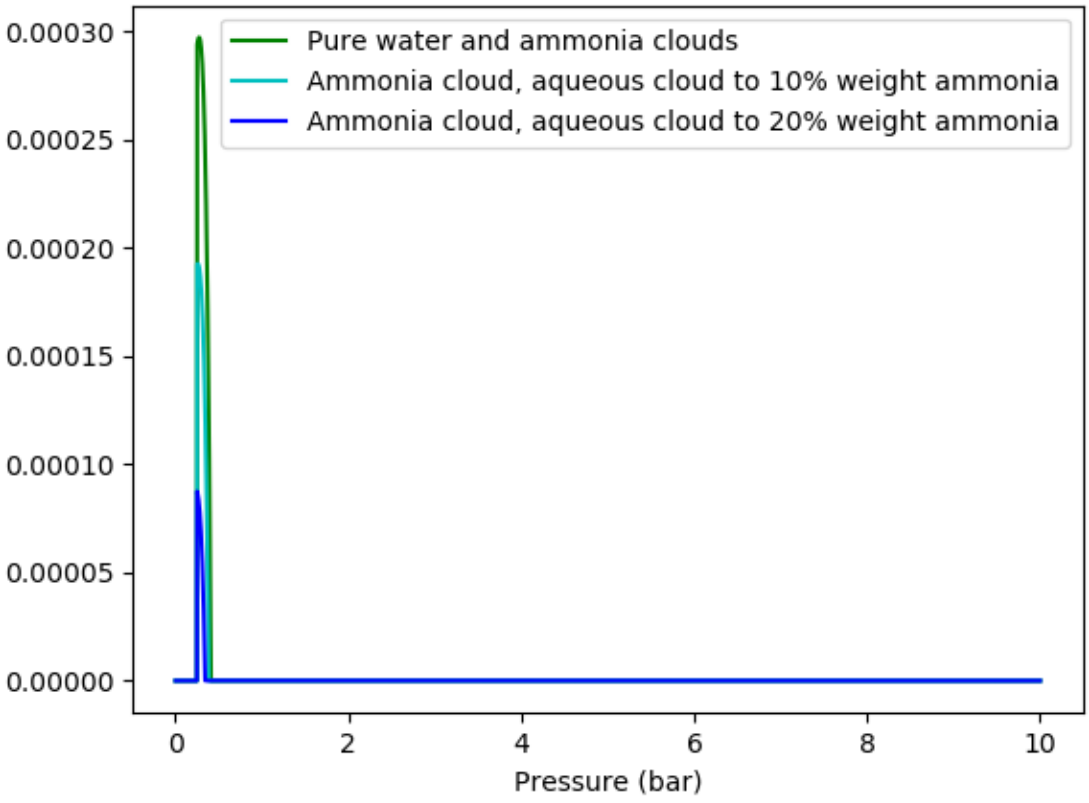
Pressure (bar)	O 32x, N 0x	O 32x, N 48x	O 60x, N 0x	O 60x, N 48x	O 80x, N 0x	O 80x, N 48x	O 100x, N 0x	O 100x, N 48x	O 128x, N 0x	O 128x, N 48x
0-5	168	302	265	953	553	1573	2816	1281	7705	12 343
5-10	1207	2290	2843	5553	9539	13 296	32 409	35 328	53 982	62 410
10-15	4565	5791	14 342	18 721	44 118	58 913	88 737	99 601	152 432	165 451
15-20	13 821	19 089	46 349	60 272	118 541	132 993	176 990	198 999	206 103	185 198
20-30	67 214	81 968	199 642	234 752	473 208	526 237	516 276	518 936	487 011	476 804
30-40	95 863	109 084	222 218	247 054	388 583	380 027	526 426	613 425	364 074	261 252
40-1000	31 803 586	98 764 804	45 068 464	154 107 177	12 726 057	29 769 784	3 050 263	6 659 619	258 803	238 994


Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
Aqueous cloud to 20% weight ammonia
Aqueous cloud to 35% weight ammonia
No ammonia in system
Vertical plume velocity (m/s)
Pressure (bar)
0
5
10
15
20
0
2
4
6
8
10

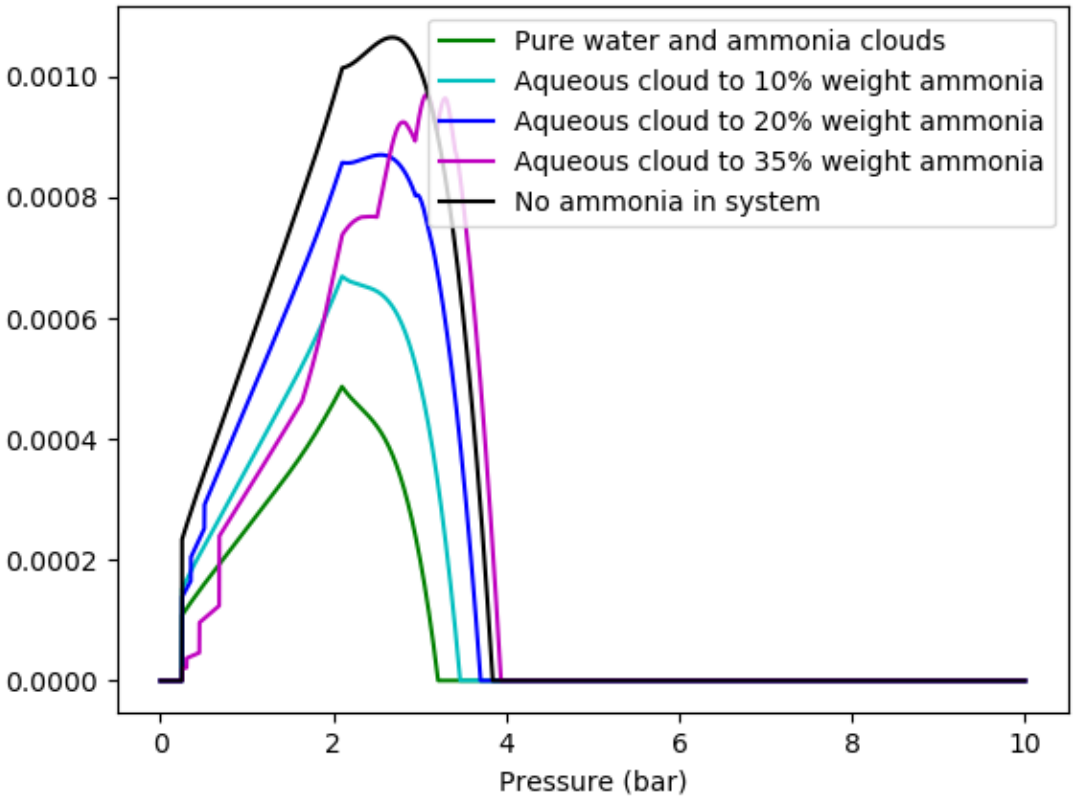
(a)


0.008
0.006
0.004
0.002
0.000
Vapor mass fraction
0
2
4
6
8
10
Pressure (bar)
Water, pure water and ammonia clouds
Aqueous, aqueous cloud to 10% weight ammonia
Aqueous, aqueous cloud to 20% weight ammonia
Aqueous, aqueous cloud to 35% weight ammonia
Water, no ammonia in system
Ammonia, pure water and ammonia clouds
Ammonia, aqueous cloud to 10% weight ammonia
Ammonia, aqueous cloud to 20% weight ammonia

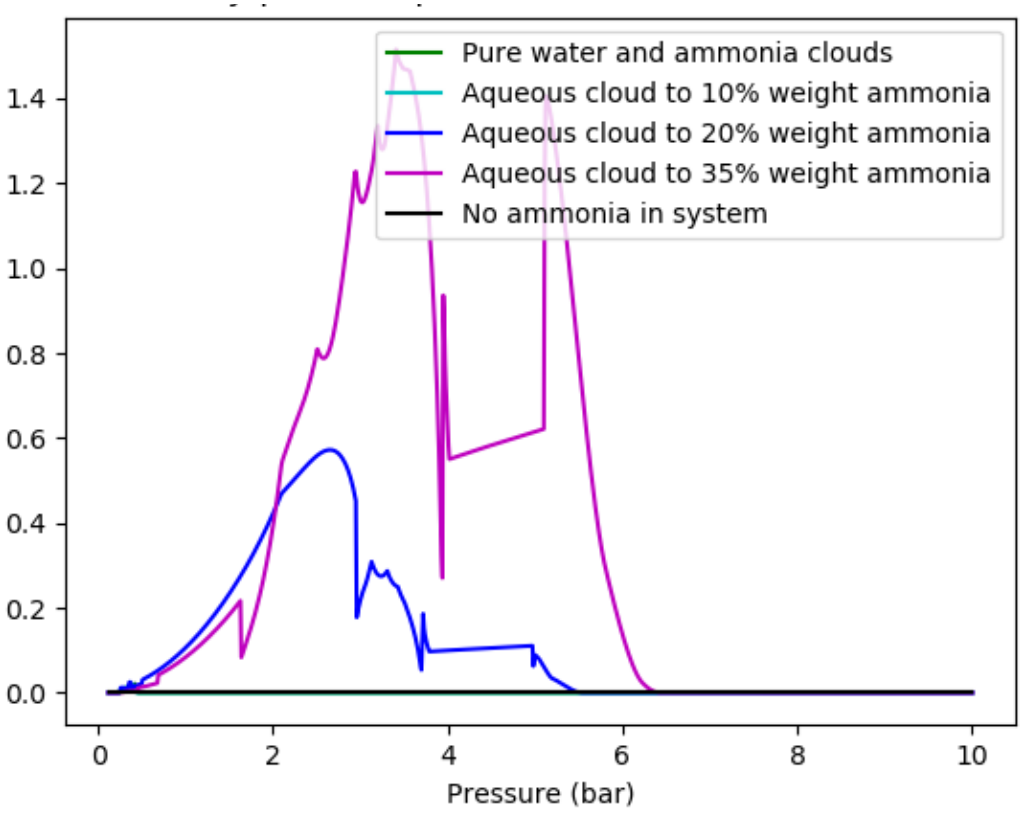
(b)


Pure water and ammonia clouds
Ammonia cloud, aqueous cloud to 10% weight ammonia
Ammonia cloud, aqueous cloud to 20% weight ammonia
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
2
4
6
8
10
Pressure (bar)

(c)

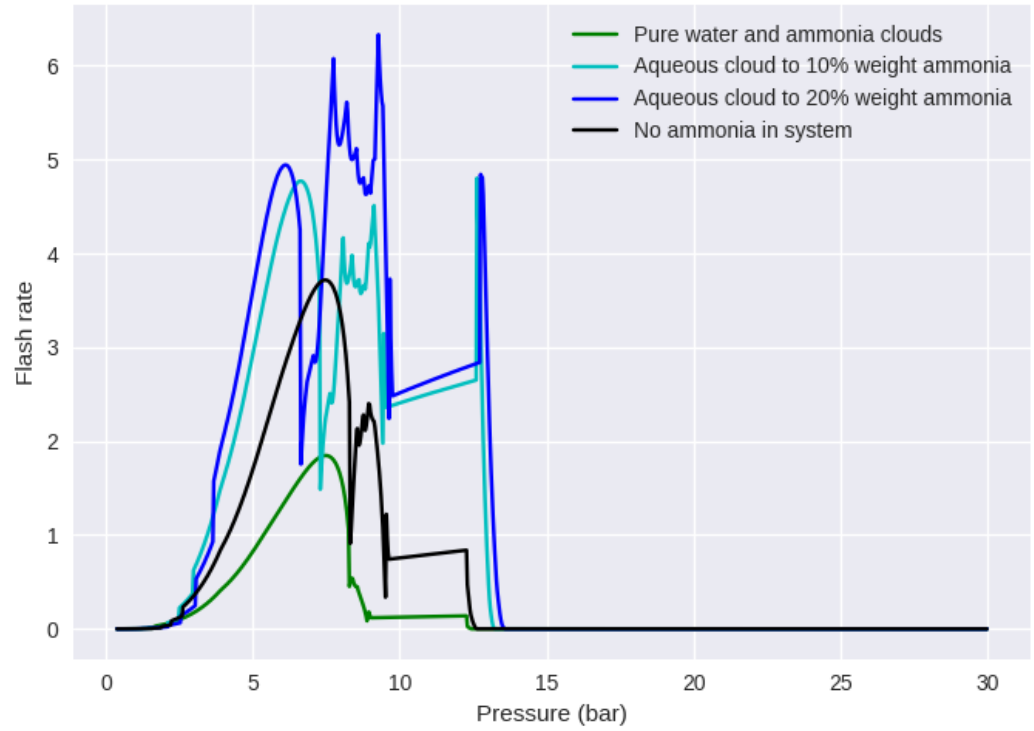

Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
Aqueous cloud to 20% weight ammonia
Aqueous cloud to 35% weight ammonia
No ammonia in system
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
2
4
6
8
10
Pressure (bar)

(d)

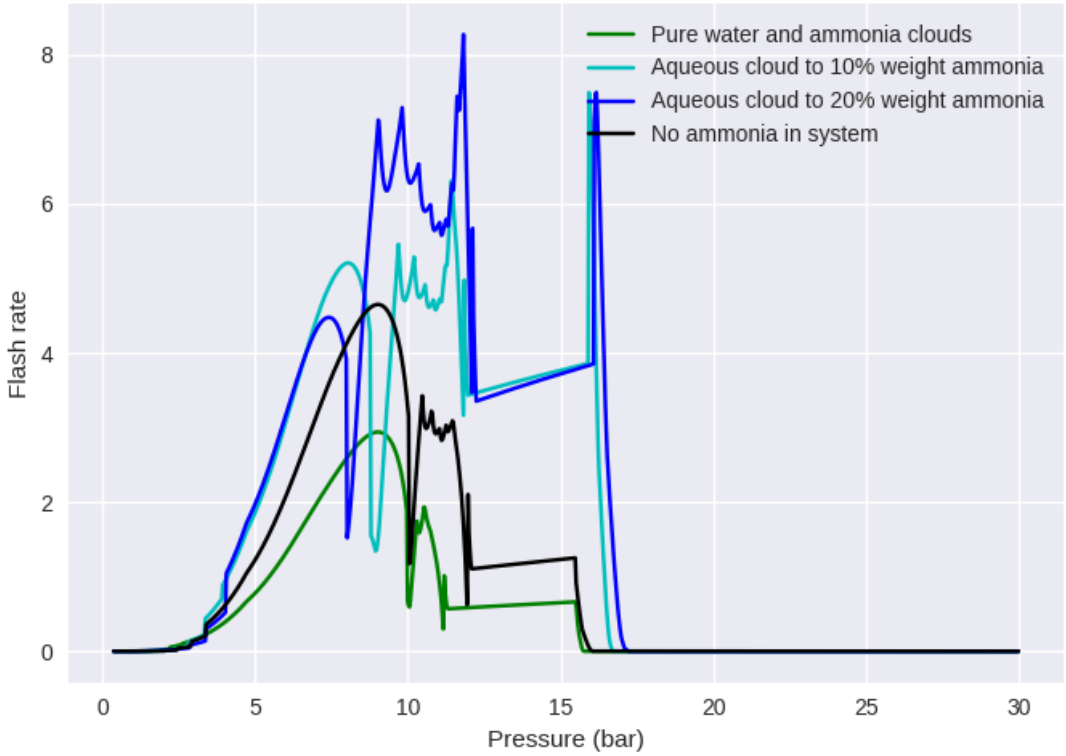


(e)

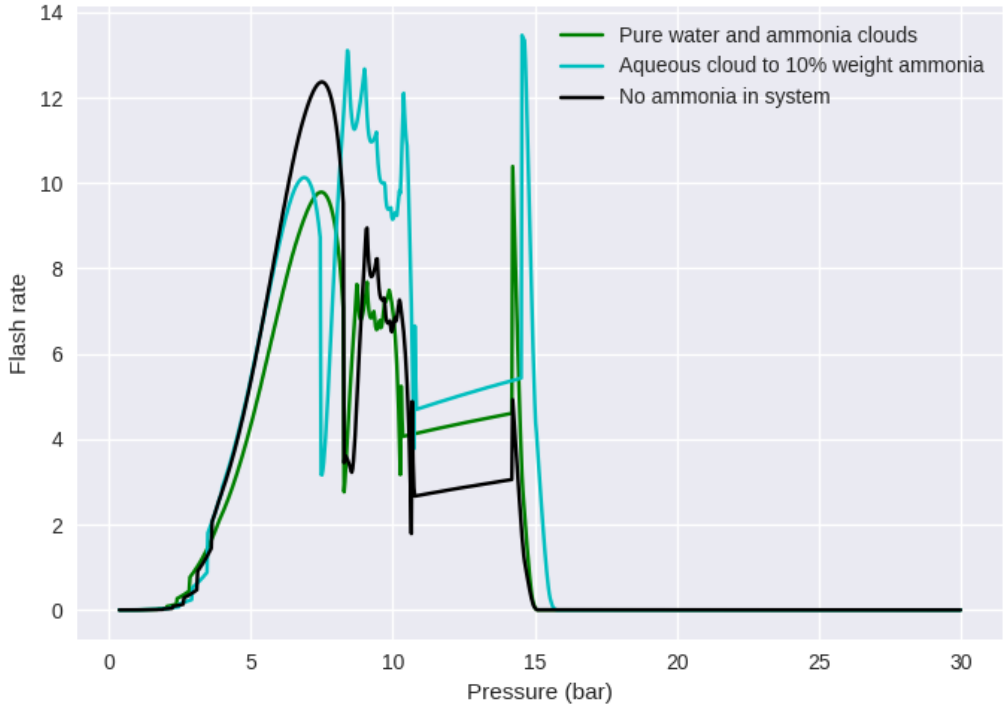
Figure 5.1: Model profiles of Jupiter, with 1x solar oxygen and 3x solar nitrogen abundance: (a) plume velocity, (b) vapor mass fraction, (c) ammonia cloud mass fraction, (d) water cloud mass fraction, and (e) flash rate, as a function of pressure, for varying ammonia admixtures into the water cloud.


Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
Aqueous cloud to 20% weight ammonia
No ammonia in system
Flash rate
Pressure (bar)
0
1
2
3
4
5
6
0
5
10
15
20
25
30

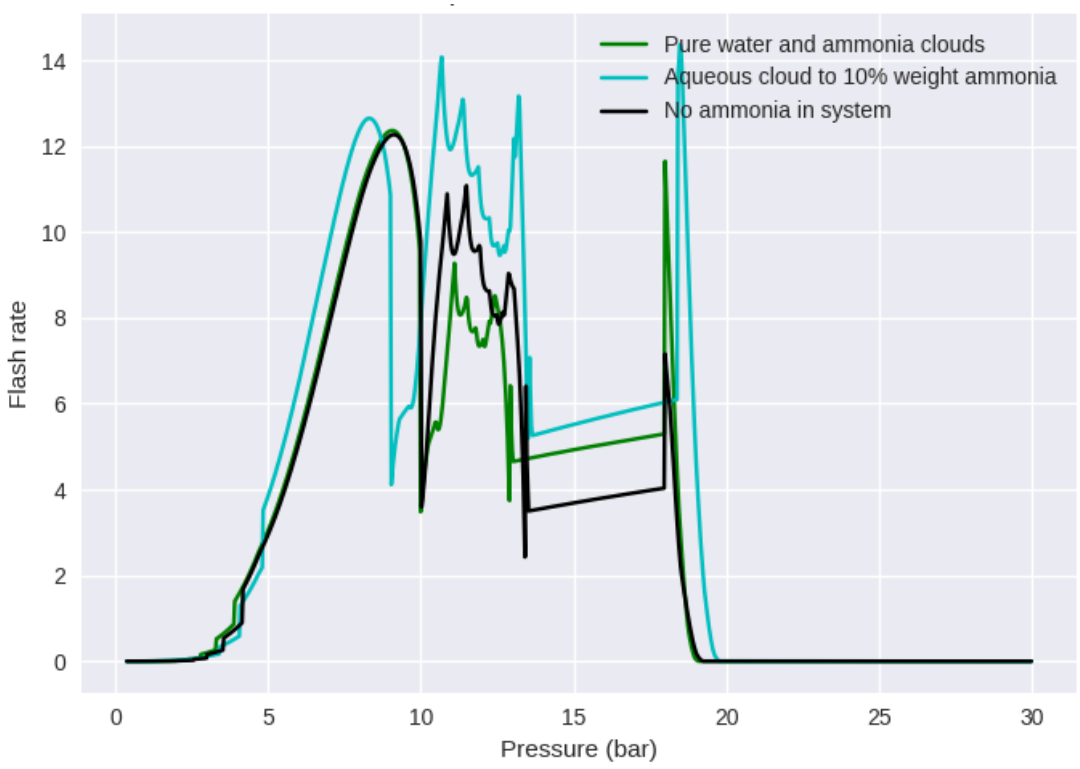
(a)


Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
Aqueous cloud to 20% weight ammonia
No ammonia in system
Flash rate
Pressure (bar)
0
2
4
6
8
0
5
10
15
20
25
30

(b)

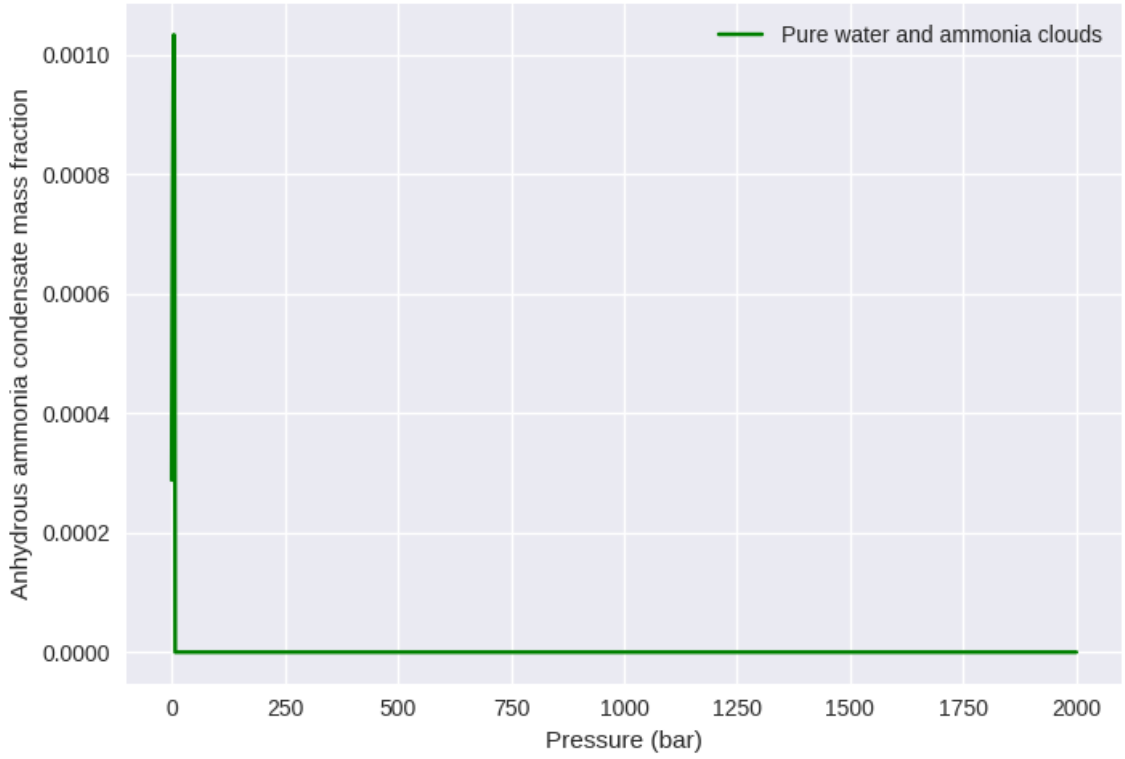

Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
No ammonia in system
Flash rate
Pressure (bar)
0
2
4
6
8
10
12
14
0
5
10
15
20
25
30

(c)

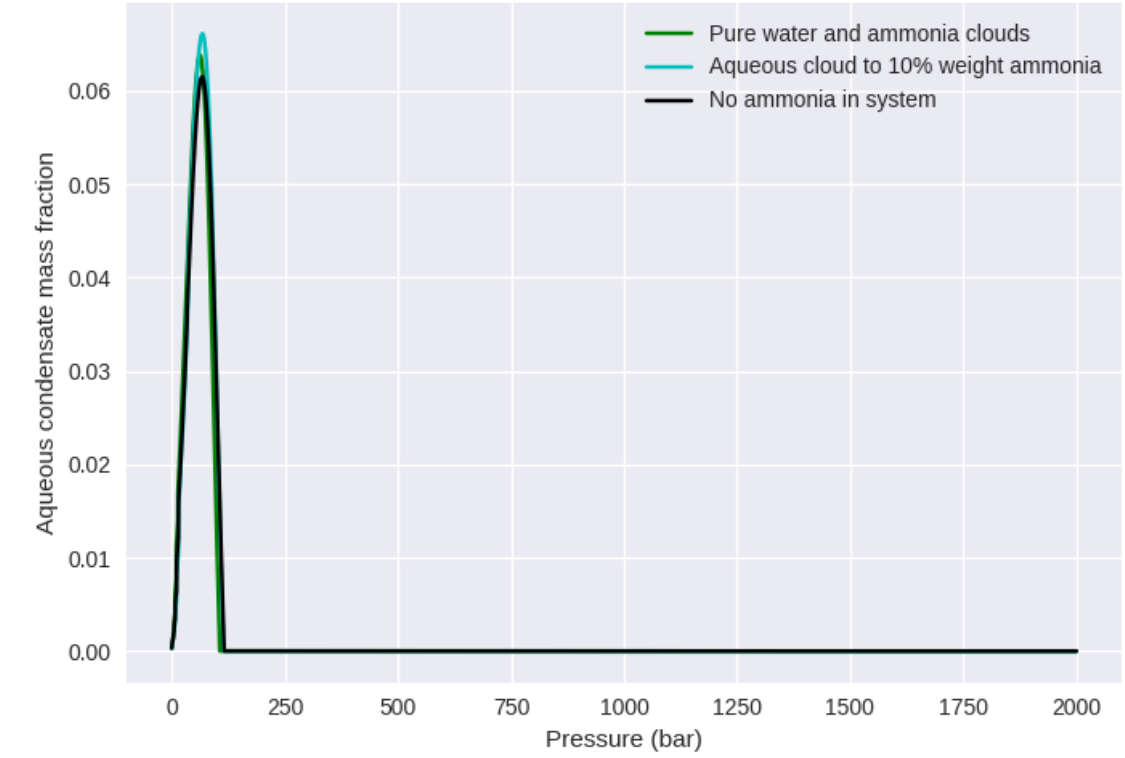


(d)

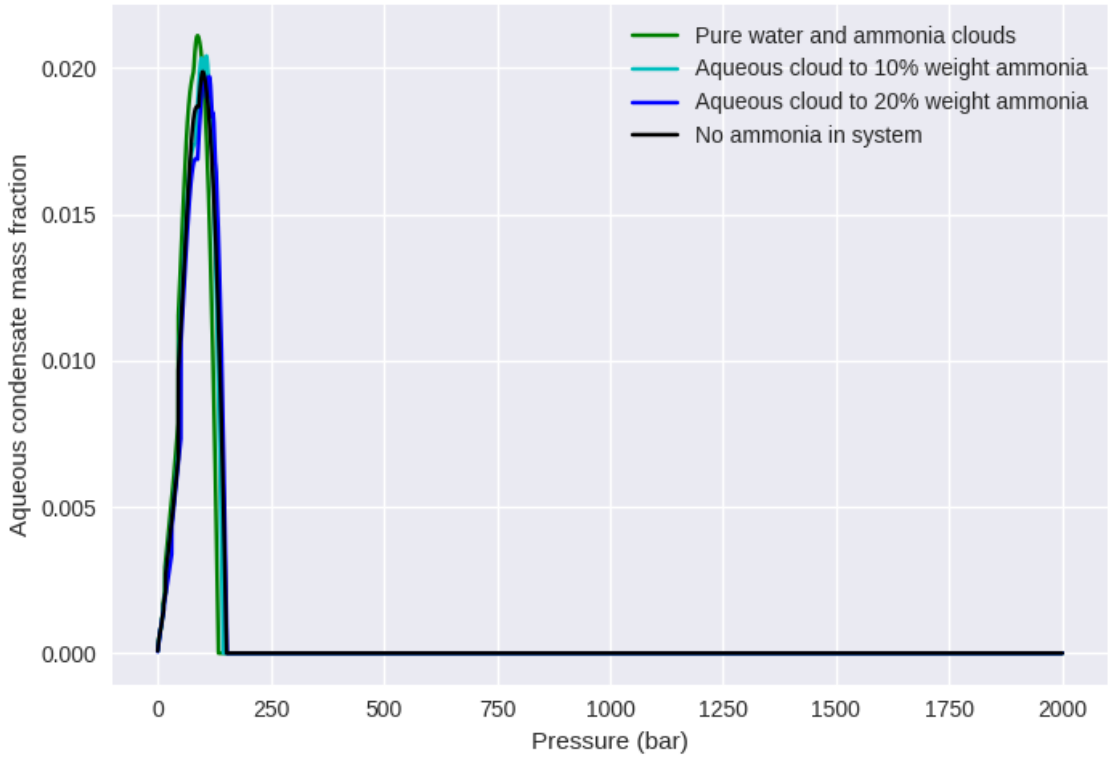
Figure 5.2: Saturn flash rate profiles, for 3x solar nitrogen abundance: (a) 2.5x solar oxygen with a 30-bar temperature of 380K, (b) 2.5x solar oxygen with a 30-bar temperature of 360K, (c) 4.5x solar oxygen with a 30-bar temperature of 380K, (d) 4.5x solar oxygen with a 30-bar temperature of 360K.


Pure water and ammonia clouds
Anhydrous ammonia condensate mass fraction
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
0
250
500
750
1000
1250
1500
1750
2000
Pressure (bar)

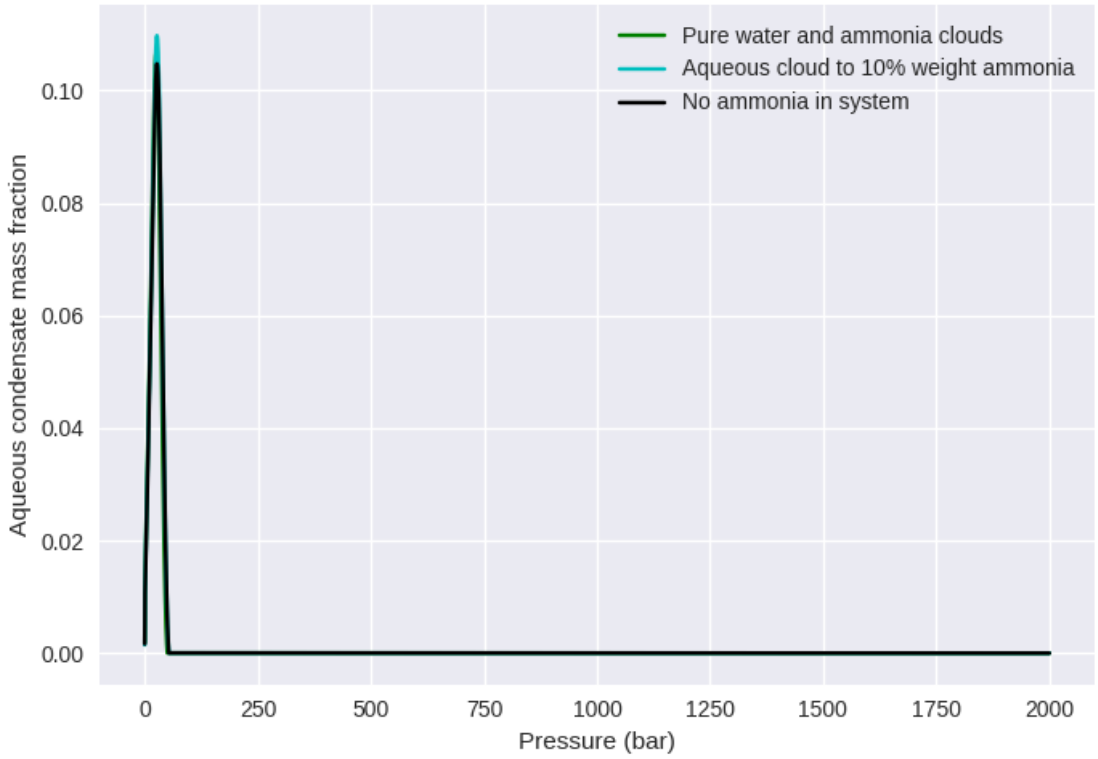
(a)


Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
No ammonia in system
Aqueous condensate mass fraction
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0
250
500
750
1000
1250
1500
1750
2000
Pressure (bar)

(b)


Pure water and ammonia clouds
Aqueous cloud to 10% weight ammonia
Aqueous cloud to 20% weight ammonia
No ammonia in system
Aqueous condensate mass fraction
0.020
0.015
0.010
0.005
0.000
0
250
500
750
1000
1250
1500
1750
2000
Pressure (bar)

(c)



(d)

Figure 5.3: Neptune profiles, for 48x solar nitrogen abundance: (a) ammonia cloud for 80x solar oxygen, (b) water cloud for 80x solar oxygen, (c) water cloud for 32x solar oxygen, (d) water cloud for 128x solar oxygen.

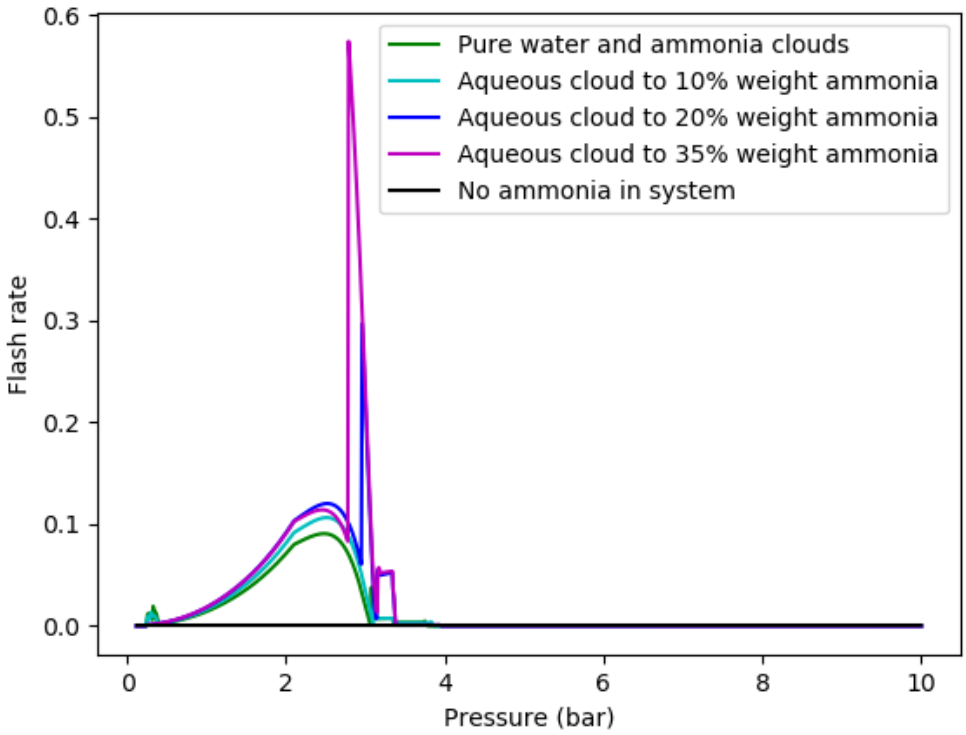


Figure 5.4: Jupiter flash rate profile with an ammonium hydrosulfide cloud as well as water and ammonia clouds, for 0.3x solar oxygen abundance, 1.2x solar nitrogen abundance, and 3x solar sulfur abundance.

CHAPTER 6

CONCLUSIONS

GEPE, a one-dimensional model of moist convection and lightning generation, replicates terrestrial lightning flash rates and altitudes with a reasonable choice of parameters. These, in turn, allow using the model to fit the lightning rates and altitudes observed by Galileo, Juno, and Cassini on Jupiter and Saturn. Observed lightning rates and the presence of relatively abundant shallow lightning (1-2 bars) require a significant amount of dissolved ammonia in the aqueous cloud acting as antifreeze for the water droplets. With a sufficient amount of such dissolved ammonia, subsolar water abundances can be consistent with the observation of lightning. However, subsolar or near-solar water abundances, and the shallow water cloudbase they imply, are not consistent with the deepest Galileo lightning flash observation at 8 bars; the detection of further such flashes would imply a supersolar oxygen abundance on Jupiter, and similar calculations at deeper pressures can be done on Saturn. A wide range of lightning altitudes is to be expected from the aqueous cloud, and the anhydrous ammonia cloud should not be expected to meaningfully contribute to electrification on Jupiter or Saturn.

Uranus and Neptune have the potential for extremely intense, but rare, thunderstorms including their deep water clouds, and the ice giant lightning observed by Voyager 2 could be either deep and attenuated, or shallow and horizontally restricted, in a way a future Uranus Orbiter and Probe has the potential to resolve. Triboelectric charging due to collisions of distinct cloud particle species may be the primary contributor to electrification in ice giant atmospheres, although further work is required to quantify the strength of this effect. The effects, including lightning, of the deep ice giant water

(or water-ammonia solution) cloud can stretch, due to convection and precipitation, a long way both above and below the cloudbase.

