## ABSTRACT

LINDSAY, ALEXANDER DAVID. Coupling of Plasmas and Liquids. (Under the direction of Steven Shannon.)

Plasma-liquids have exciting applications to several important socioeconomic areas, including agriculture, water treatment, and medicine. To realize their application potential, the basic physical and chemical phenomena of plasma-liquid systems must be better understood. Additionally, system designs must be optimized in order to maximize fluxes of critical plasma species to the liquid phase. With objectives to increase understanding of these systems and optimize their applications, we have performed both comprehensive modelling and experimental work. To date, models of plasma-liquids have focused on configurations where diffusion is the dominant transport process in both gas and liquid phases. However, convection plays a key role in many popular plasma source designs, including jets, corona discharges, and torches. In this dissertation, we model momentum, heat, and neutral species mass transfer in a convection-dominated system based on a corona discharge. We show that evaporative cooling produced by gas-phase convection can lead to a significant difference between gas and liquid phase bulk temperatures. Additionally, convection induced in the liquid phase by the gas phase flow substantially increases interfacial mass transfer of hydrophobic species like NO and $NO_2$. Finally, liquid kinetic modelling suggests that concentrations of highly reactive species like OH and ONOOH are several orders of magnitude higher at the interface than in the solution bulk.

Subsequent modelling has focused on coupling discharge physics with species transport at and through the interface. An assumption commonly seen in the literature is that interfacial loss coefficients of charged species like electrons are equal to unity. However, there is no experimental evidence to either deny or support this assumption. Without knowing the true interfacial behavior of electrons, we have explored the effects on key plasma-liquid variables of varying interfacial parameters like the electron and energy surface loss coefficients. Within a reasonable range for these parameters, we have demonstrated that the electron density on the gas phase side of the interface can vary by orders of magnitude. Significant effects can also be seen on the gas phase interfacial electron energy. Electron density and energy will play important roles in determining gas phase chemistry in more complex future models; this will in turn feed back into the liquid phase chemistry. To remove this uncertainty in interfacial behavior, we recommend finer scale atomistic or molecular dynamics simulations. Efficient coupling of the highly non-linear discharge physics equations to liquid transport required creation of a new simulation code named Zapdos, built on top of the MOOSE framework. The operation and capabilities of the code are described in this work. Moreover, changes made to the MOOSE framework allowing coupling of physics across subdomain boundaries, necessary for plasma-liquid coupling, are also detailed.

In the latter half of this work, we investigate experimental optimization and characterization of plasma-liquid interactions surrounding a unique very high frequency (VHF) plasma discharge. Several geometric configurations are considered. In the most promising set-up, the discharge is pointed upwards and water is pumped through the source's inner conductor until it forms a milimeter thick water layer on top of the powered electrode. This maximizes the amount of charged and neutral species flux received by the aqueous phase as well as the amount of water vapor created in the gas phase. Additionally, the configuration eliminates electrode damage by providing an infinitely renewable liquid surface layer. The presence of large amounts of water vapor and OH radicals is confirmed by optical emission and broadband absorption spectroscopy. Characterization of liquid phase species like $NO_3^-$, $NO_2^-$, and $H_2O_2$ is carried out through ion chromatography (IC) and colorimetric measurements.

After detailing the design and characterization of our plasma-liquid systems, we illustrate their applications to plant fertilization and wastewater disinfection. In a four-week collaborative experiment with the NCSU greenhouse, plants that received plasma-treated water grew significantly larger than plants that received tap water. This is directly attributable to the approximately hundred mg/L of $NO_3^-$ dissolved into solution by the plasma. The VHF source also proved effective at removing several aqueous contaminants designated harmful to humans by the EPA. Air plasma treatment of solutions contaminated with 1,4-dioxane showed log reduction times competitive with other advanced oxidative processes (AOP). Argon treatment of dixoane was an order of magnitude more effective in terms of log reduction time, although the associated costs are significantly higher. Perfluorooctanesulfonic acid (PFOS) proved resistant to several VHF design iterations. However, the water electrode design introduced in the passage above achieved a log reduction in low level PFOS concentrations over the course of twenty five minutes, suggesting that it may be viable as an advanced technology for degradation of persistent perfluorinated compounds.

Future work will serve to further unify the modelling and experimental work presented here. Three dimensional models are being actively developed to explore the spreading of discharges over water as a function of the water conductivity. Electromagnetic models are also being introduced into Zapdos that will enable simulation of the VHF source in contact with liquids. Completion of those models will be followed by simulations attempting to reproduce the gaseous OH density and liquid phase $NO_x^-$ and $H_2O_2$ concentration measurements.

# Coupling of Plasmas and Liquids

by
Alexander David Lindsay

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Nuclear Engineering

Raleigh, North Carolina

2016

APPROVED BY:

David Graves

Detlef Knappe

Mohamed Bourham

John Gilligan

Steven Shannon
Chair of Advisory Committee

## DEDICATION

This is dedicated to my Mom, Dad, and Sister. Without them, I would never have gotten here. They've supported me through the good times and the bad. I couldn't have been blessed with a more loving family. This degree means a lot, but they will always mean immeasurably more.

# BIOGRAPHY

Alexander David Lindsay was born in Seattle, WA on December 11, 1987 to Janet and Tom. Little sister Jessica arrived five and a half years later and has been a nuisance ever since. Alexander attended the University of Washington and obtained his bachelor's degree in chemical engineering in August of 2010. He began his doctoral studies at North Carolina State University in August 2011. After receiving his doctorate, Alexander will commence post-doctoral studies at the University of Illinois in the National Center for Supercomputing Applications.

# ACKNOWLEDGEMENTS

I would like to thank my whole dissertation committee for teaching me so much about my field. Special thanks goes to Dr. Graves for hosting me in his lab for a full year and for his invaluable recommendations for modeling plasma physics. His theoretical knowledge in our field is unmatched. My greatest thanks goes to Dr. Shannon, who has not only been the most fantastic academic advisor I could have ever asked for but also a tremendous friend. Our relationship is hopefully just beginning.

Thanks to Jessie Lindsay for being as big a participant in the first four years of my journey here as anyone. She saw through things that a lot of other people wouldn't have been able to and made me a much better person in the process. Thanks also to Kyle Weinfurther, David Peterson, and Ben Daniel for helping lift me out of a tough time that overlapped with the most important stage of my research.

Thanks to all the members of the Shannon and Graves group that I had a chance to work and hang out with, particularly Brandon Byrns who mentored me for my first two years, teaching me so much about the VHF source and atmospheric plasmas. Life's pretty good when your lab partner is also one of your closest friends.

I would like to thank Brandon Curtis for introducing me to the open source way of thinking. It's revolutionized my research and career plans in the best possible way. I know now that I can learn just about anything. Thanks also goes to Jannis Teunissen for greatly extending my knowledge of open source tools and for teaching me how to think about modelling problems. I sincerely hope we can collaborate in the future. Finally, deep thanks goes out to the Moose development team and community who have taught me so much about computer science, math, physics, and engineering. It's been a great ride.

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

CHAPTER

1

# INTRODUCTION & BACKGROUND

## 1.1 Basic Science of Plasma-Liquid Interactions

There is a general interest in the study of plasma-liquid interactions within the scientific community for an array of applications, including but not limited to biomedicine and biological disinfection [5, 6, 7, 8, 9, 10], chemical disinfection [11, 12, 13], and agricultural applications. [14, 15] In order to successfully realize these applications and develop mature technologies, the basic science underlying these coupled plasma gas-liquid systems must be well understood. The richness and complexity of plasma-liquid systems require detailed study; an appreciation for the many physics involved can be gained by examining fig. 1.1. Some of the physiochemical phenomena include electromagnetics, charged and neutral species transport, heat transport, fluid flow, gas and liquid chemistry, and circuits (e.g. transmission lines or other external circuit elements). Each of these phenomena affect and feed back into each other, creating a rich multi-physics problem. Recent experimental, modeling, and review works have enhanced our understanding of many of the important processes involved in these systems. Lukes et. al. [16] conducted an in-depth study of the aqueous phase chemistry produced by atmospheric pressure air discharges. Through the use of phenol as a chemical probe, the group saw evidence of generation of OH, NO, and $NO_2$ radicals at the interface as well as long term generation of OH and $NO_2$ through dissociation of ONOOH, itself a product of the reaction of $HNO_2$ and $H_2O_2$. Lukes's elucidation of the radical generation pathway from peroxynitrite decay corroborates the work conducted by Traylor et. al. [17] in which the long-term bacterial efficacy of plasma-activated water (PAW) was investigated. Traylor found that hydrogen peroxide and nitrite concentrations diminished significantly over the course of several days, corresponding to a significant decrease in the solution's bactericidal properties. The decay in $H_2O_2$ and $NO_2^-$ concentrations is consistent with Lukes's reaction pathway of $H_2O_2 + H^+ + NO_2^- \rightarrow ONOOH$; the drop in the solution's anti-microbial behavior likely corresponds to a drop in OH and $NO_2$ radical production through ONOOH decay.

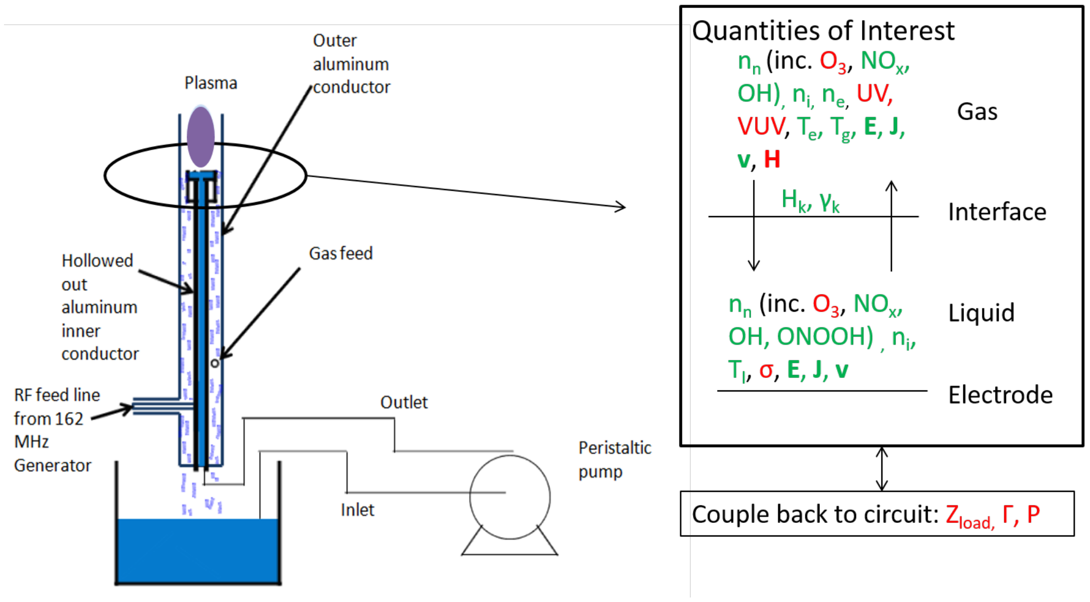

Plasma
Outer aluminum conductor
Hollowed out aluminum inner conductor
Gas feed
RF feed line from 162 MHz Generator
Outlet
Inlet
Peristaltic pump
Quantities of Interest
n_n (inc. O_3, NO_x, OH), n_i, n_e, UV, VUV, T_e, T_g, E, J, v, H
Gas
H_k, γ_k
Interface
n_n (inc. O_3, NO_x, OH, ONOOH), n_i, T_l, σ, E, J, v
Liquid
Electrode
Couple back to circuit: Z_load, Γ, P

Figure 1.1 Cartoon of plasma-liquid experiment with identification of key variables of interest. $n_n$ represents densities of neutral species, $n_i$ the densities of ions, $n_e$ the densities of electrons, UV and VUV are ultra-violet and very low wavelength ultra-violet, $T_e$ is the electron temperature, $T_g$ is the gas temperature, **E** is the electric field, **J** is the plasma current, **v** is the fluid flow velocity, **H** is the magnetic field, $H_k$ represents the Henry's law coefficient for specie $k$ and is an indicator of hydrophobicity, $\gamma_k$ represents surface loss coefficients, $T_l$ is the liquid temperature, $\sigma$ is the solution conductivity, $Z_{load}$ is the total load impedance of the plasma-liquid system, $\Gamma$ is the circuit reflection coefficient, and P is the delivered power. Green quantities have been observed experimentally and are included in at least one of the two models in chapter 2. Red quantities have been observed experimentally but have yet to be added to our models.