APPENDIX

Code for GEPE 0.6, written in Python and with the example set to include ammonia, water, and ammonium hydrosulfide clouds for Jupiter, follows.

```
#1-D lightning simulation
#6E: Inclusion of NH3 and NH4SH


import os, sys

# For scientific computing
import numpy as np

# For plotting
import matplotlib.pyplot as plt
import seaborn as sns

import glob

#Equation of state as function of pressure and molar volume, and its partial derivatives
def T_eos(P,Vm,Rbar=83.1446):
   Pbar = P*(10**-5)
   Vmbar = Vm*(10**6)
   abar = np.exp(19.599 - (Pbar**2.6013)*np.exp(-18.608)) * (Pbar**-0.8946)
   if Pbar < 100:
      bbar = 16.74519884
   else:
      vu = np.log(Pbar)
      bbar = 20.285 - 7.44171*vu + 7.318565*(vu**2) - 3.463717*(vu**3) + 0.87372903*(vu**4)
- 0.12385414*(vu**5) + 0.0098570583*(vu**6) - 0.00041153723*(vu**7) +
0.00000702499*(vu**8)
   T = (Pbar + abar/(Vmbar**2.846))*(Vmbar-bbar)/Rbar
   return T
def dTdP_eos(P,Vm,R=8.31446,delP=0.01):
   Pbar = P*(10**-5)
   Vmbar = Vm*(10**6)
   Rbar = R*10
   abar = np.exp(19.599 - (Pbar**2.6013)*np.exp(-18.608)) * (Pbar**-0.8946)
   if Pbar < 100:
      bbar = 16.74519884
   else:
      vu = np.log(Pbar)
      bbar = 20.285 - 7.44171*vu + 7.318565*(vu**2) - 3.463717*(vu**3) + 0.87372903*(vu**4)
- 0.12385414*(vu**5) + 0.0098570583*(vu**6) - 0.00041153723*(vu**7) +
0.00000702499*(vu**8)
   dbar1 = (Vmbar-bbar)/Rbar
   dbar2 = (-0.8946*(abar/Pbar)/(Vmbar**2.846))*(Vmbar-bbar)/Rbar
```

```
  dbar3 = ((-2.6013*(Pbar**1.6013)*np.exp(-18.608))*abar/(Vmbar**2.846))*(Vmbar-bbar)/
Rbar
  dbar4 = 0
  if Pbar > 100:
    #calculate dbar4
    vu = np.log(Pbar)
    db1 = -7.44171 + 7.318565*2*vu - 3.463717*(vu**2)*3 + 0.87372903*(vu**3)*4
    db2 = -0.12385414*(vu**4)*5 + 0.0098570583*(vu**5)*6 - 0.00041153723*(vu**6)*7 +
0.00000702499*(vu**7)*8
    dbt = (db1+db2)/Pbar
    dbar4 = -dbt*(Pbar + abar/(Vmbar**2.846))/Rbar
  dTdP = (dbar1+dbar2+dbar3+dbar4)*(10**-5)
  #dT/dP at constant volume
  return dTdP
def dTdVm_eos(P,Vm,R=8.31446,delVm=10.**(-9)):
  #dT/dVm at constant pressure
  Pbar = P*(10**-5)
  Vmbar = Vm*(10**6)
  Rbar = R*10
  abar = np.exp(19.599 - (Pbar**2.6013)*np.exp(-18.608)) * (Pbar**-0.8946)
  if Pbar < 100:
    bbar = 16.74519884
  else:
    vu = np.log(Pbar)
    bbar = 20.285 - 7.44171*vu + 7.318565*(vu**2) - 3.463717*(vu**3) + 0.87372903*(vu**4)
- 0.12385414*(vu**5) + 0.0098570583*(vu**6) - 0.00041153723*(vu**7) +
0.00000702499*(vu**8)
  dbar1 = (Pbar + abar/(Vmbar**2.846))/Rbar
  dbar2 = -2.846*(abar/(Vmbar**3.846))*(Vmbar - bbar)/Rbar
  dTdVm = (dbar1+dbar2)*(10**6)
  return dTdVm
#Approximation (or for simple EoS exact value) of inversion for Vm or P; use as rarely as
possible
def Vm_eos_appx(P,T,R=8.31446,tal=0.0001):
  Vmg = R*T/P #ideal gas
  ear = T_eos(P,Vmg) - T
  for ug in range(10000):
    if abs(ear) <= tal:
      return Vmg
    elif abs(ear) > tal:
      Vmgn = Vmg - ear/dTdVm_eos(P,Vmg)
      Vmg = 1.0*Vmgn
      ear = T_eos(P,Vmg) - T
  print("Error: Newton's method for Vm failed to converge.")
  return R*T/P
def P_eos_appx(Vm,T,R=8.31446,tal=0.0001):
  Pg = R*T/Vm #ideal gas
  ear = T_eos(Pg,Vm) - T
  for ug in range(10000):
    if abs(ear) <= tal:
      return Pg
    elif abs(ear) > tal:
      Pgn = Pg - ear/dTdP_eos(Pg,Vm)
      Pg = 1.0*Pgn
      ear = T_eos(Pg,Vm) - T
```

```
  print("Error: Newton's method for P failed to converge.")
  return R*T/Vm
def dTdP_esti(P,Vm,R=8.31446,delP=0.01):
  #dT/dP at constant volume
  Thigh = T_eos(P+(delP/2.),Vm)
  Tlow = T_eos(P-(delP/2.),Vm)
  dTdP = (Thigh - Tlow)/delP
  return dTdP
def dTdVm_esti(P,Vm,R=8.31446,delVm=10.**(-9)):
  #dT/dVm at constant pressure
  Thigh = T_eos(P,Vm+(delVm/2.))
  Tlow = T_eos(P,Vm-(delVm/2.))
  dTdVm = (Thigh-Tlow)/delVm
  return dTdVm

#Calculate n0s and slopes from Ns and Ms, and vice versa
#NOTE: Ms are true mass per unit volume
def NsMs(binbounds,n0s,slopes,rho):
  Ns = np.zeros(len(binbounds)-1)
  Ms = np.zeros(len(binbounds)-1)
  for a in range(len(Ns)):
    dbx = 0.5*(binbounds[a+1]-binbounds[a])
    if (n0s[a] + dbx*slopes[a] >= 0) and (n0s[a] - dbx*slopes[a] >= 0):
      #particles at all radii in bin
      Ns[a] = n0s[a]*2.*dbx
      Ms[a] = (4*np.pi*rho/3.)*((0.2*(binbounds[a+1]**5 - binbounds[a]**5)*slopes[a])
+(0.25*(binbounds[a+1]**4 - binbounds[a]**4)*(n0s[a] - 0.5*slopes[a]*(binbounds[a] +
binbounds[a+1])))
    elif (n0s[a] + dbx*slopes[a] < 0) and (n0s[a] - dbx*slopes[a] >= 0):
      #particles at small end of bin, but not large end
      upb = 0.5*binbounds[a] + 0.5*binbounds[a+1] - n0s[a]/slopes[a]
      dbp = 0.5*(upb - binbounds[a])
      Ns[a] = (n0s[a] - dbx*slopes[a])*dbp
      Ms[a] = (4*np.pi*rho/3.)*((0.2*(upb**5 - binbounds[a]**5)*slopes[a])+(0.25*(upb**4 -
binbounds[a]**4)*(n0s[a] - 0.5*slopes[a]*(binbounds[a] + binbounds[a+1])))
    elif (n0s[a] + dbx*slopes[a] >= 0) and (n0s[a] - dbx*slopes[a] < 0):
      #particles at large end of bin, but not at small end
      lob = 0.5*binbounds[a] + 0.5*binbounds[a+1] - n0s[a]/slopes[a]
      dbp = 0.5*(binbounds[a+1] - lob)
      Ns[a] = (n0s[a] + dbx*slopes[a])*dbp
      Ms[a] = (4*np.pi*rho/3.)*((0.2*(binbounds[a+1]**5 - lob**5)*slopes[a])
+(0.25*(binbounds[a+1]**4 - lob**4)*(n0s[a] - 0.5*slopes[a]*(binbounds[a] +
binbounds[a+1])))
    else:
      #No particles in bin
      Ns[a] = 0
      Ms[a] = 0
  return Ns,Ms

def n0sslopes(binbounds,Ns,Ms,rho,showdetails=0):
  n0s = np.zeros(len(binbounds)-1)
  slopes = np.zeros(len(binbounds)-1)
  for a in range(len(n0s)):
    if Ns[a] <= 0:
      r3av = binbounds[a]**3
```

```
    else:
      r3av = 3*Ms[a]/(4.*np.pi*rho*Ns[a]) #average r^3 in bin
    #Critical values of r^3 for upper or lower end of bin to have n(r)=0 and the rest of the
bin positive
    r3rise = (.4*(binbounds[a+1]**5) - .5*(binbounds[a+1]**4)*binbounds[a]
+ .1*(binbounds[a]**5))/((binbounds[a+1] - binbounds[a])**2)
    r3fall = (.1*(binbounds[a+1]**5) - .5*(binbounds[a]**4)*binbounds[a+1]
+ .4*(binbounds[a]**5))/((binbounds[a+1] - binbounds[a])**2)
    if r3av >= ((binbounds[a+1])**3):
      #Average is too large to work
      n0s[a] = 0
      slopes[a] = 0
      print('Error in n0sslopes: average mass too high in bin',a)
    elif r3av < ((binbounds[a])**3):
      #Average is too small to work
      n0s[a] = 0
      slopes[a] = 0
      print('Error in n0sslopes: average mass too low in bin',a)
    elif r3av == ((binbounds[a])**3):
      #Average is too small to work, or consequence of Ns=0
      n0s[a] = 0
      slopes[a] = 0
    elif r3av > r3rise:
      #Particles at top of bin, but not bottom
      #Try to devise a solution
      bquar = binbounds[a+1]
      r0 = 0.5*(bquar + binbounds[a])
      larray = np.roots([1.0,2.0*bquar,3.0*(bquar**2),4.0*(bquar**3) - 10.0*r3av])
      lquar = 0.
      for tt in range(len(larray)):
        lposs = larray[tt]
        if lposs.real >= binbounds[a] and lposs.real < binbounds[a+1]:
          if abs(lposs.imag) < 10**-10:
            if showdetails==1:
              print(a,lposs)
            lquar = lposs.real + 0.0
      if showdetails==1:
        print(a,lquar)
      if lquar ==0:
        print('Error in n0sslopes: cubic solver did not converge in bin',a)
      #lquar is the lower bound within the bin
      slopes[a] = 2*Ns[a]/((bquar - lquar)**2)
      n0s[a] = (r0 - lquar)*slopes[a]
    elif r3av < r3fall:
      #Particles at bottom of bin, but not top
      #Try to devise a solution
      bquar = binbounds[a]
      r0 = 0.5*(bquar + binbounds[a+1])
      uarray = np.roots([1.0,2.0*bquar,3.0*(bquar**2),4.0*(bquar**3) - 10.0*r3av])
      uquar = 0.
      for tt in range(len(uarray)):
        uposs = uarray[tt]
        if uposs.real >= binbounds[a] and uposs.real < binbounds[a+1]:
          if abs(uposs.imag) < 10**-10:
            if showdetails==1:
```

```
                print(a,uposs)
              uquar = uposs.real + 0.0
        if showdetails==1:
          print(a,uquar)
        if uquar ==0:
          print('Error in n0sslopes: cubic solver did not converge in bin',a)
        #uquar is the upper bound within the bin
        slopes[a] = -2*Ns[a]/((uquar - bquar)**2)
        n0s[a] = (r0 - uquar)*slopes[a]
      elif r3av <= r3rise and r3av >= r3fall:
        #Particles throughout bin
        dbx = 0.5*(binbounds[a+1]-binbounds[a])
        n0s[a] = 0.5*Ns[a]/dbx
        slopes[a] = (r3av*Ns[a] -
n0s[a]*(0.25*(binbounds[a+1]**4)-0.25*(binbounds[a]**4)))/(0.2*(binbounds[a+1]**5)-
0.2*(binbounds[a]**5) - (0.125*(binbounds[a+1]**4)-
0.125*(binbounds[a]**4))*(binbounds[a+1]+binbounds[a]))
  return n0s, slopes

def mu_avg(mu_dry,mus,fs):
  #Average molar mass
  fJs = np.zeros(len(fs))
  mu = 1*mu_dry
  for a in range(len(fs)):
    fJs[a] = mu_dry*fs[a]/(mus[a]*(1-fs[a]) + fs[a]*mu_dry) #*mole* fraction vapor
    mu = mu + fJs[a]*(mus[a]-mu_dry)
  return mu

def stepgrow6(cloudN,cloudM,binbounds,rhosliquid,Eij,vrel,delt):
  mmean = np.zeros([len(rhosliquid),len(binbounds)-1])
  Msnew = np.zeros([len(rhosliquid),len(binbounds)-1])
  Nsnew = np.zeros([len(rhosliquid),len(binbounds)-1])
  r0s = np.zeros(len(binbounds)-1)
  for b in range(len(binbounds)-1):
    r0s[b] = (binbounds[b] + binbounds[b+1])/2.
  for a in range(len(rhosliquid)):
    Nsloc = cloudN[a,:].copy()
    Msloc = cloudM[a,:].copy()
    n0s, slopes = n0sslopes(binbounds,Nsloc,Msloc,rhosliquid[a])
    for b in range(len(binbounds)-1):
      Nsnew[a,b] = 1*cloudN[a,b]
      Msnew[a,b] = 1*cloudM[a,b]
      if cloudN[a,b] != 0:
        mmean[a,b] = cloudM[a,b]/cloudN[a,b]
      #else:
        #mmean[a,b] = (1/3.0)*np.pi*rhosliquid[a]*(binbounds[b+1]**4 - binbounds[b]**4)
    for b in range(len(binbounds)-1):
      gtb = r0s[b] - binbounds[b]
      for c in range(len(binbounds)-1):
        if b >= c:
          lambdabc = min(Eij[a,b,c] * np.pi * (r0s[b]**2 + r0s[c]**2) * vrel[a,b,c] * cloudN[a,c]
* delt,1.0)
          Nsnew[a,c] = Nsnew[a,c] - lambdabc*cloudN[a,b]
          Msnew[a,c] = Msnew[a,c] - lambdabc*cloudN[a,b]*mmean[a,c]
          Msnew[a,b] = Msnew[a,b] + lambdabc*cloudN[a,b]*mmean[a,c]
```

```
            rx = (binbounds[b+1]**3 - r0s[c]**3)**(1/3.0)
            if n0s[b] + gtb*slopes[b] >= 0:
               rxx = binbounds[b+1]
               if n0s[b] + (rx-r0s[b])*slopes[b] > 0:
                  #particles throughout bin, at least in relevant part thereof
                  rxx = binbounds[b+1]
                  Nxx = n0s[b]*(rxx-rx) - slopes[b]*r0s[b]*(rxx-rx) +
slopes[b]*(rxx-rx)*(rxx+rx)/2.0
                  Mxx = (np.pi*rhosliquid[a]/3.0)*(rxx**4 - rx**4)*(n0s[b]-slopes[b]*r0s[b]) +
(4*np.pi*rhosliquid[a]/15.0)*slopes[b]*(rxx**5-rx**5)
               elif n0s[b] + gtb*slopes[b] > 0:
                  #particles only in uppermost part of bin
                  rw = r0s[b] - n0s[b]/slopes[b]
                  Nxx = n0s[b]*(rxx-rw) - slopes[b]*r0s[b]*(rxx-rw) + slopes[b]*(rxx-
rw)*(rxx+rw)/2.0
                  Mxx = (np.pi*rhosliquid[a]/3.0)*(rxx**4 - rw**4)*(n0s[b]-slopes[b]*r0s[b]) +
(4*np.pi*rhosliquid[a]/15.0)*slopes[b]*(rxx**5-rw**5)
               else:
                  Nxx = 0.0
                  Mxx = 0.0
            elif n0s[b] + (rx-r0s[b])*slopes[b] > 0:
               rxx = r0s[b] - n0s[b]/slopes[b]
               Nxx = n0s[b]*(rxx-rx) - slopes[b]*r0s[b]*(rxx-rx) + slopes[b]*(rxx-rx)*(rxx+rx)/2.0
               Mxx = (np.pi*rhosliquid[a]/3.0)*(rxx**4 - rx**4)*(n0s[b]-slopes[b]*r0s[b]) +
(4*np.pi*rhosliquid[a]/15.0)*slopes[b]*(rxx**5-rx**5)
            else:
               Nxx = 0.0
               Mxx = 0.0
            if (b+1)<len(n0s):
               Msnew[a,b+1] = Msnew[a,b+1] + Mxx*lambdabc + mmean[a,c]*Nxx*lambdabc
               Msnew[a,b] = Msnew[a,b] - Mxx*lambdabc - mmean[a,c]*Nxx*lambdabc
               Nsnew[a,b+1] = Nsnew[a,b+1] + Nxx*lambdabc
               Nsnew[a,b] = Nsnew[a,b] - Nxx*lambdabc
      for b in range(len(binbounds)-1):
         if Nsnew[a,b] > 0:
            if Msnew[a,b]/Nsnew[a,b] <= 4*np.pi*rhosliquid[a]*((binbounds[b])**3)/3:
               Nsnew[a,b] = 0
               Msnew[a,b] = 0
   return Nsnew, Msnew


#Particle charging, for multiple particle species
def
dQdt6(Ns,binbounds,velocities,mus,rhosadj,Qcoeffs,ioncharges=[],ionnumbers=[],ionvelocit
ies=[]):
   #velocities is particle fall velocities, rhosadj is ratio of liquid/actual density cuberoots for
use in solid cases
   r0s = np.zeros(len(binbounds)-1)
   dQidt = np.zeros([len(mus),len(binbounds)-1])
   #calculate r0 as bin center radius
   for s in range(len(binbounds)-1):
      r0s[s] = (binbounds[s+1] + binbounds[s])/2.0
   #charge transfer
   for au in range(len(mus)):
      for bu in range(len(binbounds)-1):
```

```
        dQitot = 0.0
        for cu in range(len(binbounds)-1):
          for du in range(len(mus)):
            rG = min(rhosadj[au]*r0s[bu],rhosadj[au]*r0s[cu])
            #transferred charge dependent on molar mass
            #if unknown species, don't transfer any charge
            #subject to change later
            if mus[au]>.0175 and mus[au]<.0185:
              if mus[du]>.0175 and mus[du]<.0185:
                #Water-water charging
                if rG <= .000111:
                  Gr = .0000271*((1000000.0*rG)**2.7)
                else:
                  Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = ((velocity/3.0)**2.5)*Gr*(10**-15)
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                if bu < cu:
                  dQitot = dQitot + dQi
                else:
                  dQitot = dQitot - dQi
              elif mus[du]>.0165 and mus[du]<.0175:
                #Ammonia-water charging
                #Triboelectric: water gets positive charge
                #Per Lee et al. 2018 and a lot of dubious extrapolation
                #Only goes as v^1.5 based on Lesprit et al. 2020
                if rG <= .000111:
                  Gr = .0000271*((1000000.0*rG)**2.7)
                else:
                  Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = ((velocity/3.0)**2.5)*(20.68/velocity)*Gr*(10**-15) #or 0.482/velocity if
pessimistic
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                dQitot = dQitot + abs(dQi)
              elif mus[du]>.0505 and mus[du]<.0515:
                #NH4SH-water charging
                #Triboelectric: water gets positive charge
                #Calculated as a proportional difference, as per dielectric constants
                if rG <= .000111:
                  Gr = .0000271*((1000000.0*rG)**2.7)
                else:
                  Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = ((velocity/3.0)**2.5)*(2.26/velocity)*Gr*(10**-15) #or 0.482/velocity if
pessimistic
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                dQitot = dQitot + abs(dQi)
            elif mus[au]>.0165 and mus[au]<.0175:
              if mus[du]>.0165 and mus[du]<.0175:
                #Ammonia-ammonia charging
                #Use same equation as for water, multiplied by smaller constant (ratio of
dielectric constants)
                if rG <= .000111:
                  Gr = .0000271*((1000000.0*rG)**2.7)
```

```
                else:
                    Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = (25/80.)*((velocity/3.0)**2.5)*Gr*(10**-15)
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                if bu < cu:
                    dQitot = dQitot + dQi
                else:
                    dQitot = dQitot - dQi
            elif mus[du]>.0175 and mus[du]<.0185:
                #Water-ammonia charging
                #Triboelectric: ammonia gets negative charge
                #Per Lee et al. 2018 and a lot of dubious extrapolation
                #Only goes as v^1.5 based on Lesprit et al. 2020
                if rG <= .000111:
                    Gr = .0000271*((1000000.0*rG)**2.7)
                else:
                    Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = ((velocity/3.0)**2.5)*(20.68/velocity)*Gr*(10**-15) #or 0.482/velocity if
pessimistic
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                dQitot = dQitot - abs(dQi)
            elif mus[du]>.0505 and mus[du]<.0515:
                #NH4SH-ammonia charging
                #Triboelectric: ammonia gets negative charge
                #Calculated as a proportional difference, as per dielectric constants
                if rG <= .000111:
                    Gr = .0000271*((1000000.0*rG)**2.7)
                else:
                    Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = ((velocity/3.0)**2.5)*(18.42/velocity)*Gr*(10**-15) #or 0.482/velocity if
pessimistic
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                dQitot = dQitot - abs(dQi)
        elif mus[au]>.0505 and mus[au]<.0515:
            #NH4SH
            if mus[du]>.0505 and mus[du]<.0515:
                #NH4SH-NH4SH
                #Use same equation as for water, multiplied by smaller constant (ratio of
dielectric constants)
                if rG <= .000111:
                    Gr = .0000271*((1000000.0*rG)**2.7)
                else:
                    Gr = .0988*((1000000.0*rG)**0.98)
                velocity = abs(velocities[au,bu] - velocities[du,cu])
                delQ = (74./80.)*((velocity/3.0)**2.5)*Gr*(10**-15)
                #Dielectric constant of 74 via extrapolation from refractive indices in Joiner
and Steffes (1991)
                dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                if bu < cu:
                    dQitot = dQitot + dQi
                else:
                    dQitot = dQitot - dQi
```

```
                elif mus[du]>.0175 and mus[du]<.0185:
                    #Water-NH4SH charging
                    #Triboelectric: NH4SH gets negative charge
                    #Calculated as a proportional difference, as per dielectric constants
                    if rG <= .000111:
                        Gr = .0000271*((1000000.0*rG)**2.7)
                    else:
                        Gr = .0988*((1000000.0*rG)**0.98)
                    velocity = abs(velocities[au,bu] - velocities[du,cu])
                    delQ = ((velocity/3.0)**2.5)*(2.26/velocity)*Gr*(10**-15) #or 0.482/velocity if
pessimistic
                    dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                    dQitot = dQitot - abs(dQi)
                elif mus[du]>.0165 and mus[du]<.0175:
                    #Ammonia-NH4SH charging
                    #Triboelectric: NH4SH gets positive charge
                    #Calculated as a proportional difference, as per dielectric constants
                    if rG <= .000111:
                        Gr = .0000271*((1000000.0*rG)**2.7)
                    else:
                        Gr = .0988*((1000000.0*rG)**0.98)
                    velocity = abs(velocities[au,bu] - velocities[du,cu])
                    delQ = ((velocity/3.0)**2.5)*(18.42/velocity)*Gr*(10**-15) #or 0.482/velocity if
pessimistic
                    dQi = delQ*Ns[du,cu]*np.pi*(r0s[bu]**2 + r0s[cu]**2)*rhosadj[au]*rhosadj[du]
                    dQitot = dQitot + abs(dQi)
            dQidt[au,bu] = dQitot*Qcoeffs[au]
    return dQidt


def dEdt6(Ns,binbounds,velocities,charges,mus):
    #as above, but charges is a vector of average particle charge per bin
    Jc = 0.0
    #calculate displacement current per bin
    for g in range(len(binbounds)-1):
        for h in range(len(mus)):
            Jcg = -Ns[h,g]*velocities[h,g]*charges[h,g]
            Jc = Jc + Jcg
    return -Jc/(8.854*(10**-12))

def Evapor6(n0s,slopes,binbounds,rho,velocities,P,Vm,f,mu_dry,mu_cond,det=0,Ptweak=0):
    #Evaporation for one species of several
    #mu_dry includes other condensible species
    T = T_eos(P,Vm)
    svpa = SVPs(P,T,np.asarray([mu_cond]))
    frs = (mu_cond/mu_dry)*(svpa[0]/((P+Ptweak) - svpa[0])) #mass ratio vapor
    fsat = frs/(1.+frs) #convert to mass fraction
    if fsat <= f:
        #return no evaporation
        dNdt = np.zeros(len(n0s))
        dMdt = np.zeros(len(n0s))
        if det > 0.5:
            print('no evaporation: fsat <=f')
            print(fsat,f)
    elif np.array_equal(np.zeros(len(n0s)),n0s) and np.array_equal(np.zeros(len(n0s)),slopes):
```

```
      #no drops, no evaporation
      dNdt = np.zeros(len(n0s))
      dMdt = np.zeros(len(n0s))
      if det > 0.5:
         print('no evaporation: n0s and slopes zero')
         print(n0s,slopes)
   else:
      #evaporation occurs
      #Generate arrays
      r0s = np.zeros(len(n0s))
      ove = np.zeros(len(n0s))
      oatot = np.zeros(len(n0s))
      dmddt = np.zeros(len(n0s))
      drddt = np.zeros(len(n0s))
      dNdt = np.zeros(len(n0s))
      dMdt = np.zeros(len(n0s))
      #Calculate evaporation rate
      fJ = mu_dry*f/(mu_cond*(1-f) + f*mu_dry) #mole fraction vapor
      Pextra = SVPs(P,T,np.asarray([mu_cond])) - fJ*(P+Ptweak) #vapor pressure missing to
reach saturation
      oth = Pextra/np.sqrt(2*np.pi*P*mu_cond*Vm) #impacts per unit area per unit time due
to thermal movement
      #units of oth and ove: mole per second per square meter
      for s in range(len(n0s)):
         r0s[s] = (binbounds[s+1] + binbounds[s])/2.0
         ove[s] = velocities[s]*Pextra/(4.*Vm*P)
         oatot[s] = np.sqrt(((ove[s])**2)+(oth**2))*np.pi #impacts per particle per unit time per
particle radius squared (moles per second per square meter)
         if det>0.5:
            print(P,Pextra)
         #Calculate Cevap via Kessler (1969)
         #Combine with Knudsen that has evaporation as square root of mu/(PVm) rather than
just mu/(PVm)
         #Evaporation as r^2 within bins (Knudsen), r^1.6 between bins (Kessler)
         if mu_cond > .0175 and mu_cond < .0185:
            #water
            Cevap = 0.0001527/((r0s[s])**0.4)
         else:
            #Unknown, default to water evaporation values
            Cevap = 0.0001527/((r0s[s])**0.4)
         dmddt[s] = Cevap*oatot[s]*mu_cond #mass loss per particle per second per particle
radius squared (kg per second per square meter)
         drddt[s] = dmddt[s]/(4.*np.pi*rho) #rate of change in radius per particle per second
         if n0s[s] + slopes[s]*(binbounds[s+1]-r0s[s]) >= 0:
            R3 = (binbounds[s+1]**3 - binbounds[s]**3)/3.
            R4 = 0.25*(binbounds[s+1]**4 - binbounds[s]**4)
         elif n0s[s] + slopes[s]*(binbounds[s]-r0s[s]) >= 0:
            rupeb = r0s[s] - n0s[s]/slopes[s]
            R3 = (rupeb**3 - binbounds[s]**3)/3.
            R4 = 0.25*(rupeb**4 - binbounds[s]**4)
         #Calculate rate of particle number change
         if n0s[s] + slopes[s]*(binbounds[s]-r0s[s]) >= 0:
            dNdt[s] = dNdt[s] - (n0s[s] + slopes[s]*(binbounds[s] - r0s[s]))*drddt[s]
            if s>0:
               dNdt[s-1] = dNdt[s-1] + (n0s[s] + slopes[s]*(binbounds[s] - r0s[s]))*drddt[s]
```

```
            #Calculate rate of particle mass change
            #Firstly, effect of dNdt
            dMdt[s] = dMdt[s] - (n0s[s] + slopes[s]*(binbounds[s] -
r0s[s]))*drddt[s]*rho*(4*np.pi/3.)*((binbounds[s])**3)
            if s>0:
               dMdt[s-1] = dMdt[s-1] + (n0s[s] + slopes[s]*(binbounds[s] -
r0s[s]))*drddt[s]*rho*(4*np.pi/3.)*((binbounds[s])**3)
         #Secondly, direct evaporation
         dMdt[s] = dMdt[s] - (R3*(n0s[s] - slopes[s]*r0s[s]) + R4*slopes[s])*dmddt[s]
   return dNdt,dMdt


#Entrainment rate
def phi(Vm_fall,mu_fall,radius,g):
   phiret = -0.2*Vm_fall/(radius*g*mu_fall)
   return phiret

#Velocity change
def dwdP(P,Vm_rise,Vm_fall,mu_rise,mu_fall,l_rise,w,radius,g):
   Thorn = (Vm_rise*mu_fall)/(Vm_fall*mu_rise) - 1.
   dwdPn = (l_rise*Vm_rise)/(w*mu_rise) - (Thorn*Vm_fall)/(w*mu_fall) -
w*phi(Vm_rise,mu_rise,radius,g)
   return dwdPn

def
KdwdP(P,Trise,Tfall,lcondensate,frise,ffall,w,radius=5000.0,g=24.79,R=8.31446,mu=0.0022,e
psilon=8.2):
   #Acceleration/deceleration of plume due to buoyancy and entrainment (ideal gas)
   phi = -.2*R*Trise/(radius*mu*P*g) #entrainment rate
   dwdPn = -R*(Trise*(1.0-lcondensate)*((1.0 + frise/epsilon)/(1.0 + frise))-Tfall*((1.0 +
ffall/epsilon)/(1.0 + ffall)))/(P*mu*w) - w*phi
   return dwdPn



#Moist adiabat (w/o plume motion) with multiple condensibles
def
dVmdPfall6(P,Vm_fall,fs_fall,mu_dry,Vm_out,radius,g,mus_cond,Ls,Cp=14500.0,flage=0):
   T = T_eos(P,Vm_fall)
   T_out = T_eos(P,Vm_out)
   GP = dTdP_eos(P,Vm_fall)
   GVm = dTdVm_eos(P,Vm_fall)
   mu_fall = mu_avg(mu_dry,mus_cond,fs_fall)
   ffs_fall = np.zeros(len(mus_cond))
   #Calculate mass ratios (ffs_fall) from mass fractions (fs_fall)
   for b in range(len(mus_cond)):
      ffs_fall[b] = fs_fall[b]/(1. - np.sum(fs_fall))
   SpPs = SVPs(P,T,mus_cond)
   fs_sat = np.zeros(len(mus_cond))
   for b in range(len(mus_cond)):
      fs_sat[b] = mus_cond[b]*SpPs[b]/(P*mu_dry)
   if P==0:
      print('Error: P=0')
      print(P,Vm_fall,mu_fall,mu_dry,Vm_out,radius,g,mus_cond,Ls)
   if mu_dry==0:
```

```
        print('Error: mu_dry=0')
        print(P,Vm_fall,mu_fall,mu_dry,Vm_out,radius,g,mus_cond,Ls)
    if Cp==0:
        print('Error: Cp=0')
        print(P,Vm_fall,mu_fall,mu_dry,Vm_out,radius,g,mus_cond,Ls)
    #Does condensation occur?
    fstogg = 0
    for b in range(len(mus_cond)):
        if fs_sat[b] <= ffs_fall[b]:
            fstogg = 1
    if fstogg == 0:
        #No condensation, dry adiabat
        dTdP = Vm_fall/(mu_dry*Cp)
    elif fstogg == 1:
        #condensation, moist adiabat
        crnum = Vm_fall/mu_fall
        crden = 1*Cp
        for b in range(len(mus_cond)):
            if fs_sat[b] <= ffs_fall[b]:
                crnum = crnum + (Ls[b]*fs_sat[b]/P)
                crden = crden + (mus_cond[b]*Ls[b]*Ls[b]*fs_sat[b])/(Vm_fall*T*P)
        dTdP = crnum/crden
    dVmdPn = (dTdP - dTdP_eos(P,Vm_fall))/dTdVm_eos(P,Vm_fall)
    if flage>0:
        print('dT/dP',dTdP)
    return dVmdPn



#Moist adiabat (with plume motion and multiple condensibles)
def
dVmdPrise6(P,Vm_rise,Vm_fall,fs_rise,radius,g,mus_cond,mu_dry,Ls,Cp=14500.0,flage=0):
    T = T_eos(P,Vm_rise)
    Tfall = T_eos(P,Vm_fall)
    GP = dTdP_eos(P,Vm_rise)
    GVm = dTdVm_eos(P,Vm_rise)
    ffs_rise = np.zeros(len(fs_rise))
    #Calculate mass ratios (ffs_rise) from mass fractions (rise)
    for b in range(len(mus_cond)):
        ffs_rise[b] = fs_rise[b]/(1. - np.sum(fs_rise))
    SpPs = SVPs(P,T,mus_cond)
    fs_sat = np.zeros(len(mus_cond))
    for b in range(len(mus_cond)):
        fs_sat[b] = mus_cond[b]*SpPs[b]/(P*mu_dry)
    #Does condensation occur?
    fstogg = 0
    for b in range(len(mus_cond)):
        if fs_sat[b] <= ffs_rise[b]:
            fstogg = 1
    mu_rise = mu_avg(mu_dry,mus_cond,fs_rise)
    if fstogg == 0:
        if flage>0:
            print('no condensation')
        #No condensation, dry adiabat
        dTdP = Vm_rise/(mu_rise*Cp) - (T-Tfall)*phi(Vm_rise,mu_dry,radius,g)
```

```
    elif fstogg == 1:
        #condensation, moist adiabat
        ph = phi(Vm_rise,mu_rise,radius,g)
        crnum = Vm_rise/mu_rise - ph*Cp*(T-Tfall)
        crden = 1*Cp
        for b in range(len(mus_cond)):
            if fs_sat[b] <= ffs_rise[b]:
                crnum = crnum + (Ls[b]*fs_sat[b]/P) - (ph*Ls[b]*fs_sat[b])
                crden = crden + (mus_cond[b]*Ls[b]*Ls[b]*fs_sat[b])/(Vm_rise*T*P)
        if flage>0:
            print('numerator and denominator',crnum,crden)
        dTdP = crnum/crden
    dVmdPn = (dTdP - dTdP_eos(P,Vm_rise))/dTdVm_eos(P,Vm_rise)
    if flage>0:
        print('fs used for comparison',fs_sat,ffs_rise)
        print('dT/dP',dTdP)
    return dVmdPn


def SVPs(P,T,mus_cond):
    SVPs = np.zeros(len(mus_cond))
    for a in range(len(mus_cond)):
        if mus_cond[a] > .0175 and mus_cond[a] < .0185:
            #Water vapor pressure, from Lowe 1977
            a0 = 6984.505294
            a1 = -188.9039310
            a2 = 2.133357675
            a3 = -0.01288580973
            a4 = (4.393587233)*(10**-5)
            a5 = (-8.023923082)*(10**-8)
            a6 = (6.136820929)*(10**-11)
            mb = a0 + T*(a1 + T*(a2 + T*(a3 + T*(a4 + T*(a5 + T*a6)))))
            pasc = max(mb*100.0,0.0)
            SVPs[a] = pasc
        elif mus_cond[a] < .0175 and mus_cond[a] > .0165:
            #Ammonia vapor pressure
            #Source: Lange's Handbook of Chemistry
            larth = 6.67956 - 1002.711/(T - 25.215)
            larth = min(larth,12.)
            pasc = max(10**(3 + larth),0.0)
            SVPs[a] = pasc
        elif mus_cond[a] < .0515 and mus_cond[a] > .0505:
            #NH4SH vapor pressure
            #Source: Sanchez-Lavega (2004)
            #An effective 'NH4SH vapor pressure', although the vapor is really H2S with an
equilibrium constant
            #That is, really it's a chemical reaction
            Ctteh = 120.678 + (-2915.7/T - 1.760*np.log(T) + 0.00078*T)/.167 #From Zuchowski et
al. (2009), previously 75.678
            pasc = max(100000*np.exp(Ctteh),0.0)
            SVPs[a] = pasc
        else:
            #unknown
            SVPs[a] = 0
    return SVPs
```

```
def Tstrato(P,strato):
  #Stratospheric temperature by planet
  if strato=='j':
    #Jupiter
    if P < 0.5*(12.57+28.24):
      Pl = 0.5*(5.593+12.57)
      Ph = 0.5*(12.57+28.24)
      Tl = 157.2 + (158.2-157.2)*(P - 5.593)/(12.57 - 5.593)
      Th = 158.2 + (157.4-158.2)*(P - 12.57)/(28.24 - 12.57)
    elif (0.5*(12.57+28.24) < P) and (P < 0.5*(28.24+61.92)):
      Ph = 0.5*(28.24+61.92)
      Pl = 0.5*(12.57+28.24)
      Th = 157.4 + (168.6-157.4)*(P - 28.24)/(61.92 - 28.24)
      Tl = 158.2 + (157.4-158.2)*(P - 12.57)/(28.24 - 12.57)
    elif (0.5*(61.92+134.2) > P) and (P > 0.5*(28.24+61.92)):
      Pl = 0.5*(28.24+61.92)
      Ph = 0.5*(61.92+134.2)
      Tl = 157.4 + (168.6-157.4)*(P - 28.24)/(61.92 - 28.24)
      Th = 168.6 + (160.5-168.6)*(P - 61.92)/(134.2 - 61.92)
    elif (0.5*(61.92+134.2) < P) and (P < 0.5*(134.2+307.9)):
      Ph = 0.5*(134.2+307.9)
      Pl = 0.5*(61.92+134.2)
      Th = 160.5 + (149.8-160.5)*(P - 134.2)/(307.9 - 134.2)
      Tl = 168.6 + (160.5-168.6)*(P - 61.92)/(134.2 - 61.92)
    elif (0.5*(307.9+717.7) > P) and (P > 0.5*(134.2+307.9)):
      Pl = 0.5*(134.2+307.9)
      Ph = 0.5*(307.9+717.7)
      Tl = 160.5 + (149.8-160.5)*(P - 134.2)/(307.9 - 134.2)
      Th = 149.8 + (158.1-149.8)*(P - 307.9)/(717.7 - 307.9)
    elif (0.5*(307.9+717.7) < P) and (P < 0.5*(717.7+1640.)):
      Ph = 0.5*(717.7+1640.)
      Pl = 0.5*(307.9+717.7)
      Th = 158.1 + (143.8-158.1)*(P - 717.7)/(1640. - 717.7)
      Tl = 149.8 + (158.1-149.8)*(P - 307.9)/(717.7 - 307.9)
    elif (0.5*(1640.+4374.) > P) and (P > 0.5*(717.7+1640.)):
      Pl = 0.5*(717.7+1640.)
      Ph = 0.5*(1640.+4374.)
      Tl = 158.1 + (143.8-158.1)*(P - 717.7)/(1640. - 717.7)
      Th = 143.8 + (122.6-143.8)*(P - 1640.)/(4374. - 1640.)
    elif (0.5*(1640.+4374.) < P) and (P < 0.5*(4374.+13580.)):
      Ph = 0.5*(4374.+13580.)
      Pl = 0.5*(1640.+4374.)
      Th = 122.6 + (113.2-122.6)*(P - 4374.)/(13580. - 4374.)
      Tl = 143.8 + (122.6-143.8)*(P - 1640.)/(4374. - 1640.)
    elif P > 0.5*(4374.+13580.):
      Pl = 0.5*(4374.+13580.)
      Ph = 0.5*(13580.+35150.)
      Tl = 122.6 + (113.2-122.6)*(P - 4374.)/(13580. - 4374.)
      Th = 113.2 + (122.9-113.2)*(P - 13580.)/(35150. - 13580.)
  elif strato=='u':
    #Uranus
    if P < 0.5*(100.+158.):
```

```
      Pl = 0.5*(56.+100.)
      Ph = 0.5*(100.+158.)
      Tl = 116.2 + (76.4-116.2)*(P - 56.)/(100. - 56.)
      Th = 76.4 + (70.3-76.4)*(P - 100.)/(158. - 100.)
    elif (0.5*(100.+158.) <= P) and (P < 0.5*(158.+251.)):
      Pl = 0.5*(100.+158.)
      Ph = 0.5*(158.+224.)
      Tl = 76.4 + (70.3-76.4)*(P - 100.)/(158. - 100.)
      Th = 70.3 + (68.0-70.3)*(P - 158.)/(251. - 158.)
    elif (0.5*(158.+251.) <= P) and (P < 0.5*(447.+251.)):
      Pl = 0.5*(158.+251.)
      Ph = 0.5*(447.+251.)
      Tl = 70.3 + (68.0-70.3)*(P - 158.)/(251. - 158.)
      Th = 68.0 + (68.4-68.0)*(P - 251.)/(447. - 251.)
    elif (0.5*(447.+251.) <= P) and (P < 0.5*(447.+1000.)):
      Pl = 0.5*(447.+251.)
      Ph = 0.5*(447.+1000.)
      Tl = 68.0 + (68.4-68.0)*(P - 251.)/(447. - 251.)
      Th = 68.4 + (63.4-68.4)*(P - 447.)/(1000. - 447.)
    elif (0.5*(447.+1000.) <= P) and (P < 0.5*(1995.+1000.)):
      Pl = 0.5*(447.+1000.)
      Ph = 0.5*(1995.+1000.)
      Tl = 68.4 + (63.4-68.4)*(P - 447.)/(1000. - 447.)
      Th = 63.4 + (58.2-63.4)*(P - 1000.)/(1995. - 1000.)
    elif (0.5*(1995.+1000.) <= P) and (P < 0.5*(1995.+5012.)):
      Pl = 0.5*(1995.+1000.)
      Ph = 0.5*(1995.+5012.)
      Tl = 63.4 + (58.2-63.4)*(P - 1000.)/(1995. - 1000.)
      Th = 58.2 + (53.9-58.2)*(P - 1995.)/(5012. - 1995.)
    elif (0.5*(1995.+5012.) <= P) and (P < 0.5*(10000.+5012.)):
      Pl = 0.5*(1995.+5012.)
      Ph = 0.5*(10000.+5012.)
      Tl = 58.2 + (53.9-58.2)*(P - 1995.)/(5012. - 1995.)
      Th = 53.9 + (53.0-53.9)*(P - 5012.)/(10000. - 5012.)
    elif (0.5*(10000.+5012.) <= P) and (P < 0.5*(10000.+19953.)):
      Pl = 0.5*(10000.+5012.)
      Ph = 0.5*(10000.+19953.)
      Tl = 53.9 + (53.0-53.9)*(P - 5012.)/(10000. - 5012.)
      Th = 53.0 + (54.3-53.0)*(P - 10000.)/(19953. - 10000.)
    elif 0.5*(10000.+19953.) <= P:
      Pl = 0.5*(10000.+19953.)
      Ph = 0.5*(35481.+19953.)
      Tl = 53.0 + (54.3-53.0)*(P - 10000.)/(19953. - 10000.)
      Th = 54.3 + (57.2-54.3)*(P - 19953.)/(35481. - 19953.)
  elif strato=='n':
    #Neptune
    if P < 0.5*(100.+71.):
      Pl = 0.5*(35.+71.)
      Ph = 0.5*(71.+100.)
      Tl = 130.6 + (127.1-130.6)*(P - 35.)/(71. - 35.)
      Th = 127.1 + (123.9-127.1)*(P - 71.)/(100. - 71.)
    elif (0.5*(100.+71.) <= P) and (P < 0.5*(100.+224.)):
      Pl = 0.5*(71.+100.)
      Ph = 0.5*(100.+224.)
      Tl = 127.1 + (123.9-127.1)*(P - 71.)/(100. - 71.)
```

```
      Th = 123.9 + (109.9-123.9)*(P - 100.)/(224. - 100.)
    elif (0.5*(100.+224.) <= P) and (P < 0.5*(447.+224.)):
      Pl = 0.5*(100.+224.)
      Ph = 0.5*(447.+224.)
      Tl = 123.9 + (109.9-123.9)*(P - 100.)/(224. - 100.)
      Th = 109.9 + (90.9-109.9)*(P - 224.)/(447. - 224.)
    elif (0.5*(447.+224.) <= P) and (P < 0.5*(447.+1000.)):
      Pl = 0.5*(447.+224.)
      Ph = 0.5*(447.+1000.)
      Tl = 109.9 + (90.9-109.9)*(P - 224.)/(447. - 224.)
      Th = 90.9 + (69.7-90.9)*(P - 447.)/(1000. - 447.)
    elif (0.5*(447.+1000.) <= P) and (P < 0.5*(1995.+1000.)):
      Pl = 0.5*(447.+1000.)
      Ph = 0.5*(1995.+1000.)
      Tl = 90.9 + (69.7-90.9)*(P - 447.)/(1000. - 447.)
      Th = 69.7 + (60.4-69.7)*(P - 1000.)/(1995. - 1000.)
    elif (0.5*(1995.+1000.) <= P) and (P < 0.5*(1995.+5012.)):
      Pl = 0.5*(1995.+1000.)
      Ph = 0.5*(1995.+5012.)
      Tl = 69.7 + (60.4-69.7)*(P - 1000.)/(1995. - 1000.)
      Th = 60.4 + (54.1-60.4)*(P - 1995.)/(5012. - 1995.)
    elif (0.5*(1995.+5012.) <= P) and (P < 0.5*(10000.+5012.)):
      Pl = 0.5*(1995.+5012.)
      Ph = 0.5*(10000.+5012.)
      Tl = 60.4 + (54.1-60.4)*(P - 1995.)/(5012. - 1995.)
      Th = 54.1 + (51.8-54.1)*(P - 5012.)/(10000. - 5012.)
    elif (0.5*(10000.+5012.) <= P) and (P < 0.5*(10000.+19953.)):
      Pl = 0.5*(10000.+5012.)
      Ph = 0.5*(10000.+19953.)
      Tl = 54.1 + (51.8-54.1)*(P - 5012.)/(10000. - 5012.)
      Th = 51.8 + (52.3-51.8)*(P - 10000.)/(19953. - 10000.)
    elif 0.5*(10000.+19953.) <= P:
      Pl = 0.5*(10000.+19953.)
      Ph = 0.5*(35481.+19953.)
      Tl = 51.8 + (52.3-51.8)*(P - 10000.)/(19953. - 10000.)
      Th = 52.3 + (55.1-52.3)*(P - 19953.)/(35481. - 19953.)
  elif strato=='s':
    #Saturn
    if P < 0.5*(100.+71.):
      Pl = 0.5*(56.+71.)
      Ph = 0.5*(71.+100.)
      Tl = 143.0 + (141.3-143.0)*(P - 56.)/(71. - 56.)
      Th = 141.3 + (141.6-141.3)*(P - 71.)/(100. - 71.)
    elif (0.5*(100.+71.) <= P) and (P < 0.5*(100.+224.)):
      Pl = 0.5*(71.+100.)
      Ph = 0.5*(100.+224.)
      Tl = 141.3 + (141.6-141.3)*(P - 71.)/(100. - 71.)
      Th = 141.6 + (137.3-141.6)*(P - 100.)/(224. - 100.)
    elif (0.5*(100.+224.) <= P) and (P < 0.5*(447.+224.)):
      Pl = 0.5*(100.+224.)
      Ph = 0.5*(447.+224.)
      Tl = 141.6 + (137.3-141.6)*(P - 100.)/(224. - 100.)
      Th = 137.3 + (126.2-137.3)*(P - 224.)/(447. - 224.)
    elif (0.5*(447.+224.) <= P) and (P < 0.5*(447.+1000.)):
      Pl = 0.5*(447.+224.)
```

```
            Ph = 0.5*(447.+1000.)
            Tl = 137.3 + (126.2-137.3)*(P - 224.)/(447. - 224.)
            Th = 126.2 + (104.8-126.2)*(P - 447.)/(1000. - 447.)
        elif (0.5*(447.+1000.) <= P) and (P < 0.5*(1995.+1000.)):
            Pl = 0.5*(447.+1000.)
            Ph = 0.5*(1995.+1000.)
            Tl = 126.2 + (104.8-126.2)*(P - 447.)/(1000. - 447.)
            Th = 104.8 + (90.5-104.8)*(P - 1000.)/(1995. - 1000.)
        elif (0.5*(1995.+1000.) <= P) and (P < 0.5*(1995.+5012.)):
            Pl = 0.5*(1995.+1000.)
            Ph = 0.5*(1995.+5012.)
            Tl = 104.8 + (90.5-104.8)*(P - 1000.)/(1995. - 1000.)
            Th = 90.5 + (82.3-90.5)*(P - 1995.)/(5012. - 1995.)
        elif (0.5*(1995.+5012.) <= P) and (P < 0.5*(10000.+5012.)):
            Pl = 0.5*(1995.+5012.)
            Ph = 0.5*(10000.+5012.)
            Tl = 90.5 + (82.3-90.5)*(P - 1995.)/(5012. - 1995.)
            Th = 82.3 + (83.4-82.3)*(P - 5012.)/(10000. - 5012.)
        elif (0.5*(10000.+5012.) <= P) and (P < 0.5*(10000.+19953.)):
            Pl = 0.5*(10000.+5012.)
            Ph = 0.5*(10000.+19953.)
            Tl = 82.3 + (83.4-82.3)*(P - 5012.)/(10000. - 5012.)
            Th = 83.4 + (89.8-83.4)*(P - 10000.)/(19953. - 10000.)
        elif 0.5*(10000.+19953.) <= P:
            Pl = 0.5*(10000.+19953.)
            Ph = 0.5*(35481.+19953.)
            Tl = 83.4 + (89.8-83.4)*(P - 10000.)/(19953. - 10000.)
            Th = 89.8 + (97.9-89.8)*(P - 19953.)/(35481. - 19953.)
    elif strato=='e':
        #Earth
        if P > 5474.9:
            T = 216.6
        elif (P <= 5474.9) and (P > 868.02):
            LPoff = np.log(P/5474.9)/np.log(868.02/5474.9)
            T = 216.6 + 12.0*LPoff
        elif (P <= 868.02) and (P > 110.91):
            LPoff = np.log(P/868.02)/np.log(110.91/868.02)
            T = 228.6 + 42.0*LPoff
        elif (P <= 110.91) and (P > 66.939):
            T = 270.6
        elif P <= 66.939:
            LPoff = np.log(P/66.939)/np.log(3.9564/66.939)
            T = 270.6 - 56.0*LPoff
    if strato != 'e':
        T = Th*(P-Pl)/(Ph-Pl) + Tl*(Ph-P)/(Ph-Pl)
    return T

def
iayer(Tbase,Pbase,fsbase,mus,Rplmin,Tsfreeze,Ssliquid,Sssolid,rhosliquid,rhossolid,LHs,g=2
4.79,mu_dry=.0022,Nfactor=1.0,NQfactor=100,NEfactor=100,NSfactor=10,strato='j'):
    P = (10**5)*Pbase #convert bars to pascals
    Pmax = (10**5)*Pbase
    Vmrin = Vm_eos_appx(P,Tbase)
    Vmfin = Vmrin + 0.0
    Rplume = Rplmin + 0.0 #initial plume radius in meters
```

```
    Cdrag = 0.5 #particle drag coefficient

    Pstep = Pbase*Nfactor #Pascals
    w = 0
    fs = fsbase.copy()
    ls = np.zeros(len(fs))
    Vmrise = Vmrin + 0.0
    Vmfall = Vmfin + 0.0
    Trise = T_eos(P,Vmrise)
    Tfall = T_eos(P,Vmfall)
    stepmax = int(P/Pstep) - 10
    Pressures = []
    Tempsrise = []
    Tempsfall = []
    Volsrise = []
    Volsfall = []
    Tempsdiff = []
    Velocities = []
    fsrise = []
    lsrise = []
    Radii = []
    binbounds = np.geomspace(.00001,10.48576,41)
    nbin = 40
    r0s = np.zeros(nbin)
    for b in range(nbin):
        r0s[b] = 0.5*binbounds[b] + 0.5*binbounds[b+1]
    rhoratios = np.zeros(len(rhossolid))
    for a in range(len(rhoratios)):
        rhoratios[a] = (rhosliquid[a]/rhossolid[a])**(1./3.)
    Cpidry = 3.5*8.31446/mu_dry #heat capacity of dry gas

    uzQQ = np.sqrt((8.0/(3.0*Cdrag))*g) #factor to use in terminal velocity, now without
density
    vrel = np.ones([len(fs),nbin,nbin])*10.0
    delt = 0.01
    togglecondense = np.zeros(len(mus))
    toggleplume = 0
    Nscloudlist = []
    Mscloudlist = []
    NPrecipitations = []
    MPrecipitations = []
    wjslist = []
    cloudN = np.zeros([len(mus),nbin])
    cloudM = np.zeros([len(mus),nbin])
    precipN = np.zeros([len(mus),nbin])
    precipM = np.zeros([len(mus),nbin])
    Pscondens = np.zeros(len(mus))
    #Pcondens=0

    for i in range(stepmax):
        Pressures.append(P)
        Volsrise.append(Vmrise)
        Volsfall.append(Vmfall)
        Tempsrise.append(Trise)
        Tempsfall.append(Tfall)
```

```
        fsrise.append(fs)
        lsrise.append(ls)
        Velocities.append(w)
        Tempsdiff.append(Trise-Tfall)
        Radii.append(Rplume)
        Pnew = P - Pstep

        fJs = np.zeros(len(fs))
        muecurr = 1*mu_dry
        for a in range(len(fs)):
          fJs[a] = mu_dry*fs[a]/(mus[a]*(1-fs[a]) + fs[a]*mu_dry) #*mole* fraction vapor
          muecurr = muecurr + fJs[a]*(mus[a]-mu_dry) #molecular weight of moist plume
        Cpcurr = 3.5*8.31446/muecurr #heat capacity as 3.5R
        flgmhn = np.zeros(len(mus))
        Vmfallnew = Vmfall -
Pstep*dVmdPfall6(P,Vmfall,flgmhn,mu_dry,Vmfall,Rplume,g,mus,LHs,Cp=Cpidry)
        if (P < 25000) or (Vmfallnew > .038):
          #stratospheric temperature profile for Tfall
          #fractional adjustment
          #can assume ideal gas for stratosphere
          fadjTf = Pstep/(100+Pstep)
          Vmfallnew = (1-fadjTf)*Vmfallnew + fadjTf*8.31446*Tstrato(Pnew,strato)/P
        ltotal = np.sum(ls)
        if w!=0:
          wnew = max(w - Pstep*dwdP(P,Vmrise,Vmfall,muecurr,mu_dry,ltotal,w,Rplume,g),0)
        else:
          Thorn = (Vmrise*mu_dry)/(Vmfall*muecurr) - 1.
          dw2dP = -2.*(ltotal*Vmrise)/muecurr + 2.*(Thorn*Vmfall)/mu_dry
          if dw2dP > 0:
            wnew = np.sqrt(Pstep*dw2dP)
          else:
            wnew = 0
        #Is there a plume?
        if w > 0:
          #plume and entrainment
          Vmrisenew = Vmrise -
Pstep*dVmdPrise6(P,Vmrise,Vmfall,fs,Rplume,g,mus,mu_dry,LHs,Cp=Cpcurr)
          Trisenew = T_eos(Pnew,Vmrisenew)
          SpPs = SVPs(Pnew,Trisenew,mus)
          fs_satnew = np.zeros(len(mus))
          for a in range(len(mus)):
            ffs_sat = (mus[a]/mu_dry)*(SpPs[a]/(P - SpPs[a]))
            fs_satnew[a] = ffs_sat/(1.+ffs_sat)
          phinew = phi(Vmrisenew*(P/Pnew),muecurr,Rplume,g)
          fracdelm = -phinew*Pstep
          #dilute vapor and particles
          fs = fs*(1.0-fracdelm)
          ls = ls*(1.0-fracdelm)
          cloudN = cloudN*(1.0-fracdelm)*(Pnew/(Pnew+Pstep))
          cloudM = cloudM*(1.0-fracdelm)*(Pnew/(Pnew+Pstep))
        else:
          Vmrisenew = Vmrise -
Pstep*dVmdPfall6(P,Vmrise,fs,mu_dry,Vmfall,Rplume,g,mus,LHs,Cp=Cpcurr)
          Trisenew = T_eos(Pnew,Vmrisenew)
          SpPs = SVPs(Pnew,Trisenew,mus)
```

```
        fs_satnew = np.zeros(len(mus))
        for a in range(len(mus)):
          ffs_sat = (mus[a]/mu_dry)*(SpPs[a]/(P - SpPs[a]))
          fs_satnew[a] = ffs_sat/(1.+ffs_sat)
      #muecrys from fs to reflect changed muecurr
      muecrys = mu_avg(mu_dry,mus,fs)
      fsnew = np.zeros(len(fs))
      fs_cond = np.zeros(len(fs))
      for a in range(len(fs)):
        if fs_satnew[a] < fs[a]:
          #condensation of new particles
          fs_cond[a] = fs[a] - fs_satnew[a]
          fsnew[a] = fs_satnew[a] + 0.0
        else:
          #no condensation, but plume area still increases
          fs_cond[a] = 0.0
          fsnew[a] = fs[a] + 0.0
      muenew = mu_avg(mu_dry,mus,fsnew)
      if w > 0:
        #increase plume area
        frdrho = (Vmrise - Vmrisenew)/Vmrisenew + (muenew-muecrys)/muenew
        if wnew > 0:
          frdRpl = 0.5*fracdelm - 0.5*frdrho - 0.5*(wnew-w)/wnew
        else:
          frdRpl = 0.5*fracdelm - 0.5*frdrho
        Rplume = max(min(Rplume*(1.0 + frdRpl),Rplmin*np.sqrt(2.)),Rplmin)
      #Initial particle size 10 microns
      #Particle growth
      #Collision efficiency depends on phase, and so does terminal velocity due to density
difference
      wjs = np.zeros(len(fs))
      Eij = np.zeros((len(fs),nbin,nbin))
      for a in range(len(fs)):
        if Trisenew > Tsfreeze[a]:
          #liquid stable
          for b in range(nbin):
            for c in range(nbin):
              Eij[a,b,c] = Ssliquid[a]
          wjs[a] = uzQQ*np.sqrt(Vmrise*rhosliquid[a]/muecurr)
        else:
          #solid stable, move effective bins to account for density difference
          for b in range(nbin):
            for c in range(nbin):
              Eij[a,b,c] = Sssolid[a]*((rhoratios[a])**2)
          wjs[a] = uzQQ*np.sqrt(Vmrise*rhossolid[a]*rhoratios[a]/muecurr)
      #particle terminal velocities
      for a in range(len(fs)):
        for b in range(nbin):
          for c in range(nbin):
            wiprea = wjs[a]*np.sqrt(binbounds[max(b,c)+1])
            wjprea = wjs[a]*np.sqrt(binbounds[min(b,c)])
            vrel[a,b,c] = abs(wiprea-wjprea)
      #Vertical distance covered by the pressure step, and therefrom time spent doing
particle growth
      verticalrise = Pstep*Vmfall/(mu_dry*g)