Bruggeman et. al. [18] investigated DC discharges generated directly in liquid. They observed two distinct modes: for low aqueous solution electrical conductivities a streamer-like discharge formed in the liquid itself; at higher conductivities the discharge was generated in a large gaseous bubble. For both modes the group was able to measure the electron density and gas temperature of the discharges. Pavlovich et. al. [19] investigated regime changes in gas chemistry when treating E. coli. They discovered that at low power densities, the dominant reactive specie is ozone; at high power densities the gas phase chemistry becomes $NO_x$ dominated. The researchers also observed that in the low power density regime, the gas has to undergo substantial mixing with the liquid phase in order to kill the bacteria infesting the solution. They hypothesized that this is because of the relatively high hydrophobicity of ozone; diffusion alone does not generate sufficient mass transfer of ozone between gas and liquid phases. Yagi and co-authors investigated the impact of varying gas flow rate on humidity and OH using two-dimensional laser induced fluorescence (LIF). [20] They found that increasing gas flow rate creates a low-humidity region in the vicinity of the discharge, which in turn leads to a decreased rate of OH radical production. They reason that this change in gas phase composition will likely affect radical fluxes to the liquid surface. In another work by Bruggeman [21], the behavior of point-to-plane DC discharges impinging on water is explored. When water is the cathode, the discharge is filamentary in nature; when the water is the anode, the discharge is much more stable and diffuse. In the latter configuration, the rotational temperatures of OH and $N_2$, good indicators of the gas temperature, are identically 3250 K in the discharge's positive column; however, near the water anode, the temperature drops by 2500 K. The authors hypothesize that in the case of the water anode, the water acts as both an electrical stabilizer and a heat sink.

What is required for a thorough understanding of coupled plasma-liquid systems is a model capable of describing the unity of all the experimental observations cited above. Such a model must be able to describible phenomena that occur on vastly different time and length scales. For instance electron transport occurs on nanosecond time scales whereas some reactions in aqueous solution take place over the course of days as witnessed by the observations in [17]. Some modeling works have recently been conducted that begin to realize the comprehensive plasma-liquid description we desire. One excellent work, [22], uses three decoupled regions to explore the plasma-liquid dynamics: a bulk gas region, a gas-liquid interface layer, and a semi-infinite liquid region. One of the key conclusions of that work is that highly reactive plasma chemistry (represented by $OH_2$ or $O_3$) only penetrates about 10-20 $\mu$m into the liquid bulk. With He-$O_2$ as the working gas and using a low power density, they predict that dry downstream chemistry will be dominated by O, $O_2$(a) and $O_3$. They predict that liquid phase chemistry will depend principally on superoxide ($O_2^-$), $H_2O_2$, and either $HO_2$ or $O_3$, with the latter two species decaying in the first tens of microns while the former two persist for milimeter scales. While the work gives a comprehensive description of the discharge conditions and the chemistries in the respective gas and liquid phases, the decoupled nature of the three domains makes it a non-ideal framework for investigating full coupling between the phases. For instance, particle fluxes are assumed to be mono-directional from the discharge phase to the gas-liquid interface layer. Thus evaporation of species and its effect on discharge physics and chemistry are not considered. Similarly, the effect of water evaporation on both gas and liquid temperature profiles and subsequently on reaction rates are not considered. Many of these effects can only be realized with a fully-coupled, bidirectional model.

Some of the most detailed plasma-liquid modeling work has come out of Mark Kushner's group at the University of Michigan. A particularly seminal work is that of Tian et. al. [1], wherein they report on results obtained using the model *nonPDPSIM*. More detailed descriptions of *nonPDPSIM* can be found in [23, 24]. The model includes solution of Poisson's equation, (drift)-diffusion equations for (charged) and neutral species, an electron energy equation, and a radiation transport equation. Using this highly detailed description of the physics, the authors are able to make predictions about the important species formed in both gas and liquid phases as well as the mechanisms by which they are formed. For instance, aqueous $O_3$ comes primarily from dissolution of $O_3$(g) whereas significant portions of the OH(aq) and H(aq) concentrations can be attributed to photionization and photodissociation of $H_2O$ at the liquid surface. Reactivity at a substrate positioned a few hundred microns into the liquid phase comes primarily from $H_2O_2$, $O_3$, and $ONOO^-$, all species that have received significant attention in the experimental plasma-liquid work. The work in [25] extends *nonPDPSIM* to invesigation of how cells and tissue below a water layer might distort and affect the electric fields and particles fluxes coming form the plasma. Though *nonPDPSIM* is perhaps the finest code currently being used in the plasma-liquid community, it does have some limitations. As described in [24] much of the physics is segregated, e.g. charged particle densities are updated before updating the electron energy which is in turn updated before neutral particle densities, etc. While in some simulation cases segregated methods may be more efficient in terms of memory usage, fully coupled methods are required when physics are very tightly coupled. In the case of very tight coupling, segregated methods may not converge. In a comparison of monolithic (fully-coupled) and segregated solvers for fluid-structure interaction (FSI) problems, it was found that segregated solves diverged rapidly in unsteady cases unless very strong under-relaxation was applied. [26]. Moreover, the authors found that monolithic solves competed very favorably with segregated counterparts even under weakly interacting conditions. Even in large problems, by removing select blocks from their preconditioner, the authors were able to efficiently solve a variety of challenging non-linear systems with the monolithic method. The above research suggests that a monolithic architecture is a good choice in almost all cases and the ideal in highly coupled systems like plasma-liquids when charged and neutral species transport, heat and momentum transport, radiation transport, and Maxwell's equations are all considered simultaneously.

Building off the work of [27], Shirafuji et. al. investigated electric double layer formation in an arbitrary liquid medium XY in contact with an RF discharge. [28] In their simulation, implemented in the proprietary package Comsol, the bottom of the liquid phase is grounded while the potential at the other end of the domain oscillates. When the powered electrode is at a positive potential and ions are being pushed from the gas phase into the liquid phase, the authors observe a postive charge layer on the liquid side of the interface and a small negative charge layer at the bottom of the liquid volume. When the powered electrode is negative, a negative charge layer forms on the liquid side of the interface with a small positive charge layer at the liquid bottom. The authors conclude by suggesting that less mobile ions may preferentially appear at the interface.

Despite many of the excellent modeling works already published, numerous basic science questions about plasma-liquid systems remain. Several important questions are enumerated below:

1. None of the comprehensive models described in the literature consider the role of convective fluid flow in the plasma-liquid dynamics. How does advection in atmospheric jets or corona discharges affect temperature profiles and reaction kinetics, the rate of mass transfer from gas to liquid, or the distribution of species in both gas and liquid phases?

2. How do assumptions about conditions at the interface feed back into the plasma and liquid dynamics? For example, modellers [1, 28] generally assume that electrons moving from gas to liquid have a sticking coefficient of unity while there is no experimental evidence to either disprove or support that assumption. How would changing electron and/or energy absorption at the interface affect plasma dyanmics and liquid characteristics?

3. In a work investigating solvation of electrons, Rumbach et. al. [2] observed spreading of a needle-to-water discharge as solution conductivity decreased (see fig. 1.2). What is the mechanism that explains this spreading?

The work presented in this dissertation does much to answer the questions raised in items 1

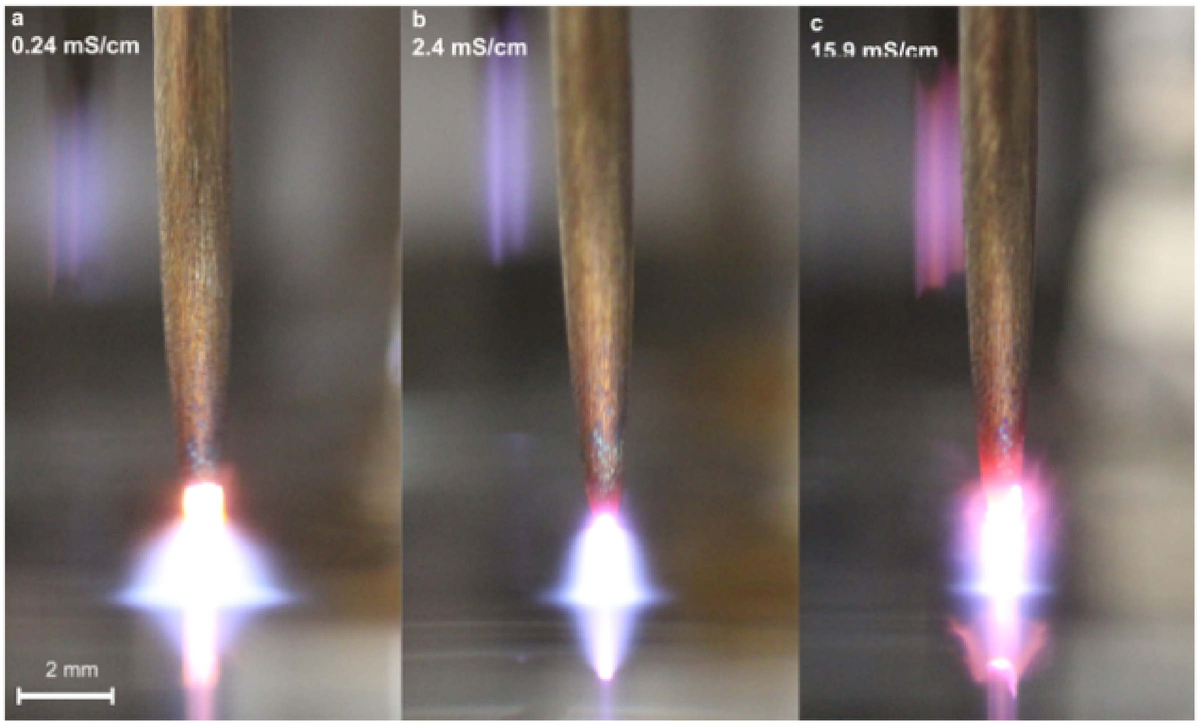


Figure 1.2 Figure taken from [2] showing the spreading of a DC discharge as solution conductivity is decreased.

and 2 and in the process lays a roadmap for answering item 3. The questions raised in item 1, while unique to the low-temperature plasma community, fall largely under the umbrella of traditional chemical engineering transport and consequently can be answered using traditional chemical engineering software such as Comsol. [29] However, items 2 and 3 require a thorough description of electron and electron energy transport in the plasma. In the author's personal experience, Comsol is highly inefficient and may even fail to converge when solving the highly non-linear equations that describe plasmas. The problem is not impossible, as evidenced by work in [28, 30, 31]; however, in all the works referenced the number of degrees of freedom is relatively small, thus an inefficient solve may be tolerable (to patient people). As problems get larger, however, inefficient solves become intolerable. It becomes necessary to have control over definition of Jacobian functions to ensure the most efficient solutions of non-linear equations. Moreover, in order to reduce computation time, it may be highly desirable to deploy the simulation across many CPU cores. This may be cost prohibitive when using a commercial simulation package. Recognizing the need for a multi-physics framework code and preferring not to start from scratch when many codes already exist, we perused OpenFOAM[32], Elmer[33], and Code Aster[34] before settling on the Multiphysics Object-Oriented Simulation Environment (MOOSE) primarily developed by Idaho National Lab (INL).[35] MOOSE is a finite element framework that pushes coding of the physical governing equations onto an application developer; it is the responsibility of the developer to write residual and Jacobian functions that define the physics and dictate the efficiency of the non-linear solve respectively. We have created a MOOSE application called Zapdos, whose source code can be found at [36], for simulation of low-temperature plasmas with the possiblity for coupling to liquid systems. For those familiar with commercial multi-physics packages like Comsol, MOOSE can be thought of as being analogous to the base Comsol Multiphysics framework, while Zapdos is somewhat analogous to Comsol's Plasma Module. Though Zapdos is stacked on top of MOOSE, its construction has been an intensive task. To date Zapdos has over 23,000 lines of code; that number does not include the 1,400 lines that we added to the MOOSE framework itself necessary to enable interfacing of plasma and liquid domains.

To summarize, having reviewed the literature and having identified some key fundamental questions remaining in the field of plasma-liquids, it is our opinion that a new tool for plasma-liquid simulation, Zapdos, is necessary for the following reasons:

- Coupling of plasmas and liquids is a rich area of physics and chemistry that involves ionized gas dynamics, heat, momentum, and mass transport of both charged and neutral species, behavior of charges at interfaces, electrochemistry, etc. Tens to hundreds of species can be involved in hundreds to tens of thousands of reactions. This is a multi-physics problem with the potential for hundreds of thousands to millions of degrees of freedom. Consequently a versatile simulation framework should be massively parallelizable. Any applications built on top of MOOSE are automatically parallel; some applications have run on over 100,000 CPU cores. MOOSE is of particular value to our group because it is certified to run on ORNL's Titan, for which we have designated computing time allotments.

- By default, MOOSE applications are fully implicit and fully coupled. Full coupling enables solution of tightly coupled, highly-nonlinear physics problems such as those encountered in plasma-liquid systems. Fully implicit schemes allow stable simulation of physics that occur over dramatically different time scales.