```

```
      if wnew != 0:
         timefly = verticalrise/wnew
      else:
         timefly = 0
      stepsfly = int(np.ceil(timefly/delt))
      lsnewinit = ls + fs_cond
      ltni = np.sum(lsnewinit)
      if wnew<=0 and ltni<=0:
         #No plume, no cloud
         lsnew = np.zeros(len(ls))
         toggleplume = 0
         cloudN = np.zeros([len(mus),nbin])
         cloudM = np.zeros([len(mus),nbin])
         precipN = np.zeros([len(mus),nbin])
         precipM = np.zeros([len(mus),nbin])
         if np.array_equal(togglecondense,flgmhn):
            #Below cloudbase
            #Set Vmrise equal to Vmfall
            Vmrisenew = 1.*Vmfallnew
            Trisenew = T_eos(Pnew,Vmrisenew)
      elif wnew>0 and ltni<=0:
         #Plume, but no cloud
         lsnew = np.zeros(len(ls))
         cloudN = np.zeros([len(mus),nbin])
         cloudM = np.zeros([len(mus),nbin])
         if toggleplume == 0:
            #start plume anew
            toggleplume = 1
         sizescrit = np.zeros(len(mus))
         binssed = np.zeros(len(mus))
         for a in range(len(mus)):
            sizescrit[a] = min((wnew/wjs[a])**2,binbounds[-1])
            binssed[a] =
max(int(np.floor((np.log((sizescrit[a]/binbounds[0])))/np.log(np.sqrt(2.0)))),0) #smallest bin
that will precipitate
         precipN = np.zeros([len(mus),nbin])
         precipM = np.zeros([len(mus),nbin])
      elif wnew<=0 and ltni>0:
         #No plume, but cloud
         toggleplume = 0
         lsnew = np.zeros(len(ls))
         cloudN = np.zeros([len(mus),nbin])
         cloudM = np.zeros([len(mus),nbin])
         precipN = np.zeros([len(mus),nbin])
         precipM = np.zeros([len(mus),nbin])
         fcni = np.sum(fs_cond)
         for a in range(len(mus)):
            if lsnewinit[a] > 0:
               #species a condenses
               n0temp = (lsnewinit[a]/(1.0-ltni))*(muenew/(Vmrisenew*rhosliquid[a]*((np.pi/
3.0)*((binbounds[1]**4) - (binbounds[0]**4)))))
               precipN[a,0] = (binbounds[1]-binbounds[0])*n0temp
               precipM[a,0] = 0.25*n0temp*((binbounds[1]**4) -
(binbounds[0]**4))*(4*np.pi*rhosliquid[a]/3.0)
               if togglecondense[a] == 0:
```

```
                togglecondense[a] = 1
                Pscondens[a] = Pnew+0
                print('Cloud base for species',a,'is',Pscondens[a]/100000.,'bars')
        elif wnew>0 and ltni>0:
          #Both plume and cloud
          lsnew = np.zeros(len(ls))
          for a in range(len(mus)):
            if lsnewinit[a]>0 and togglecondense[a]==0:
              togglecondense[a] = 1
              Pscondens[a] = Pnew+0
              print('Cloud base for species',a,'is',Pscondens[a]/100000.,'bars')
          if toggleplume == 0:
            #start plume anew
            toggleplume = 1
            cloudN = np.zeros([len(mus),nbin])
            cloudM = np.zeros([len(mus),nbin])
            sizescrit = np.zeros(len(mus))
            fcni = np.sum(fs_cond)
            for a in range(len(mus)):
              if lsnewinit[a] > 0:
                #species a condenses
                n0temp = (lsnewinit[a]/(1.0-ltni))*(muenew/(Vmrisenew*rhosliquid[a]*((np.pi/
3.0)*((binbounds[1]**4) - (binbounds[0]**4)))))
                cloudN[a,0] = (binbounds[1]-binbounds[0])*n0temp
                cloudM[a,0] = 0.25*n0temp*((binbounds[1]**4) -
(binbounds[0]**4))*(4*np.pi*rhosliquid[a]/3.0)
            for a in range(len(mus)):
              sizescrit[a] = min((wnew/wjs[a])**2,binbounds[-1])
            timme = 0.0
            #particle growth
            for q in range(stepsfly):
              cloudN, cloudM = stepgrow6(cloudN,cloudM,binbounds,rhosliquid,Eij,vrel,delt)
              timme = timme + delt
          else:
            #continue plume
            sizescrit = np.zeros(len(mus))
            fcni = np.sum(fs_cond)
            for a in range(len(mus)):
              if lsnewinit[a] > 0:
                #species a condenses
                n0temp =
(fs_cond[a]/(1.0-fcni))*(muenew/(Vmrisenew*rhosliquid[a]*((np.pi/3.0)*((binbounds[1]**4) -
(binbounds[0]**4)))))
                cloudN[a,0] = cloudN[a,0] + (binbounds[1]-binbounds[0])*n0temp
                cloudM[a,0] = cloudM[a,0] + 0.25*n0temp*((binbounds[1]**4) -
(binbounds[0]**4))*(4*np.pi*rhosliquid[a]/3.0)
            for a in range(len(mus)):
              sizescrit[a] = min((wnew/wjs[a])**2,binbounds[-1])
            timme = 0.0
            #particle growth
            for q in range(stepsfly):
              cloudN, cloudM = stepgrow6(cloudN,cloudM,binbounds,rhosliquid,Eij,vrel,delt)
              timme = timme + delt
          binssed = np.zeros(len(mus))
          for a in range(len(mus)):
```

```
            binssed[a] =
max(int(np.floor((np.log((sizescrit[a]/binbounds[0])))/np.log(np.sqrt(2.0)))),0) #smallest bin
that will precipitate
          precipN = np.zeros([len(mus),nbin])
          precipM = np.zeros([len(mus),nbin])
          for a in range(len(mus)):
             if lsnewinit[a] > 0:
                Mpout = 0.0
                Mpin = 0.0
                for b in range(int(binssed[a]),nbin,1):
                   Mpout = Mpout + cloudM[a,b]
                   precipM[a,b] = 1*cloudM[a,b]
                   precipN[a,b] = 1*cloudN[a,b]
                   cloudM[a,b] = 0.
                   cloudN[a,b] = 0.
                for b in range(0,int(binssed[a]),1):
                   Mpin = Mpin + cloudM[a,b]
                lsnew[a] = lsnewinit[a]*Mpin/(Mpout+Mpin)
       P = Pnew + 0.0
       Vmrise = Vmrisenew + 0.0
       Vmfall = Vmfallnew + 0.0
       Trise = T_eos(P,Vmrise)
       Tfall = T_eos(P,Vmfall)
       fs = 1*fsnew
       ls = 1*lsnew
       w = wnew + 0.0
       wjslist.append(wjs)
       Nscloudlist.append(cloudN)
       Mscloudlist.append(cloudM)
       NPrecipitations.append(precipN)
       MPrecipitations.append(precipM)
    #At top of atmosphere, precipitate anything left
    precipN = np.zeros([len(mus),nbin])
    precipM = np.zeros([len(mus),nbin])
    for a in range(len(mus)):
       for b in range(nbin):
          precipN[a,b] = 1*cloudN[a,b]
          precipM[a,b] = 1*cloudM[a,b]
    NPrecipitations[-1] = NPrecipitations[-1] + cloudN
    MPrecipitations[-1] = MPrecipitations[-1] + cloudM
    NQstep = int((stepmax+10)/NQfactor - 10)
    print('beginning evaporation')
    #Virga: calculate evaporation of precipitation
    #Move from top of atmosphere downwards
    #Pcondens is the cloud base
    #Form arrays
    NPrecloc = []
    MPrecloc = []
    ibbtop = np.zeros([len(mus),nbin])
    for cw in range(NQstep):
       NPrecloc.append(np.zeros([len(mus),nbin]))
       MPrecloc.append(np.zeros([len(mus),nbin]))
    for a in range(len(mus)):
       #Run evaporation for species a
       NVstep = int(np.ceil(((Pbase*(10**5))-Pscondens[a])/(NQfactor*Pstep)))
```

```
precipNcb = np.zeros(nbin)
precipMcb = np.zeros(nbin)
precipCcb = np.zeros(nbin)
icb = int((NVstep-1)*NQfactor)
Tcb = Tempsrise[icb]
Vmcb = Volsrise[icb]
Pcb = Pressures[icb]
fcb = fsrise[icb].copy()
mucb = mu_avg(mu_dry,mus,fcb)
wjs = wjslist[icb].copy()
for b in range(nbin):
   blow = binbounds[b]
   bhigh = binbounds[b+1]
   pCN = bhigh-blow
   pCD = 2*wjs[a]*(bhigh*np.sqrt(bhigh) - blow*np.sqrt(blow))
   precipCcb[b] = abs(3.*pCN/pCD)
for k in range(icb,stepmax):
   aNp = NPrecipitations[k]
   aMp = MPrecipitations[k]
   for b in range(nbin):
      precipNcb[b] = precipNcb[b] + aNp[a,b]*Velocities[k]*precipCcb[b]
      precipMcb[b] = precipMcb[b] + aMp[a,b]*Velocities[k]*precipCcb[b]
ibwar = int(NVstep-1)
for b in range(nbin):
   NPrecloc[ibwar][a,b] = precipNcb[b]
   MPrecloc[ibwar][a,b] = precipMcb[b]
for ah in range(NQstep-NVstep):
   ib = int(NVstep+ah)
   i = int((NVstep+ah)*NQfactor)
   cloudN = Nscloudlist[i].copy()
   cloudM = Mscloudlist[i].copy()
   P = Pressures[i]
   Vm = Volsrise[i]
   T = Tempsrise[i]
   w = Velocities[i]
   fs = fsrise[i].copy()
   wjs = wjslist[i].copy()
   mu = mu_avg(mu_dry,mus,fs)
   velpart = np.zeros(nbin)
   for b in range(nbin):
      velpart[b] = w - wjs[a]*np.sqrt(r0s[b])
      if ibbtop[a,b] == 0 and velpart[b] > 0:
         ibbtop[a,b] = ib
   precipNV = np.zeros(nbin)
   precipMV = np.zeros(nbin)
   precipCV = np.zeros(nbin)
   for b in range(nbin):
      rD = (w/wjs[a])**2
      if rD >= binbounds[b+1]:
         precipCV[b] = 0.
      else:
         blow = max(binbounds[b],rD)
         bhigh = blow*np.sqrt(2.)
         pCN = binbounds[b+1] - blow
         pCD = blow*(3*w - 2*wjs[a]*np.sqrt(blow)) - bhigh*(3*w - 2*wjs[a]*np.sqrt(bhigh))
```

```
            precipCV[b] = abs(3.*pCN/pCD)
        for k in range(i,stepmax):
            aNp = NPrecipitations[k]
            aMp = MPrecipitations[k]
            for b in range(nbin):
                precipNV[b] = precipNV[b] + aNp[a,b]*Velocities[k]*precipCV[b]
                precipMV[b] = precipMV[b] + aMp[a,b]*Velocities[k]*precipCV[b]
        for b in range(nbin):
            NPrecloc[ib][a,b] = cloudN[a,b] + precipNV[b]
            MPrecloc[ib][a,b] = cloudM[a,b] + precipMV[b]
    for ag in range(NVstep-1):
        #Set variables
        ib = int(NVstep-ag-2)
        i = int((NVstep-ag-2)*NQfactor)
        cloudN = Nscloudlist[i].copy()
        cloudM = Mscloudlist[i].copy()
        P = Pressures[i]
        Vm = Volsrise[i]
        Vmfall = Volsfall[i]
        T = Tempsrise[i]
        w = Velocities[i]
        fs = fsrise[i].copy()
        mu = mu_avg(mu_dry,mus,fs)
        wjs = wjslist[i].copy()
        #Flight times
        verticalrise = NQfactor*Pstep*Vmfall/(mu_dry*g)
        flighttimes = np.zeros(nbin)
        velpart = np.zeros(nbin)
        for b in range(nbin):
            flighttimes[b] = verticalrise/(wjs[a]*np.sqrt(r0s[b]))
            velpart[b] = wjs[a]*np.sqrt(r0s[b])
        #Evaporation of precipitation
        #starting conditions
        Npre = NPrecloc[ib+1][a,:].copy()
        Mpre = MPrecloc[ib+1][a,:].copy()
        n0s, slopes = n0sslopes(binbounds,Npre,Mpre,rhosliquid[a])
        Nnew = np.zeros(nbin)
        Mnew = np.zeros(nbin)
        #Calculate evaporation rates
        if np.array_equal(np.zeros(len(Npre)),Npre):
            #No loop
            for b in range(nbin):
                Nnew[b] = 1*Npre[b]
                Mnew[b] = 1*Mpre[b]
            #No condensate
        elif ag > 10:
            #far from cloudbase
            #Calculate mu_other (average molar mass of everything but species a)
            fs_other = fs.copy()
            fs_other[a] = 0
            fs_other = fs_other*(1./(1. - fs[a]))
            mu_other = mu_avg(mu_dry,mus,fs_other)
            for bbq in range(NSfactor):
                #Calculate Nnew and Mnew
                Nnew = np.zeros(nbin)
```

```
            Mnew = np.zeros(nbin)
            edNdt, edMdt =
Evapor6(n0s,slopes,binbounds,rhosliquid[a],velpart,P,Vm,fs[a],mu_other,mus[a],Ptweak=Pst
ep)
            for b in range(nbin):
               Nnew[b] = max(Npre[b] + edNdt[b]*flighttimes[b]/NSfactor,0.)
               Mnew[b] = max(Mpre[b] + edMdt[b]*flighttimes[b]/NSfactor,0.)
               if Nnew[b] <= 0:
                  Mnew[b] = 0
               if Mnew[b] <= 0:
                  Nnew[b] = 0
            #Set Npre and Mpre to new values
            for b in range(nbin):
               Npre[b] = 1*Nnew[b]
               Mpre[b] = 1*Mnew[b]
            #Calculate n0s and slopes
            n0s, slopes = n0sslopes(binbounds,Npre,Mpre,rhosliquid[a])
            Npre, Mpre = NsMs(binbounds,n0s,slopes,rhosliquid[a])
         else:
            #Full loop over NEfactor
            #Calculate mu_other (average molar mass of everything but species a)
            fs_other = fs.copy()
            fs_other[a] = 0
            fs_other = fs_other*(1./(1. - fs[a]))
            mu_other = mu_avg(mu_dry,mus,fs_other)
            for bbq in range(NEfactor):
               #Calculate Nnew and Mnew
               Nnew = np.zeros(nbin)
               Mnew = np.zeros(nbin)
               edNdt, edMdt =
Evapor6(n0s,slopes,binbounds,rhosliquid[a],velpart,P,Vm,fs[a],mu_other,mus[a],Ptweak=Pst
ep)
               for b in range(nbin):
                  Nnew[b] = max(Npre[b] + edNdt[b]*flighttimes[b]/NEfactor,0.)
                  Mnew[b] = max(Mpre[b] + edMdt[b]*flighttimes[b]/NEfactor,0.)
                  if Nnew[b] <= 0:
                     Mnew[b] = 0
                  if Mnew[b] <= 0:
                     Nnew[b] = 0
               #Set Npre and Mpre to new values
               for b in range(nbin):
                  Npre[b] = 1*Nnew[b]
                  Mpre[b] = 1*Mnew[b]
               #Calculate n0s and slopes
               n0s, slopes = n0sslopes(binbounds,Npre,Mpre,rhosliquid[a])
               Npre, Mpre = NsMs(binbounds,n0s,slopes,rhosliquid[a])
         for b in range(nbin):
            NPrecloc[ib][a,b] = Nnew[b]
            MPrecloc[ib][a,b] = Mnew[b]
   #With a particle distribution completed, begin electrical model
   #Larger step size due to computational expense: factor of NQfactor
   print('beginning charging')
   J1ss = np.zeros(NQstep)
   J2ss = np.zeros(NQstep)
   tcrits = np.zeros(NQstep)
```

```
Rflash = np.zeros(NQstep-1)
Qsfall = np.zeros([len(mus),nbin])
enbtop = np.zeros([len(mus),nbin])
for ivb in range(NQstep):
   ib = int(NQstep-ivb-1)
   i = ib*NQfactor
   P = Pressures[i]
   Vm = Volsrise[i]
   Vmf = Volsfall[i]
   T = Tempsrise[i]
   w = Velocities[i]
   fs = fsrise[i]
   mu = mu_avg(mu_dry,mus,fs)
   wjs = wjslist[i]
   Ns = NPrecloc[ib]
   Ms = MPrecloc[ib]
   velpart = np.zeros([len(mus),nbin])
   for a in range(len(mus)):
      for b in range(nbin):
         velpart[a,b] = w - wjs[a]*np.sqrt(r0s[b])
   verticalrise = NQfactor*Pstep*Vmf/(mu_dry*g)
   flighttimesa = np.zeros(nbin)
   Qcoeffs = np.zeros(len(mus))
   rhosadj = np.zeros(len(mus))
   for b in range(nbin):
      flighttimesa[b] = verticalrise/(wjs[a]*np.sqrt(r0s[b]))
   for a in range(len(mus)):
      if T > Tsfreeze[a]:
         Qcoeffs[a] = 1.0 - Ssliquid[a]
         rhosadj[a] = 1.0
      else:
         Qcoeffs[a] = 1.0 - Sssolid[a]
         rhosadj[a] = rhoratios[a]
   kara = dQdt6(Ns,binbounds,velpart,mus,rhosadj,Qcoeffs)
   for a in range(len(mus)):
      #Above or below the cloudbase for species a?
      NVstep = int(np.ceil(((Pbase*(10**5))-Pscondens[a])/(NQfactor*Pstep)))
      if ib > NVstep-2:
         #Above cloudbase, E~t^2
         #Calculate field growth at 1 second
         qara = dEdt6(Ns,binbounds,velpart,kara,mus)
         J1ss[ib] = J1ss[ib] + (8.854*(10**-12))*qara
         #Add to falling charges if necessary
         for b in range(nbin):
            if (ib+1)==ibbtop[a,b]:
               enbtop[a,b]=1
            if enbtop[a,b] > 0:
               delQ = flighttimesa[b]*kara[a,b]
               Qsfall[a,b] = Qsfall[a,b] + delQ
      elif ib <= NVstep-2:
         #Below cloudbase, E~t
         for b in range(nbin):
            delQ = flighttimesa[b]*kara[a,b]
            Qsfall[a,b] = Qsfall[a,b] + delQ
         qara = dEdt6(Ns,binbounds,velpart,Qsfall,mus)
```

```
            J2ss[ib] = J2ss[ib] + (8.854*(10**-12))*qara #current density
    for ivb in range(NQstep-1):
        ib = int(NQstep-ivb-2)
        i = ib*NQfactor
        P = Pressures[i]
        T = Tempsrise[i]
        Vmf = Volsfall[i]
        Emax = 5.0*P
        J1t = J1ss[ib]/(2*8.854*(10**-12))
        J2t = J2ss[ib]/(8.854*(10**-12))
        #Power accumulation: E = J1/eps0 t^2 + J2/eps0 t
        #Energy per volume = eps0/2 * E^2
        if J1t==0 and J2t==0:
            PPV = 0
        elif J1t == 0:
            #Purely linear charging
            tc = abs(Emax/J2t)
            PPV = 0.5*tc*(8.854*(10**-12))*(J2t**2)
        else:
            #First, calculate time to flash
            #Are we going positive or negative?
            invtc = 0
            if (J2t**2 + 4*J1t*Emax) >= 0:
                invtct = (2.*J1t)/(np.sqrt(J2t**2 + 4*J1t*Emax) - J2t)
                if invtct > invtc:
                    invtc = invtct+0
                invtct = (2.*J1t)/(-np.sqrt(J2t**2 + 4*J1t*Emax) - J2t)
                if invtct > invtc:
                    invtc = invtct+0
            if (J2t**2 - 4*J1t*Emax) >= 0:
                invtct = (2.*J1t)/(np.sqrt(J2t**2 - 4*J1t*Emax) - J2t)
                if invtct > invtc:
                    invtc = invtct+0
                invtct = (2.*J1t)/(-np.sqrt(J2t**2 - 4*J1t*Emax) - J2t)
                if invtct > invtc:
                    invtc = invtct+0
            if invtc == 0:
                PPV = 0
            else:
                tc = 1./invtc
                PPV = 0.5*tc*(8.854*(10**-12))*((J2t + J1t*tc)**2)
        verticalrise = NQfactor*Pstep*Vmf/(mu_dry*g)
        Rflash[ib] = abs(verticalrise*PPV)
    Plim = np.zeros(NQstep-1)
    for a in range(NQstep-1):
        Plim[a] = Pressures[NQfactor*a]
    return Pressures, Velocities, Tempsrise, Tempsfall, Tempsdiff, Radii, Plim, Rflash, fsrise,
lsrise, Volsrise, Volsfall, Nprecloc

#Example model setup
def main():
    stdout_fileno = sys.stdout
    ndesc = 'Jupiter Temp 330K 3x sulfur 3x nitrogen b' #parameters
    sys.stdout = open(os.path.join('6E', ndesc+' flash rates.txt'), 'w')
    print('Parameters;',ndesc)
```

```
   T_base=330.
   radiu=5000.
   Nti = 3.0 #Nitrogen enrichment relative to solar
   Sfu = 3.0 #Sulfur enrichment relative to solar
   Nti = Nti - 0.2*Sfu
   for Oyx in [3.0,1.0,0.3]:
      #Oyx is oxygen enrichment relative to solar
      ncurr = ndesc+' water '+str(Oyx)
      #Test dissolved ammonia weight-percent: 0%, 10%, 20%, eutectic (35%)
      #That is, without supercooling: 273K, 263K, 243K, 173K
      #Add 10K supercooling, though.
      R10W =
iayer(T_base,10.0,np.asarray([Oyx*.00625]),np.asarray([.018015]),radiu,np.asarray([263.1]),n
p.asarray([0.8]),np.asarray([0.0]),np.asarray([1000.]),np.asarray([400.]),np.asarray([2257000.0
]))
      R10 =
iayer(T_base,10.0,np.asarray([Oyx*.00625,Nti*.00083333,Sfu*.0005]),np.asarray([.018015,.0
17031,.051107]),radiu,np.asarray([263.1,185.4,261.0]),np.asarray([0.8,0.8,0.8]),np.asarray([0.
0,0.0,0.0]),np.asarray([1000.,680.,1200.]),np.asarray([400.,350.,520.]),np.asarray([2257000.0,
1370000.,2509000.0]))
      P = R10[0]
      Pbar = R10[0].copy()
      Pl = R10[6]
      Plbar = R10[6].copy()
      for a in range(len(P)):
         Pbar[a] = P[a]/(10.**5)
      for a in range(len(Pl)):
         Plbar[a] = Pl[a]/(10.**5)
      #Calculate R20, R40, R110
      arhtum = 4
      if 9*Nti <= 7.5*Oyx:
         #Insufficient nitrogen for 10% ammonia
         #Put all of it in
         arhtum = 1
         aqtotal = Oyx*.00625 + Nti*.00083333
         famm = Nti*.00083333/aqtotal
         fJamm = (Nti/.017031)/((Nti/.017031)+(Oyx*7.5/.018015))
         mu_aq = .017031*fJamm + .018015*(1-fJamm)
         L_aq = 1370000.*famm + 2257000.0*(1-famm)
         defrpt = 263.1 - (12.*Nti/Oyx)
         R20 =
iayer(T_base,10.0,np.asarray([aqtotal,Sfu*.0005]),np.asarray([mu_aq,.051107]),radiu,np.asar
ray([defrpt,261.0]),np.asarray([0.8,0.8]),np.asarray([0.0,0.0]),np.asarray([1000.,1200.]),np.asa
rray([400.,520.]),np.asarray([L_aq,2509000.0]))
         R40 = R20 + tuple()
         R110 = R20 + tuple()
      else:
         aqtotal = Oyx*.00625*(10/9.)
         NH3eff = Nti*.00083333 - Oyx*.00625/9.
         famm = 0.1
         fJamm = (.1/.017031)/((.1/.017031)+(.9/.018015))
         mu_aq = .017031*fJamm + .018015*(1-fJamm)
         L_aq = 1370000.*famm + 2257000.0*(1-famm)
         R20 =
iayer(T_base,10.0,np.asarray([aqtotal,NH3eff,Sfu*.0005]),np.asarray([mu_aq,.017031,.05110
```

```
7]),radiu,np.asarray([253.1,185.4,261.0]),np.asarray([0.8,0.8,0.8]),np.asarray([0.0,0.0,0.0]),np.
asarray([1000.,680.,1200.]),np.asarray([400.,350.,520.]),np.asarray([L_aq,1370000.,2509000.0
]))
        if 4*Nti <= 7.5*Oyx:
           #Insufficient nitrogen for 20% ammonia
           #Put all of it in
           arhtum = 2
           aqtotal = Oyx*.00625 + Nti*.00083333
           famm = Nti*.00083333/aqtotal
           fJamm = (Nti/.017031)/((Nti/.017031)+(Oyx*7.5/.018015))
           mu_aq = .017031*fJamm + .018015*(1-fJamm)
           L_aq = 1370000.*famm + 2257000.0*(1-famm)
           defrpt = 253.1 - 0.8*((24*Nti/Oyx)-20.)
           R40 =
iayer(T_base,10.0,np.asarray([aqtotal,Sfu*.0005]),np.asarray([mu_aq,.051107]),radiu,np.asar
ray([defrpt,261.0]),np.asarray([0.8,0.8]),np.asarray([0.0,0.0]),np.asarray([1000.,1200.]),np.asa
rray([400.,520.]),np.asarray([L_aq,2509000.0]))
           R110 = R40 + tuple()
        else:
           aqtotal = Oyx*.00625*(5/4.)
           NH3eff = Nti*.00083333 - Oyx*.00625/4.
           famm = 0.2
           fJamm = (.2/.017031)/((.2/.017031)+(.8/.018015))
           mu_aq = .017031*fJamm + .018015*(1-fJamm)
           L_aq = 1370000.*famm + 2257000.0*(1-famm)
           R40 =
iayer(T_base,10.0,np.asarray([aqtotal,NH3eff,Sfu*.0005]),np.asarray([mu_aq,.017031,.05110
7]),radiu,np.asarray([233.1,185.4,261.0]),np.asarray([0.8,0.8,0.8]),np.asarray([0.0,0.0,0.0]),np.
asarray([1000.,680.,1200.]),np.asarray([400.,350.,520.]),np.asarray([L_aq,1370000.,2509000.0
]))
           if (13/7.)*Nti <= 7.5*Oyx:
              #Insufficient nitrogen for 35% ammonia
              #Put all of it in
              arhtum = 3
              aqtotal = Oyx*.00625 + Nti*.00083333
              famm = Nti*.00083333/aqtotal
              fJamm = (Nti/.017031)/((Nti/.017031)+(Oyx*7.5/.018015))
              mu_aq = .017031*fJamm + .018015*(1-fJamm)
              L_aq = 1370000.*famm + 2257000.0*(1-famm)
              defrpt = 233.1 - ((104*Nti/Oyx)-195.)*14/45.
              R110 =
iayer(T_base,10.0,np.asarray([aqtotal,Sfu*.0005]),np.asarray([mu_aq,.051107]),radiu,np.asar
ray([defrpt,261.0]),np.asarray([0.8,0.8]),np.asarray([0.0,0.0]),np.asarray([1000.,1200.]),np.asa
rray([400.,520.]),np.asarray([L_aq,2509000.0,2509000.0]))
           else:
              aqtotal = Oyx*.00625*(20/13.)
              NH3eff = Nti*.00083333 - Oyx*.00625*7/13.
              famm = 0.35
              fJamm = (.35/.017031)/((.35/.017031)+(.65/.018015))
              mu_aq = .017031*fJamm + .018015*(1-fJamm)
              L_aq = 1370000.*famm + 2257000.0*(1-famm)
              R110 =
iayer(T_base,10.0,np.asarray([aqtotal,NH3eff,Sfu*.0005]),np.asarray([mu_aq,.017031,.05110
7]),radiu,np.asarray([163.1,185.4,261.0]),np.asarray([0.8,0.8,0.8]),np.asarray([0.0,0.0,0.0]),np.
```