- Because of its open source nature, MOOSE is highly flexible and highly extensible. As will be described later in section 3.2, a competent programmer can add capabilities to the framework that are necessary to his application's success. Moreover, MOOSE's user object system allows wrapping of external libraries. This has the potential to be extremely useful for some future work; in section 2.2 we conclude from our results that atomistic or molecular simulations at the interface may be necessary for elucidation of some key interfacial properties. Other MOOSE researchers are already investigating wrapping SPPARKS (a kinetic Monte Carlo simulator) and LAMMPS (a molecular dynamics simulator) for their own applications. We may do something similar.

- In the same vein as the above bullet, MOOSE employs perhaps the most general technique for discretizing partial differential equations, the Finite Element Method (FEM). By default MOOSE applications use a Continuous Galerkin discretization, but the application developer can easily add Discontinuous Galerkin methods. Additionally, MOOSE offers a wide array of test and shape functions, allowing the developer to, for example, reproduce finite volume methods.

- MOOSE gives the application developer complete control over residual and Jacobian function definitions. Consequently, any physics that can be written in a weak form can be included in a MOOSE application.

- In the era of a global internet, it is our firm belief that codes used for academic purposes should be openly available for both peer-review and collaborative development. Many of the leading codes used in the low-temperature plasma community, including *nonPDPSIM*, are not open to the public and thus cannot be held to the highest standard of review. In addition, without knowledge of the code's inner workings, reproduction of modeling results is more difficult. These are not criticisms of the code authors but rather a motivation for new code that *is* openly available for public inspection and review as well as modification and customization.

## 1.2 Experimental Design and Applications of Plasma-Liquid Interactions

### 1.2.1 Maximizing Interactions between Plasmas and Liquids

The above introduction and background motivates the need for additional modelling and simulation research on plasma-liquid systems. However, the experimental design of atmospheric pressure plasmas and their coupling with liquids is also critical for their applications in biomedicine, wastewater disinfection, agriculture, etc. A good review of atmospheric pressure plasma sources is given in [37]. In the article, the authors overview several popular discharges, including pulsed corona, DBD, pencil torches, ICP torches, the atmospheric pressure plasma jet (developed by Jeong et. al. [38]), and microwave discharges like the TIA developed by Moisan et. al. [39] In addition to the review by Tendero, Locke et. al. outline some discharges commonly used in plasma-liquid systems, focusing mostly on HV systems including coronas and gliding arc discharges. [40] It is worthwile to note that many of the discharges used in plasma-liquid systems are filamentary in nature with water interaction spot sizes often around 1 mm in radius. For applications which require generation of large amounts of reactive species in the liquid, e.g. waste-water disinfection for example, such small areas of interaction are not ideal. The atmospheric pressure source developed by our group, described in detail in [41], generates discharges roughly 2 cm in diameter with a plasma column that can span tens of centimeters. Such a large volume discharge that presents large surface areas for interaction is ideal for water treatment. This dissertation presents various geometric configurations for coupling the group's very high frequency (VHF) source to water. Along with exploiting the large volume of the discharge, we consider a configuration in which the water solution sits on top of the powered electrode, exposing the solution to increased fluxes of charged particles. This is somewhat similar to the gliding arc discharge shown in Figure 1K of [40], however, our glow discharge is steady and continuous as opposed to transient and filamentary. Consequently, we expect our powered electrode configuration, described in detail in section 4.5 to be state-of-the-art in terms of integrated charged, neutral, and UV fluxes from plasma to liquid.

### 1.2.2 Fertigation

Before the 1900s, nitrogen fixation occurred only naturally through lightning induced dissociation and reaction between atmospheric nitrogen and oxygen. In 1903 the Norwegian team of Birkeland and Eyde attempted a copy of nature's fixation process when they flowed air through a thermal arc, creating nitric oxides which were then converted into nitric acid and finally a solid nitrate salt. [42] Because of the intense energy requirements-17 kWh/kg nitric acid-the Birkeland-Eyde process was gradually replaced in Norway by the Haber and Ostwald processes.[43] In more recent years, generation of nitrates and nitrites in aqueous solution has been demonstrated with multiple non-thermal air discharges, including gliding arc, corona, and DBD.[44, 40, 45]

In agreement with the solution chemistries produced by the preceding non-thermal discharges, our VHF large volume glow discharge has demonstrated the ability to infuse nitrates into an aqueous medium, motivating a study of the impact of plasma activated water (PAW) on some traditional plant crops, outlined in detail in section 5.1. This research contributes to the early work of Birkeland and Eyde and more contemporary work into agricultural applications of plasmas. There is significant literature on exposure of plant seeds either directly to the discharge or the discharge afterglow. Sera et. al.[46] investigated the effects of four plasma source types, including gliding arc, downstream microwave, and surface DBD (SDBD) on the growth of buckwheat seeds. They found that gliding arc improved seed growth while SDBD in close proximity to the seeds inhibited growth. Bormashenko et. al.[47] found that an RF plasma generated under vacuum conditions increased the wettability of seed surfaces and seed germination rates. Zhou et. al.[48] used the afterglow of a DBD to increase seed growth in tomatoes while [49] and [50] have examined plasmas for improving wheat seed germination and eliminating fungus in grains and legumes respectively. While the literature for plasma treatment of seeds is extensive, the literature for using plasmas as a long-term aid in plant growth is less so, perhaps because of the historical failure of Birkeland and Eyde. A couple of recent studies have been published, however. Takaki et. al.[51] developed a unique system in a which a high voltage pulsed electrohydraulic discharge was used to both eliminate bacteria in recycled fertilizer water and to generate nitric acid, itself a fertilizer. They were able to demonstrate a significant increase in plant growth. Park et. al.[52] examined thermal spark discharge, gliding arc, and transferred arcs for generation of nitrogen species in water and subsequent application to a variety of plants. They found that the non-thermal gliding arc discharge produced the most promising plant growth results. However, in a study with radishes, banana peppers, and tomatoes, gliding arc treated water and a spring water control produced growth rates that were within experimental error of each other. Ingels et. al. [53] suggest that non-thermal plasma processes have a theoretical limit of 6.4 gigajoules per ton nitrogen (GJ/tN) net energy consumption which is six times lower than the Haber-Bosch process which requires 36 GJ/tN. The plasma figure incorporates the novel idea of using PAW to acidify readily available manure on farms, converting volatile ammonia into ammonium that can then be used as a fertilizer. To build on the work of these pioneering studies, we present a unique low-voltage large-volume glow discharge capable of generating PAW which has a statistically significant positive effect on the growth of radishes, marigolds, and tomatoes.