asarray([1000.,680.,1200.]),np.asarray([400.,350.,520.]),np.asarray([L_aq,1370000.,2509000.0
]))

```
    v10 = R10[1]
    v20 = R20[1]
    v40 = R40[1]
    v110 = R110[1]
    v10W = R10W[1]
    Tr10 = R10[2]
    Tr20 = R20[2]
    Tr40 = R40[2]
    Tr110 = R110[2]
    Tr10W = R10W[2]
    Tf10 = R10[3]
    Tf20 = R20[3]
    Tf40 = R40[3]
    Tf110 = R110[3]
    Tf10W = R10W[3]
    Td10 = R10[4]
    Td20 = R20[4]
    Td40 = R40[4]
    Td110 = R110[4]
    Td10W = R10W[4]
    Rf10 = R10[7]
    Rf20 = R20[7]
    Rf40 = R40[7]
    Rf110 = R110[7]
    Rf10W = R10W[7]
    rpl10 = R10[5]
    rpl20 = R20[5]
    rpl40 = R40[5]
    rpl110 = R110[5]
    rpl10W = R10W[5]
    fst10 = R10[8]
    fst20 = R20[8]
    fst40 = R40[8]
    fst110 = R110[8]
    fst10W = R10W[8]
    lst10 = R10[9]
    lst20 = R20[9]
    lst40 = R40[9]
    lst110 = R110[9]
    lst10W = R10W[9]
    fsw10 = R10[5].copy()
    fsw20 = R10[5].copy()
    fsw40 = R10[5].copy()
    fsw110 = R10[5].copy()
    fsw10W = R10[5].copy()
    fsa10 = R10[5].copy()
    fsa20 = R10[5].copy()
    fsa40 = R10[5].copy()
    fsa110 = R10[5].copy()
    fss10 = R10[5].copy()
    fss20 = R10[5].copy()
    fss40 = R10[5].copy()
```

```
fss110 = R10[5].copy()
lsw10 = R10[5].copy()
lsw20 = R10[5].copy()
lsw40 = R10[5].copy()
lsw110 = R10[5].copy()
lsw10W = R10[5].copy()
lsa10 = R10[5].copy()
lsa20 = R10[5].copy()
lsa40 = R10[5].copy()
lsa110 = R10[5].copy()
lss10 = R10[5].copy()
lss20 = R10[5].copy()
lss40 = R10[5].copy()
lss110 = R10[5].copy()
for za in range(len(P)):
    fsw10[za] = fst10[za][0]
    lsw10[za] = lst10[za][0]
    fsa10[za] = fst10[za][1]
    lsa10[za] = lst10[za][1]
    fss10[za] = fst10[za][2]
    lss10[za] = lst10[za][2]
    fsw20[za] = fst20[za][0]
    lsw20[za] = lst20[za][0]
    if arhtum>1.5:
        fsa20[za] = fst20[za][1]
        lsa20[za] = lst20[za][1]
        fss20[za] = fst20[za][2]
        lss20[za] = lst20[za][2]
        if arhtum>2.5:
            fsa40[za] = fst40[za][1]
            lsa40[za] = lst40[za][1]
            fss40[za] = fst40[za][2]
            lss40[za] = lst40[za][2]
            if arhtum>3.5:
                fsa110[za] = fst110[za][1]
                lsa110[za] = lst110[za][1]
                fss110[za] = fst110[za][2]
                lss110[za] = lst110[za][2]
    if arhtum <= 3.5:
        fss110[za] = fst110[za][1]
        lss110[za] = lst110[za][1]
        if arhtum <= 2.5:
            fss40[za] = fst40[za][1]
            lss40[za] = lst40[za][1]
            if arhtum <= 1.5:
                fss20[za] = fst20[za][1]
                lss20[za] = lst20[za][1]
    fsw40[za] = fst40[za][0]
    lsw40[za] = lst40[za][0]
    fsw110[za] = fst110[za][0]
    lsw110[za] = lst110[za][0]
    fsw10W[za] = fst10W[za][0]
    lsw10W[za] = lst10W[za][0]

plt.figure()
```

```
plt.plot(Pbar,v10,'g-',label='Pure water and ammonia clouds')
plt.plot(Pbar,v20,'c-',label='Aqueous cloud to 10% weight ammonia')
if arhtum>1.5:
   plt.plot(Pbar,v40,'b-',label='Aqueous cloud to 20% weight ammonia')
   if arhtum>2.5:
      plt.plot(Pbar,v110,'m-',label='Aqueous cloud to 35% weight ammonia')
plt.plot(Pbar,v10W,'k-',label='No ammonia in system')
plt.title(ncurr)
plt.xlabel('Pressure (bar)')
plt.ylabel('Vertical plume velocity (m/s)')
plt.legend()
ncurrn = ncurr+' velocities.png'
plt.savefig(os.path.join('6E', ncurrn))

plt.figure()
plt.plot(Pbar,Tr10,'g-',label='Pure water and ammonia clouds')
plt.plot(Pbar,Tr20,'c-',label='Aqueous cloud to 10% weight ammonia')
if arhtum>1.5:
   plt.plot(Pbar,Tr40,'b-',label='Aqueous cloud to 20% weight ammonia')
   if arhtum>2.5:
      plt.plot(Pbar,Tr110,'m-',label='Aqueous cloud to 35% weight ammonia')
plt.plot(Pbar,Tr10W,'k-',label='No ammonia in system')
plt.title(ncurr)
plt.xlabel('Pressure (bar)')
plt.ylabel('Plume temperature (K)')
plt.legend()
ncurrn = ncurr+' plume temps.png'
plt.savefig(os.path.join('6E', ncurrn))

plt.figure()
plt.plot(Pbar,Tf10,'g-',label='Pure water and ammonia clouds')
plt.plot(Pbar,Tf20,'c-',label='Aqueous cloud to 10% weight ammonia')
if arhtum>1.5:
   plt.plot(Pbar,Tf40,'b-',label='Aqueous cloud to 20% weight ammonia')
   if arhtum>2.5:
      plt.plot(Pbar,Tf110,'m-',label='Aqueous cloud to 35% weight ammonia')
plt.plot(Pbar,Tf10W,'k-',label='No ammonia in system')
plt.title(ncurr)
plt.xlabel('Pressure (bar)')
plt.ylabel('Environment temperature (K)')
plt.legend()
ncurrn = ncurr+' environ temps.png'
plt.savefig(os.path.join('6E', ncurrn))

plt.figure()
plt.plot(Pbar,Td10,'g-',label='Pure water and ammonia clouds')
plt.plot(Pbar,Td20,'c-',label='Aqueous cloud to 10% weight ammonia')
if arhtum>1.5:
   plt.plot(Pbar,Td40,'b-',label='Aqueous cloud to 20% weight ammonia')
   if arhtum>2.5:
      plt.plot(Pbar,Td110,'m-',label='Aqueous cloud to 35% weight ammonia')
plt.plot(Pbar,Td10W,'k-',label='No ammonia in system')
plt.title(ncurr)
plt.xlabel('Pressure (bar)')
plt.ylabel('Plume-environment temperature difference (K)')
```

```
    plt.legend()
    ncurrn = ncurr+' temp diffs.png'
    plt.savefig(os.path.join('6E', ncurrn))

    plt.figure()
    plt.plot(Pbar,rpl10,'g-',label='Pure water and ammonia clouds')
    plt.plot(Pbar,rpl20,'c-',label='Aqueous cloud to 10% weight ammonia')
    if arhtum>1.5:
      plt.plot(Pbar,rpl40,'b-',label='Aqueous cloud to 20% weight ammonia')
      if arhtum>2.5:
        plt.plot(Pbar,rpl110,'m-',label='Aqueous cloud to 35% weight ammonia')
    plt.plot(Pbar,rpl10W,'k-',label='No ammonia in system')
    plt.title(ncurr)
    plt.xlabel('Pressure (bar)')
    plt.ylabel('Plume radius (m)')
    plt.legend()
    ncurrn = ncurr+' radii.png'
    plt.savefig(os.path.join('6E', ncurrn))

    plt.figure()
    plt.plot(Plbar,Rf10,'g-',label='Pure water and ammonia clouds')
    plt.plot(Plbar,Rf20,'c-',label='Aqueous cloud to 10% weight ammonia')
    if arhtum>1.5:
      plt.plot(Plbar,Rf40,'b-',label='Aqueous cloud to 20% weight ammonia')
      if arhtum>2.5:
        plt.plot(Plbar,Rf110,'m-',label='Aqueous cloud to 35% weight ammonia')
    plt.plot(Plbar,Rf10W,'k-',label='No ammonia in system')
    plt.title(ncurr)
    plt.xlabel('Pressure (bar)')
    plt.ylabel('Flash rate')
    plt.legend()
    ncurrn = ncurr+' flash.png'
    plt.savefig(os.path.join('6E', ncurrn))

    plt.figure()
    plt.plot(Pbar,fsw10,'g-',label='Water, pure water and ammonia clouds')
    plt.plot(Pbar,fsw20,'c-',label='Aqueous, aqueous cloud to 10% weight ammonia')
    if arhtum>1.5:
      plt.plot(Pbar,fsw40,'b-',label='Aqueous, aqueous cloud to 20% weight ammonia')
      if arhtum>2.5:
        plt.plot(Pbar,fsw110,'m-',label='Aqueous, aqueous cloud to 35% weight ammonia')
    plt.plot(Pbar,fsw10W,'k-',label='Water, no ammonia in system')
    plt.plot(Pbar,fsa10,'g--',label='Ammonia, pure water and ammonia clouds')
    if arhtum>1.5:
      plt.plot(Pbar,fsa20,'c--',label='Ammonia, aqueous cloud to 10% weight ammonia')
      if arhtum>2.5:
        plt.plot(Pbar,fsa40,'b--',label='Ammonia, aqueous cloud to 20% weight ammonia')
        if arhtum>3.5:
          plt.plot(Pbar,fsa110,'m--',label='Ammonia, aqueous cloud to 35% weight
ammonia')
    plt.plot(Pbar,fss10,'g-.',label='NH4SH, pure water and ammonia clouds')
    plt.plot(Pbar,fss20,'c-.',label='NH4SH, aqueous cloud to 10% weight ammonia')
    if arhtum>1.5:
      plt.plot(Pbar,fss40,'b-.',label='NH4SH, aqueous cloud to 20% weight ammonia')
      if arhtum>2.5:
```

```
            plt.plot(Pbar,fss110,'m-.',label='NH4SH, aqueous cloud to 35% weight ammonia')
        plt.title(ncurr)
        plt.xlabel('Pressure (bar)')
        plt.ylabel('Vapor mass fraction')
        plt.legend()
        ncurrn = ncurr+' vapor.png'
        plt.savefig(os.path.join('6E', ncurrn))

        plt.figure()
        plt.plot(Pbar,lsw10,'g-',label='Pure water and ammonia clouds')
        plt.plot(Pbar,lsw20,'c-',label='Aqueous cloud to 10% weight ammonia')
        if arhtum>1.5:
           plt.plot(Pbar,lsw40,'b-',label='Aqueous cloud to 20% weight ammonia')
           if arhtum>2.5:
              plt.plot(Pbar,lsw110,'m-',label='Aqueous cloud to 35% weight ammonia')
        plt.plot(Pbar,lsw10W,'k-',label='No ammonia in system')
        plt.title(ncurr)
        plt.xlabel('Pressure (bar)')
        plt.ylabel('Aqueous condensate mass fraction')
        plt.legend()
        ncurrn = ncurr+' water cloud.png'
        plt.savefig(os.path.join('6E', ncurrn))

        plt.figure()
        plt.plot(Pbar,lss10,'g-',label='Pure water and ammonia clouds')
        plt.plot(Pbar,lss20,'c-',label='Aqueous cloud to 10% weight ammonia')
        if arhtum>1.5:
           plt.plot(Pbar,lss40,'b-',label='Aqueous cloud to 20% weight ammonia')
           if arhtum>2.5:
              plt.plot(Pbar,lss110,'m-',label='Aqueous cloud to 35% weight ammonia')
        plt.title(ncurr)
        plt.xlabel('Pressure (bar)')
        plt.ylabel('NH4SH condensate mass fraction')
        plt.legend()
        ncurrn = ncurr+' NH4SH cloud.png'
        plt.savefig(os.path.join('6E', ncurrn))

        plt.figure()
        plt.plot(Pbar,lsa10,'g-',label='Pure water and ammonia clouds')
        if arhtum>1.5:
           plt.plot(Pbar,lsa20,'c-',label='Ammonia cloud, aqueous cloud to 10% weight
ammonia')
           if arhtum>2.5:
              plt.plot(Pbar,lsa40,'b-',label='Ammonia cloud, aqueous cloud to 20% weight
ammonia')
              if arhtum>3.5:
                 plt.plot(Pbar,lsa110,'m-',label='Ammonia cloud, aqueous cloud to 35% weight
ammonia')
        plt.title(ncurr)
        plt.xlabel('Pressure (bar)')
        plt.ylabel('Anhydrous ammonia condensate mass fraction')
        plt.legend()
        ncurrn = ncurr+' ammonia cloud.png'
        plt.savefig(os.path.join('6E', ncurrn))
```

```
      print('For water abundance '+str(Oyx)+' solar, flash rates are '+str(np.sum(Rf10))+' watts
per square meter with ammonia clouds and '+str(np.sum(Rf10W))+' watts per square meter
without')
      print('Pentile flash rates without ammonia:')
      print(str(np.sum(Rf10W[800:]))+' '+str(np.sum(Rf10W[600:800]))+'
'+str(np.sum(Rf10W[400:600]))+' '+str(np.sum(Rf10W[200:400]))+'
'+str(np.sum(Rf10W[:200])))
      print('Pentile flash rates with ammonia:')
      print(str(np.sum(Rf10[800:]))+' '+str(np.sum(Rf10[600:800]))+'
'+str(np.sum(Rf10[400:600]))+' '+str(np.sum(Rf10[200:400]))+' '+str(np.sum(Rf10[:200])))
      print('When ammonia is mixed into the water clouds:')
      print('At 10% by weight ammonia, flash rate is '+str(np.sum(Rf20))+' watts per square
meter')
      print('Pentile flash rates')
      print(str(np.sum(Rf20[800:]))+' '+str(np.sum(Rf20[600:800]))+'
'+str(np.sum(Rf20[400:600]))+' '+str(np.sum(Rf20[200:400]))+' '+str(np.sum(Rf20[:200])))
      if arhtum>1.5:
         print('At 20% by weight ammonia, flash rate is '+str(np.sum(Rf40))+' watts per square
meter')
         print('Pentile flash rates')
         print(str(np.sum(Rf40[800:]))+' '+str(np.sum(Rf40[600:800]))+'
'+str(np.sum(Rf40[400:600]))+' '+str(np.sum(Rf40[200:400]))+' '+str(np.sum(Rf40[:200])))
         if arhtum>2.5:
            print('At 35% by weight ammonia, flash rate is '+str(np.sum(Rf110))+' watts per
square meter')
            print('Pentile flash rates')
            print(str(np.sum(Rf110[800:]))+' '+str(np.sum(Rf110[600:800]))+'
'+str(np.sum(Rf110[400:600]))+' '+str(np.sum(Rf110[200:400]))+' '+str(np.sum(Rf110[:200])))
   sys.stdout.close()
   sys.stdout = stdout_fileno


if __name__ == '__main__': main()
```