### 1.2.3 Pollutant Remediation in Wastewater

#### 1.2.3.1 Dioxane

1,4-dioxane, more commonly known as simply dioxane, has the chemical structure shown in fig. 1.3. [54] It is largely immune to conventional water treatment techniques. It is very hydrophilic and has a very low vapor pressure, so carbon adsorption and air stripping are not feasible. [55] Additionally, dioxane is resistant to biotransformations. [55] With conventional techniques unable to degrade the chemical, a class of treatments known as Advanced Oxidative Techniques (AOTs) must be used. A typical AOT process may employ ozone, hydrogen peroxide, or other hydroxyl producing precursors. The general mechanism for hydroxyl production using $H_2O_2$ as a precursor is homolytic bond cleavage of the O-O bond with UV light: [56]

$$H_2O_2 + UV \rightarrow 2OH \tag{1.1}$$

The mechanism for OH production via $O_3$ is a little more complex: [57]

$$O_3 + OH^- \rightarrow HO_2^- + O_2 \tag{1.2}$$

$$O_3 + HO_2^- \rightarrow HO_2 + O_3^- \tag{1.3}$$

$$O_3^- + H^+ \rightarrow HO_3 \tag{1.4}$$

$$HO_3 \rightarrow OH + O_2 \tag{1.5}$$

Photocatalytic oxidation with titanium dioxide ($TiO_2$) is another AOT that has received considerable attention in the literature. [58, 58, 59, 60] OH production from $TiO_2$ and UV is theorized to proceed through the following steps: [57]

$$TiO_2 + UV \rightarrow e^- + h^+ \tag{1.6}$$

$$Ti(IV) + H_2O \rightleftharpoons Ti(IV) - H_2O \tag{1.7}$$

$$Ti(IV) - H_2O + h^+ \rightleftharpoons Ti(IV) - OH + H^+ \tag{1.8}$$

where $e^-$ here represents an excited electron and $h^+$ an electron gap. Both ozone and hydrogen peroxide are readily created by atmospheric pressure discharges in contact with liquids depending on the power density of the discharge. [19] Additionally UV and OH, created at the interface from impinging gas processes and in the bulk from decomposition of longer lived species, are expected to be formed in abundance. [1] Thus plasmas are likely to be a suitable candidate for treating aqueous dixoane solutions. This is explored using the VHF source in section 5.2.2.

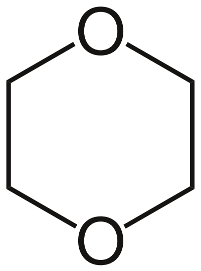


Figure 1.3 Chemical structure of dioxane.

Once OH is generated in solution, degradation of contaminants is expected to proceed through either hydrogen abstraction or electrophilic addition. This is illustrated through attack on benzene in fig. 1.4. [61] It is speculated that as long as there are sufficient OH radicals, contaminant degradation will proceed until all fragments are converted into small, stable, terminal species like $H_2O$ and $CO_2$.

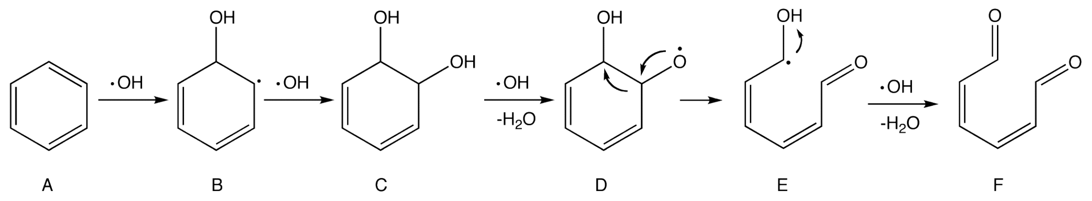


Figure 1.4 OH radical attack on benzene

#### 1.2.3.2 PFOS

Perfluorooctane sulfonate (PFOS), like dixoane, is another persistent environmental pollutant (structure shown in fig. 1.5 [62]) and is a member of the perfluorinated compound (PFC) class that is notoriously difficult to degrade. PFOS is an essential chemical for photolithographic processes and thus is extensively used in the semiconductor industry; it has no known substitutes. [63] It is similarly resistant to conventional wastewater treatment techniques and is in addition resistant to AOTs because of its slow rate of reaction with hydroyxl radicals. [64] Perfluorochemicals like PFOS can theoretically be treated using techniques like activated carbon, nanofiltration, and reverse osmosis; however, the effectiveness of these treatments on wastewater can be significantly impaired by other contaminants in the water matrix. [63, 65] Tang et. al. observed that adding isopropyl alcohol, another chemical used ubiquitously in the semiconductor industry, to solution had a detrimental effect on reverse osmosis treatment of PFOS. [63]

A 2008 study showed that PFOS levels of 90 parts per billion (equivalent to 90 $\mu$g/L for aqueous solutions) compromised immune systems in male mice. [66] The study concludes by saying that human immune systems could be compromised by similar PFOS levels. That $\mu$g/L levels of PFOS may cause negative health effects presents unique problems for degradation researchers. It is common for studies to use tens of mg/L initial contaminant concentrations when testing degradation techniques. In general treatment efficacy will be lower at lower contaminant concentrations. Another complication with PFC removal is its resistance to attack from hydroxyl radicals. Degradation techniques that show promise include photochemical decomposition of persulfate ion followed by oxidation [67], reduction using zerovalent iron [68], and acoustic cavitation that is speculated to break apart PFCs through pyrolysis. [69, 64] These techniques will be compared against plasma degradation in section 5.2.

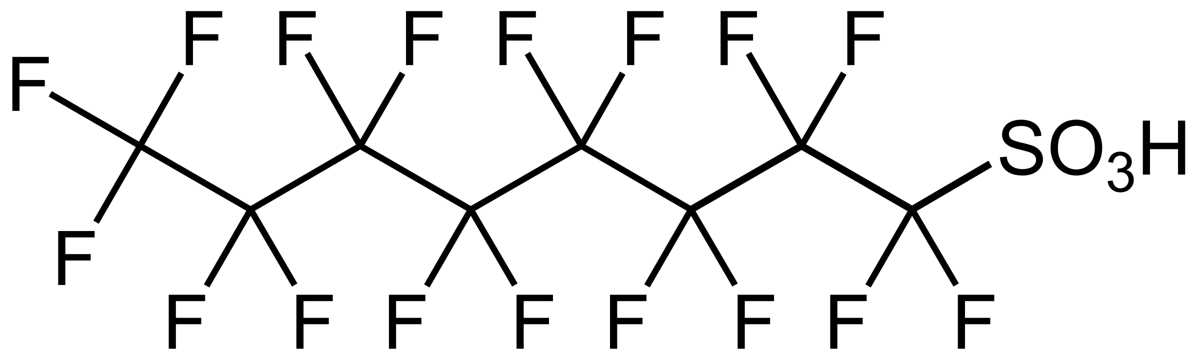


Figure 1.5 Chemical structure of PFOS

### 1.2.4 Plasma Medicine

In sections 1.2.2 and 1.2.3 we highlighted the literature related to applications explored in our laboratory. However, there are many other applications of plasma-liquids. One of the most promising applications is in plasma medicine. A recent review article ([70]) highlighted the role of reactive oxygen and nitrogen species (RONS), species efficiently generated by atmospheric plasmas, in actions of antimicrobial and anti-parasite drugs, cancer therapies, wound healing therapies, and therapies involving the cardiovascular system. Many of the RONS-relevant medical fields described in the review article are being actively explored by plasma researchers. In [71] the authors describe antimicrobial treatment of heat-sensitive materials using miniaturized atmospheric pressure plasma jets. Keidar et. al. [72] performed *in-vitro* and *in-vivo* studies of cold atmospheric plasma (CAP) action on cancer cells. They found that CAP selectively eradicates cancer cells *in-vitro* without damaging normal cells; additionally, they were able to significantly reduce tumor size *in-vivo*. In [73] Nastuta et. al. showed accelerated re-epithelization of burn wounds in rats by plasma treatment. Potential mechanisms of CAP therapies are described in [74]. The author speculates that CAP triggers a beneficial shielding response in tissue by creating a time- and space-localized oxy-nitrosative stress on near-surface celles through plasma-generated RONS. Surface cell layers then communicate to deeper tissue levels through a form of the “bystander effect.” In this way the plasma stimulates a natural cellular survival mechanism by which organisms protect themselves from infections and other malignant agents. [74] As plasma-related biochemical models have improved, the important biological role of plasma-generated RONS has become increasingly apparent. It is thus critical to effectively and accurately model RONS generation in order to optimize delivery of RONS to treatment substrates and increase efficacy of plasma-medicine.

## 1.3 Dissertation Outline

This disseration is laid out in the following way. Chapter 2 outlines modelling work investigating fundamental basic science questions in plasma-liquid systems. Section 2.1 addresses the questions of item 1 above; section 2.2 addresses item 2. Chapter 3 describes the functionality of our code Zapdos, the need for which was outlined in section 1.1. Additionally, a description of the code added to the MOOSE framework itself allowing coupling of theplasma and liquid domains is given in section 3.2. A description of the various experimental configurations explored for optimizing plasma-liquid interactions is given in chapter 4. Finally, our research on applications of plasma-liquid systems, including fertigation and remediation of wastewater contaminants like 1,4-dioxane and PFOS, is presented in chapter 5. Concluding remarks and a roadmap for future research are provided in chapter 6.

CHAPTER

# 2 BASIC SCIENCE OF PLASMA-LIQUID SYSTEMS

Chapter 1 outlined a few fundamental questions in plasma-liquids that had yet to be explored. These included:

- What is the role of convection in transport processes between plasma and liquid phases for discharges like coronas and jets?
- How does varying interfacial parameters like the electron surface loss coefficient affect important plasma-liquid variables?
- Why does decreasing solution conductivity increase the spreading of discharges over the liquid surface?

The role of convection is investigated in section 2.1. Influence of interfacial parameters is considered in section 2.2. While not explored in this dissertation, section 2.2 lays much needed groundwork for exploring the relationship between solution conductivity and discharge spreading.

## 2.1 Momentum, Heat, and Neutral Mass Transport

For a published version of much of the transport modelling work described in section 2.1, the author encourages the reader to navigate to [75].

### 2.1.1 Model Description

The convective system chosen for modelling is shown in fig. 2.1. It is essentially a point-to-plane pulsed-streamer in which the liquid surface serves as the gas discharge cathode. The streamer is self-pulsed because of a ballasting resistor; typical discharge voltages are between 6 and 8 kV and pulse frequencies are generally 10 to 30 kHz. The sharp anode tip can be stationed anywhere between 3 and 15 mm above the water surface. The water is contained within a glass petri dish of radius 3 cm; water treatment volumes are generally between 10 and 20 mL.

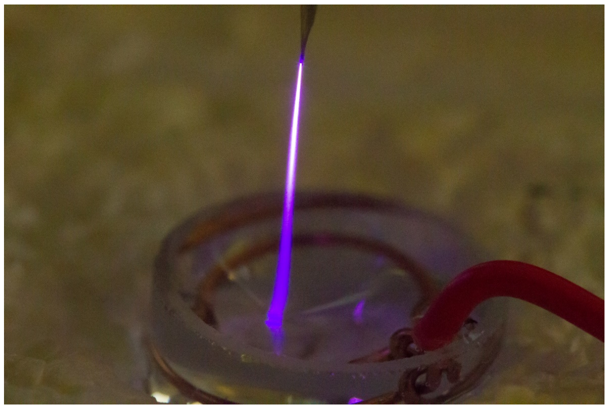


Figure 2.1 Experimental set-up for pulsed streamer-liquid system. Discharge voltages are typically between 6 and 8 kV with pulse frequencies between 10 and 30 kHz. The gap from the needle tip to the water surface is between 3 and 15 mm. Water treatment volumes are between 10 and 20 mL.

For the work in section 2.1, we have chosen not to explicitly model the atmospheric discharge due to the computational cost of modeling of gas-liquid interfacial transport over time-scales of minutes (the equivalent of millions of individual pulses) using Comsol's seemingly inefficient plasma module. Instead, gas phase concentrations of the species modeled are taken from the DBD-water calculations in [1] and diluted by a factor of 10; the gas phase species included and their respective inlet concentrations are shown in table 2.5. The factor 10 dilution is done for numerical stability. The potential effects of the dilution are discussed in section 2.1.2. High concentrations of reactive oxygen and nitrogen species (RONS) at the interface combined with very fast rate coefficients, particularly for reactions involving OH, can lead to singularities over our time scales of interest unless the computational mesh is sufficiently refined, requiring significant computational power. All concentration results and discussion will be focused on relative magnitudes and variation with space and time. The results in [1] are chosen as an input basis because of the absence of experimental work for atmospheric plasmas in which a suite of gas-phase plasma species concentrations are reported. Most of the measurements in the literature focus on single species like OH [76, 77, 78, 79], atomic oxygen [80], or NO [81] that are accessible through Laser Induced Fluorescence (LIF) or resonant absorption spectroscopy. [82] Some of our own absorption measurements of hydroxyl are discussed in section 4.5.3.