REFERENCES

Aglyamov, Y.S., Lunine, J., Becker, H.N., Guillot, T., Gibbard, S.G., Atreya, S., Bolton, S.J., Levin, S., Brown, S.T., and Wong, M.H .(2021). Lightning Generation in Moist Convective Clouds and Constraints on the Water Abundance in Jupiter. *JGR Planets* 126, https://doi.org/10.1029/2020JE006504.

Aglyamov, Y.S., Lunine, J., Atreya, S., Guillot, T., Becker, H.N., Levin, S. and Bolton, S.J. (2023a). Giant Planet Lightning in Non-Ideal Gases. Submitted to *The Planetary Science Journal*.

Aglyamov, Yury S. (2023b). Giant-planetary Entraining Plume Electrification. Zenodo. https://doi.org/10.5281/zenodo.4039311

Aplin, K.L., Fischer, G., Nordheim, T.A., Konovalenko, A., Zakharenko, V. and Zarka, P. (2020). Atmospheric Electricity at the Ice Giants. *Space Sci Rev* 216:26, https://doi.org/10.1007/s11214-020-00647-0

Asplund, M., Grevess, N., Sauval, A.J., & Scott, P. (2009). The chemical composition of the Sun. *Annual Review of Astronomy and Astrophysics*, 47, 481-522.

Atreya, S.K., Wong, M.H., Owen, T.C., Mahaffy, P.R., Niemann, H.B., de Pater, I., Drossart, P., & Encrenaz, T. (1999). A comparison of the atmospheres of Jupiter and Saturn: deep atmospheric composition, cloud structure, vertical mixing, and origin. *Planetary and Space Science*, 47, 1243–1262.

Atreya, S.K. and Wong, A. (2005). Coupled Clouds and Chemistry of the Giant Planets— A Case for Multiprobes. *Space Science Reviews* 116, 121-36. https://doi.org/10.1007/s11214-005-1951-5

Atreya, S. K. *et al*. (2019). The Origin and Evolution of Saturn, with Exoplanet Perspective in Saturn in the 21st Century (eds. K. H Baines, et al.), Cambridge University Press. https://doi.org/10.1017/9781316.227220.002.

Atreya, S. K. *et al*. (2020). Deep Atmosphere Composition, Structure, Origin, and Exploration, with Particular Focus on Critical in situ Science at the Icy Giants, *Space Sci. Rev.* 216:18, Issue 1, 31pp. https://doi.org/10.1007/s11214-020-0640-8.

Becker, H.N., Alexander, J.W., Atreya, S.K., Bolton, S.J., Brennan, M.J., Brown, S.T., *et al*. (2020). Small lightning flashes from shallow electrical storms on Jupiter. *Nature*, 584, 55-58. https://doi.org/10.1038/s41586-020-2532-1.

Bellotti, A., Steffes, P.G., and Chinsomboom, G. (2016). Laboratory measurements of the 5–20 cm wavelength opacity of ammonia, water vapor, and methane under simulated conditions for the deep jovian atmosphere. *Icarus*, 280, 255-67. http://dx.doi.org/10.1016/j.icarus.2016.07.013

Bjoraker, G., Wong, M.H., de Pater, I., Hewagama, T., Adamkovics, M., and Orton, G.S. (2018). The gas composition and deep cloud structure of Jupiter's Great Red Spot. *Astron. J.*, 156:101 (15 pp).

Borucki, W.J., Bar-Nun, A., Scarf, F.L., Cook II, A.F., & Hunt, G.E. (1982). Lightning activity on Jupiter. *Icarus*, 52, 492-502.

Borucki, W.J., & Williams, M.A. (1986). Lightning in the Jovian water cloud. *Journal of Geophysical Research – Atmospheres* 91, 9893-903 (1986).

Brown, S., Janssen, M., Adumitroale, V., Atreya, S.K., Bolton, S., Gulkis, S., *et al*. (2018). Prevalent lightning sferics at 600 megahertz near Jupiter's poles. *Nature*, 558, 87-90.

Burns, J.A., Showalter, M.R., Cuzzi, J.N., and Durisen, R.H. (1983). Saturn's electrostatic discharges: Could lightning be the cause? *Icarus* 54, 280-95.

Chase, M.W. Jr. (1998). NIST-JANAF Thermochemical Tables, 4th edition. *Journal of Physical and Chemical Reference Data Monographs*, American Institute of Physics, New York.

Christian, H.J., Blakeslee, R.J., Boccippio, D.J., Boeck, W.L., Buechler, D.E., Driscoll, K.T., *et al*. (2003). Global frequency and distribution of lightning as observed from space by the Optical Transient Detector. *J. Geophys. Res. Atmospheres*, 108 (D1), 4005, doi:10.1029/2002JD002347

Cook II, A.F., Duxbury, T.C. & Hunt, E.G (1979). First results on Jovian lightning. *Nature*, 280,794.

Dye, J.E. & Bansemer, A. (2019). Electrification in mesoscale updrafts of deep stratiform and anvil clouds in Florida. *Journal of Geophysical Research: Atmospheres*, 124, 1021-1049.

Dyudina, U.A., Ingersoll, A.P., Vasavada, A.R., Ewald, S.P., and Galileo SSI Team (2002). Monte Carlo Radiative Transfer Modeling of Lightning Observed in Galileo Images of Jupiter. *Icarus*, 160, 336–349.

Dyudina, U.A., Del Genio, A.D., Ingersoll, A.P., Porco, C.C., West, R.A., Vasavada, A.R., and Barbara, J.M. (2004). Lightning on Jupiter observed in the Hα line by the Cassini imaging science subsystem. *Icarus*, 172, 24-36.

Dyudina, U.A., Ingersoll, A.P., Ewald, S.P., Porco, C.C., Fischer, G., Kurth, W.S., and West, R.A. (2010). Detection of visible lightning on Saturn. *Geophysical Research Letters* 37, L09205.

Dyudina, U.A., Ingersoll, A.P., Ewald, S.P., Porco, C.C., Fischer, G. and Yair, Y. (2013). Saturn's visible lightning, its radio emissions, and the structure of the 2009–2011 lightning storms. *Icarus* 226, 1020-37.
http://dx.doi.org/10.1016/j.icarus.2013.07.013

Encrenaz, T. *et al*. (1998). ISO observations of Uranus: The stratospheric distribution of C_2H_2 and the eddy diffusion coefficient. *A&A* 333, L43.

Fierro, A.O., Mansell, E.R., MacGorman, D.R. and Ziegler, C.L. (2013). The Implementation of an Explicit Charging and Discharge Lightning Scheme within the

WRF-ARW Model: Benchmark Simulations of a Continental Squall Line, a Tropical Cyclone, and a Winter Storm. *Monthly Weather Review* 141, 2390-415.

Fischer, G., Dyudina, U.A., Kurth, W.S., Gurnett, D.A., Zarka, P., Barry, T., Delcroix, M., Go, C., Peach, D., Vandebergh, R. and Wesley, A. Overview of Saturn lightning observations. Planetary Radio Emissions VII, Proceedings of the 7th International Workshop held at Graz, Austria, September 15-17, 2010. Edited by H.O. Rucker, W.S. Kurth, P. Louarn, and G. Fischer. Austrian Academy of Sciences Press, Vienna, 2011, p. 135-144.

Fletcher, L.N., Baines, K.H., Momary, T.W., Showman, A.P., Irwin, P.G.J., Orton, G.S., Roos-Serote, M., and Merlet, C. (2011). Saturn's Tropospheric Composition and Clouds from Cassini/VIMS 4.6-5.1 μm Nightside Spectroscopy. *Icarus* 214, 510-33. https://doi.org/10.1016/j.icarus.2011.06.006

Gautier, D., Hersant, F., Mousis, O., & Lunine, J.I. (2001). Enrichments in volatiles in Jupiter: a new interpretation of the Galileo measurements. *The Astrophysical Journal*, 550,L227-30 (Erratum *Ap.J.* 559, L183, 2001).

Gibbard, S.G. (1996). Lightning in the Solar System. Dissertation, University of Arizona.

Gibbard, S., Levy, E.H. & Lunine, J.I. (1995). Generation of lightning in Jupiter's water cloud. *Nature*, 378,592-5.

Gibbard, S., Levy, E.H. and Morfill, G.E. (1997). On the possibility of lightning in the protosolar nebula. *Icarus*, 130, 517-533.

Goodman, S.J. & MacGorman, D.R. (1986). Cloud-to-ground lightning activity in mesoscale convective complexes. *Monthly Weather Review*,114, 2320-8.

Goodman, S.J., Blakeslee, R.J., Koshak, W.J., *et al*. (2013). The GOES-R Geostationary Lightning Mapper (GLM). *Atmospheric Research* 125-126, 34-49. https://doi.org/10.1016/j.atmosres.2013.01.006

Guillot, T. (1995). Condensation of Methane, Ammonia, and Water and the Inhibition of Convection in Giant Planets. *Science* 269, 1697-9. https://doi.org/10.1126/science.7569896

Guillot, T., Stevenson, D.J., Atreya, S.K., Bolton, S.J., & Becker, H.N. (2020). Storms and the depletion of ammonia in Jupiter: I. Microphysics of mushballs. *J. Geophys. Res.: Planets*, 125, e2020JE006403. https://doi.org/10.1029/2020JE006403.

Gurnett, D.A., Shaw. R.R, Anderson, R.R., Kurth, W.S., & Scarf, F.L. (1979). Whistlers Observed by Voyager 1: Detection of lightning on Jupiter. *Geophysical Research Letters*, 6, 511-4.

Hemmes, H., Driessen, A., and Griessen, R. (1986). Thermodynamic properties of hydrogen at pressures up to 1 Mbar and temperatures between 100 and 1000 K. *J. Phys. C: Solid State Phys.*19, 3571-85.

Herbert, F. L., *et al*. (1987) . The upper atmosphere of Uranus: EUV occultations observed by Voyager 2. *JGR* 92, 15093.

Hueso, R., Sanchez-Lavega, A. & Guillot, T. (2002). A model for large-scale convective storms in Jupiter. *J. Geophys Res.*, 10 (E10), 5075. doi:10.1029/2001JE001839, 2002

Ingersoll, A.P., Gierasch. P.J., Banfield, D., Vasavada, A.R. & Galileo Imaging Team (2000). Moist convection as an energy source for the large-scale motions in Jupiter's atmosphere. *Nature*, 403, 630-2.

Ingersoll, A.P. and Kanamori, H. (1995). Waves from the collisions of comet Shoemaker-Levy 9 with Jupiter. *Nature*, 374, 706–708.

Janssen, M.A., Oswald, J.E., Brown, S.T., Gulkis, S., Levin, S.M., Bolton, S.J. *et al*. (2017). MWR: Microwave Radiometer for the Juno Mission to Jupiter. *Space Science Reviews* 213, 139-85.

Joiner, J. and Steffes, P.G. (1991). Modeling of Jupiter's Millimeter Wave Emission Utilizing Laboratory Measurements of Ammonia (NH_3) Opacity. *Journal of Geophysical Research* 96, 17463-70.

Kaiser, M.L., Zarka, P., Desch, M.D. and Farrell, W.M. (1991). Restrictions on the Characteristics of Neptunian Lightning. *Journal of Geophysical Research* 96, 19043-7.

Karpowicz, B.M. and Steffes, P.G. (2011). In search of water vapor on Jupiter: Laboratory measurements of the microwave properties of water vapor under simulated jovian conditions. *Icarus* 212, 210-23. doi:10.1016/j.icarus.2010.11.035

Keith, W.D. & Saunders, C.P.R. (1989). Charge transfer during multiple large ice crystal interactions with a riming target. *Journal of Geophysical Research – Atmospheres*, 94,13103-6.

Kessler, E. (1969). On the Distribution and Continuity of Water Substance in Atmospheric Circulation, chapter 8-I, pg. 29-30. Published by the American Meteorological Society, Boston, MA.

Koskinen, T.T. and Guerlet, S. (2018). Atmospheric structure and helium abundance on Saturn from Cassini/UVIS and CIRS observations. *Icarus*, 307, 161-71. https://doi.org/10.1016/j.icarus.2018.02.020

Lange, N.A. (1967). Lange's Handbook of Chemistry, 10th edition. Published by McGraw-Hill, New York, NY.

Lanzerotti, L.J., Rinnert, K., Dehmel, G., Gliem, F.O., Krider, E.P., Uman, M.A., and Bach, J. (1996). Radio frequency signals in Jupiter's atmosphere, *Science*, 272, 858-60.

Leconte, J., Selsis, F., Hersant, F. and Guillot, T. (2017). Condensation-inhibited convection in hydrogen-rich atmospheres - Stability against double-diffusive

processes and thermal profiles for Jupiter, Saturn, Uranus, and Neptune. *Astronomy & Astrophysics* 598, A98.

Lee, V., James, N.M., Waitukaitis, S., and Jaeger, H.M. (2018). Collisional Charging of Individual Sub-Millimeter Particles: Using Ultrasonic Levitation to Initiate and Track Charge Transfer. *Physical Review Materials* 2, 053602. https://doi.org/10.1103/PhysRevMaterials.2.035602

Lesprit, U., Paillat, T., Zouzou, N., Paquier, A., and Yonger, M. (2020). Triboelectric charging of a glass bead impacting against polymers: Influence of mechanical properties. *Journal of Electrostatics* 107, 103474. https://doi.org/10.1016/j.elstat.2020.103474

Li, C., Ingersoll, A., Bolton, S., Levin, S., Janssen, M., Atreya, S., *et al*. (2020). The water abundance in Jupiter's equatorial zone. *Nature Astronomy*. https://doi.org/10.1038/s41550-020-1009-3.

Li, C. & Chen, X. (2019). Simulating nonhydrostatic atmospheres on planets (SNAP): Formulation, validation, and application to the Jovian atmosphere. *Astrophys. J. Suppl. Series*, 240, 37. https://doi.org/10.3847/1538-4365/aafdaa.

Li, C. and Ingersoll, A.P. (2015). Moist convection in hydrogen atmospheres and the frequency of Saturn's giant storms. *Nature Geoscience* 8, 398-403.

Li, L., Ingersoll, A.P., Vasavada, A.R., Porco, C.C., Del Genio, A.D., Ewald, S.P. (2004). Life cycles of spots on Jupiter from Cassini images. *Icarus* 172:9-23.

Lindal, G.F. (1992). The atmosphere of Neptune: an analysis of radio occultation data acquired with Voyager 2. *The Astronomical Journal* 103, 967-82.

Little, B., Anger C.D., Ingersoll, A.P., Vasavada, A.R., Senske, D.A., Breneman, H.H., *et al*. (1999). Galileo images of lightning on Jupiter. *Icarus* 142:306-23.

Lowe, P.R. (1977). An approximating polynomial for the computation of saturation vapor pressure. *Journal of Applied Meteorology*, 16, 100-3.

Lunine, J.I., Hubbard, W.B., Burrows, A., Wang, Y-P., and Garlow, K. (1989). The effect of gas and grain opacity on the cooling of brown dwarfs. *The Astrophysical Journal*, 338, 314-37.

MacGregor, J.A., Winebrenner, D.P., Conway, H., Matsuoka, K., Mayewski, P.A. and Clow, G.D. (2007). Modeling englacial radar attenuation at Siple Dome, West Antarctica, using ice chemistry and temperature data. *Journal of Geophysical Research* 112, F03008.

Maggio, C.R., Marshall, T.C., and M Stolzenburg, M. (2009). Estimations of charge transferred and energy released by lightning flashes. *Journal of Geophysical Research*, 114, D14203.

Mondal, S. & Bhattacharya, A.B. (2015). Climatic variance and its effects on decametric Jovian signal reception at a high altitude station Darjeeling. *Indian Journal of Radio & Space Physics* 44, 126-31.

Moore, J.C. (2000). Models of Radar Absorption in Europan Ice. *Icarus* 147, 292-300.

Moses, J.I., Cavailé, T., Fletcher, L.N. and Roman, M.T. Atmospheric chemistry on Uranus and Neptune. *Philosophical Transactions of the Royal Society A* 378, 20190477. http://dx.doi.org/10.1098/rsta.2019.0477

NAS 2022. Origins, Worlds, and Life: A Decadal Strategy for Planetary Science and Astrobiology 2023-2032.

Orton, G.S., Fisher, B.M., Baines, K.H., Stewart, S.T., Friedson, A.J., Ortiz, J.L., Marinova, M., Ressler, M., Dayal, A., Hoffmann, W., Hora, J., Hinkley, S., Krishnan, V., Masanovic, M., Tesic, J., Tziolas, A., & Parija, K.C. (1998). Characteristics of the Galileo probe entry site from Earth-based remote sensing observations. *Journal of Geophysical Research*, 103, 22791–22814.

Orville, R. E., & Spencer, D.W. (1979), Global lightning flash frequency, *Mon. Weather Rev.*, 107, 934–943.

Palotai, C., Dowling, T.E., & Fletcher, L.N. (2014). 3D Modeling of interactions between Jupiter's ammonia clouds and large anticyclones. *Icarus* 232:141-56.

Peterson, M. (2019). Using Lightning Flashes to Image Thunderclouds. *Journal of Geophysical Research – Atmospheres* 124, 10175-85. https://doi.org/10.1029/2019JD031055

Pruppacher, H.R. and Klett, J.D. 1997. Microphysics of Clouds and Precipitation, Kluwer, Dordrecht, 954pp.

Qu, Y., Khain, A., Phillips, V., Ilotoviz,E., Shpund, J., Patade, S., & Chen, B.(2020). The role of ice splintering on microphysics of deep convective clouds forming under different aerosol conditions: Simulations using the model with spectral bin microphysics. *Journal of Geophysical Research:Atmospheres*,125, e2019JD031312.https://doi.org/10.1029/2019JD031312

Rinnert, K., Lanzerotti, L.J., Uman, M.A., Dehmel, G., Gliem, F.O., Krider, E.P., and Bach, J. Measurements of radio frequency signals from lightning in Jupiter's atmosphere (1998) *J. Geophys. Res.*, 103, 22,979-22,992.

Rosenkranz, P.W. (1998). Water vapor microwave continuum absorption: A comparison of measurements and models. *Radio Science* 33, 919-28.

Rutledge, S.A. & Hobbs, P.V. (1984). The Mesoscale and Microscale Structure and Organization of Clouds and Precipitation in Midlatitude Cyclones. XII: A Diagnostic Modeling Study of Precipitation Development in Narrow Cold-Frontal Rainbands. *Journal of the Atmospheric Sciences*, 41, 2949–2972.

Ruzaikin, V., Lukashov, I., Fedorenko, T., and Abashin, S. (2022). The equilibrium contact angle of ammonia-stainless steel interface. Results in Engineering 16, 100691. https://doi.org/10.1016/j.rineng.2022.100691

Saunders, C. P. R. (1993). A review of thunderstorm electrification processes. *J. Appl. Meteor.*, 32, 642–655.

Saunders, C. (2008). Charge separation mechanisms in clouds. *Space Sci. Rev.*, 137, 335-353.

Segelstein, D.J. (1981). The Complex Refractive Index of Water. Master of Science thesis, University of Missouri-Kansas City.

Seiff, A., Kirk, D.B., Knight, T.C.D., Young, R.E., Mihalov, J.D., Young, L.A., *et al*. (1998). Thermal structure of Jupiter's atmosphere near the edge of a 5-μm hot spot in the north equatorial belt. *Journal of Geophysical Research*, 103, 22857-89.

Spilhaus, AF. (1948). Raindrop size, shape, and falling speed. *Journal of Meteorology*, 5,108-10.

Stetten, A.Z., Golovko, D.S., Weber, S.A.L., and Butt, H.-J. (2019). Slide electrification: charging of surfaces by moving water drops. *Soft Matter*, 15, 8667-79.

Stolzenberg, M., Rust, D., Smull, B.F. and Marshall, T.C. 1998. Electrical structure in thunderstorm convective regions.1. Mesoscale convective systems. *Journal of Geophys. Research*, 103, 14059-14078.

Stoker, C.R. (1986). Moist convection: a mechanism for producing the vertical structure of the Jovian equatorial plumes. *Icarus*, 67, 106-125.

Sullivan, S.C., Hoose, C., Kiselev, A., Leisner, T., and Nenes, A. (2018). Initiation of secondary ice production in clouds. *Atmospheric Chemistry and Physics* 18, 1593-610. https://doi.org/10.5194/acp-18-1593-2018

Visscher, C. & Fegley, B. (2005). Chemical constraints on the water and total oxygen abundances in the deep atmosphere of Saturn. *The Astrophysical Journal* 623, 1221-7.

von Zahn, U., Hunten, D.M., and Lehmacher, G. (1998). Helium in Jupiter's atmosphere: Results from the Galileo probe helium interferometer experiment. *Journal of Geophysical Research*, 103, 22815–22830.

Wahl, S.M., Hubbard, W.B., Militzer, B., Guillot, T., Miguel, Y. Movshovitz, N., *et al*. (2017). Comparing Jupiter interior structure models to Juno gravity measurements and the role of a dilute core. *Geophysical Research Letters*, 44, 4649-59.

Wang, D., Gierasch, P.J., Lunine, J.I., and Mousis, O. (2015). New insights on Jupiter's deep water abundance from disequilibrium species. *Icarus*, 250, 154–164.

Wexler, A. (1977). Vapor Pressure Formulation for Ice. *The Journal of Research of the National Bureau of Standards Section A* 81A:5-20.

Wiktorowicz, S.J. & Ingersoll, A.P. (2007). Liquid water oceans in ice giants. *Icarus* 186, 436-47.

Wong, M.H., Lunine, J., Atreya, S.K., Johnson, T., Mahaffy, P.R., Owen, T.C., & Encrenaz, T. (2008). Oxygen and other volatiles in the giant planets and their satellites. In Reviews in Mineralogy and Geochemistry Vol. 68: Oxygen in the Solar System, Chapter 10 (G.J. MacPherson et al. eds.), Mineralogical Society of America, Chantilly, VA.

Wong, M.H., Mahaffy, P.R., Atreya, S.K., Niemann, H.B., & Owen, T.C. (2004). Updated Galileo probe mass spectrometer measurements of carbon, oxygen, nitrogen, and sulfur on Jupiter. *Icarus*, 171, 153-70.

Workman, E.J. and Reynolds, S.E. (1950). Electrical phenomena occurring during the freezing of dilute aqueous solutions and their possible relationship to thunderstorm electricity. *Phys. Rev.* , 78, 254-259.

Yair, Y., Levin, Z., & Tzivion, S. (1992). Water-Cumulus in Jupiter's Atmosphere: Numerical Experiments with an Axisymmetric Cloud Model. *Icarus*, 98, 72-81. https://doi.org/10.1016/0019-1035(92)90208-O

Yair, Y., Levin, Z., & Tzivion, S. (1995a). Microphysical processes and dynamics of a Jovian thundercloud. *Icarus* 114, 278-299.

Yair, Y., Levin, Z., & Tzivion, S. (1995b). Lightning generation in a Jovian thundercloud: Results from an axisymmetric numerical cloud model. *Icarus*, 115,421-34.

Yair, Y., Levin, Z., & Tzivion, S. (1998). Model interpretation of Jovian lightning activity and the Galileo Probe results. *Journal of Geophysical Research*, 103, 14157-66.

Yair, Y., Fischer, G., Simões, F., Renno, N. & Zarka, P. (2008). Updated review of planetary atmospheric electricity. *Space Science Reviews*, 137, 29-49.

Zarka, P. & Pedersen, B.M. (1986). Radio detection of uranian lightning by Voyager 2. *Nature* 323, 605-8.

Zarka, P., Farrell, W.M., Kaiser, M.L., Blanc, E. and Kurth, W.S. (2004). Study of solar system planetary lightning with LOFAR. *Planetary and Space Science* 52, 1435-47.

Zuchowski, L.C., Yamazaki, Y.H., & Read, P.L. (2009). Modeling Jupiter's cloud bands and decks 2. Distribution and motion of condensates. *Icarus* 200, 563-73.