A key feature of streamer discharges is the ionic wind. The ionic wind is a net flow of gas towards the cathode that results from the effective drag of cations on neutral gas molecules. In the streamer-liquid system studied, the ionic wind creates convective forces that affect species transport at the gas-liquid interface. Since the discharge is not modelled, another mechanism must be used to produce the ionic wind present in a streamer discharge. The mechanism chosen is a jet-like inlet with diameter equal to the diameter of the needle used in the experimental streamer set-up. The velocity profile at the inlet of the jet flow development channel is chosen such that the channel exit profile closely mimics that expected from streamer experiments and modeling. [83] The profile is built from the expected streamer maximum axial velocity which is in turn determined from the work of Zhao et. al. Using values of 6 kV for the discharge voltage and 6.5 mm for the gap distance, the maximum axial velocity is interpolated to be 7.75 m/s near the needle tip. The spatial inlet profile is then computed from eq. (2.1) assuming laminar no-slip conditions in the inlet channel: [84]

$$v_z = v_{z,max}\left(1 - \left(\frac{r}{R}\right)^2\right) \tag{2.1}$$

The model geometry is shown in fig. 2.2; model inputs are summarized in tables 2.1 to 2.5. The governing equations used to model the gas and liquid phases are the incompressible Navier-Stokes equations for momentum transport and convection-diffusion equations for heat and mass transport; they are shown in eqs. (2.2) to (2.5)

$$\nabla \cdot \vec{u} = 0 \tag{2.2}$$

$$\rho \left( \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \right) = -\nabla p + \mu \nabla^2 \vec{u} \tag{2.3}$$

$$\rho C_p \vec{u} \cdot \nabla T = \nabla \cdot (k \nabla T) \tag{2.4}$$

$$\nabla \cdot (-D_i \nabla C_i) + \vec{u} \cdot \nabla C_i = R_i \tag{2.5}$$

with $\vec{u}$ representing the fluid velocity, $\rho$ the overall mass density, p the static pressure, $\mu$ the dynamic viscosity, $C_p$ the constant pressure heat capacity, T the temperature, k the thermal conductivity, $D_i$ the diffusivity of species i, $C_i$ the concentration of species i, and $R_i$ the source term representing chemical reactions. When solving the model equations, the transient equations 2.2 and 2.3 are solved until the velocity components and pressure reach a steady-state. Then the steady-state velocity field is inserted into the convection terms in equations 2.4 and 2.5. The heat and mass transport equations are typically solved for a physical time of 1000 seconds in order to match common experimental treatment times in our laboratory. The temperature dependence of transport parameters in equations 2.4 and 2.5, including reaction rate coefficients, is considered. Gaseous diffusion coefficients are assumed to scale with $T^{3/2}$ as predicted by Chapman-Enskog theory. [85, p. 119] The temperature dependence of liquid phase diffusion coefficients is constructed using the Stokes-Einstein equation [84, p. 529] and the following equation for viscosity: [84, p. 31]

$$\mu \propto exp(3.8 T_b / T) \tag{2.6}$$

where $T_b$ is the boiling point of the solvent (373 K in the case of water). Reaction rate coefficients and their temperature dependence are contained in table 2.6. Because the fluid flow equations are solved prior to solution of the heat transport equation, the temperature dependence of transport parameters in equations 2.2 and 2.3 is not included in this work. Model boundary conditions are as follows: for fluid flow, all solid walls are assumed to be no-slip, i.e. all velocity components at the walls are zero. For most results presented the gas-liquid interface is assumed to be flat and static. For the static interface, the z-velocity component at the interface is set to zero in both the gas and liquid phases. Both the normal and shear stresses are continous across the interface. One may question whether the assumption of a flat interface is valid when in reality the interface is deformed by the gas flow. In the results section it will be demonstrated that for the purpose of this work, an assumption of a flat interface is sufficient. For temperature calculations, all solid surfaces are assumed to be insulated; at the interface, the temperature is assumed continuous. Additionally, evaporation of water at the surface is coupled to the system's heat transport in the following way: [84]

$$Q_b = J_{z,H_2O} \cdot H_{vap} = -D_{H_2O,g} \cdot \left.\frac{\partial C_{H_2O(g)}}{\partial z}\right|_{z=interface} \cdot H_{vap} \tag{2.7}$$

where $Q_b$ is the heat flux, $J_{z,H_2O}$ is the molar flux of $H_2O$ coming from evaporation, and $H_{vap}$ is the latent heat of vaporization for water. Equation (2.7) is not strictly true during transient dynamics; however, the approximation is good enough for our purposes. The concentration of water vapor at the interface is determined from Antoine's equation: [86]

$$\log_{10} p = 8.07131 - \frac{1730.63}{233.426 + T} \tag{2.8}$$

with p in units of mmHg, T in units of degrees Celsius, and the constant values taken from [87]. The temperature at the gas inlet is set to 300 K for all times. Initially the temperature in both gas and liquid domains is 300 K. Inlet concentrations for species other than water vapor are specified at the beginnning of the jet flow development channel and are given in table 2.5. Dilute species that are present in both gas and aqueous phases (OH, $H_2O_2$, NO, $NO_2$, $N_2O_4$, $HNO_2$, and $HNO_3$) have continuous fluxes across the gas-liquid interface. Concentrations immediately above and below the interface are assumed to be in equilibrium as described by their Henry's law coefficients, listed in table 2.3. A limited set of gas and liquid phase reactions encompassing the most important $NO_x$ chemistry is used to reduce computational expense. Even with this simplified reaction set and diluted concentration inputs, the simulation takes multiple days on a quad-core desktop and approaches 16 GB in required memory. A move towards a more parallelizable software framework (see section 2.2 and chapter 3) will enable incorporation of a more complete set of reacting species and reactions. Some important species that were not included in this simplified set include oxygen and hydrogen radicals as well as water cluster ions to name a few. [88] The reactions used in this study and their corresponding rate coefficients are listed in table 2.6. It should be noted that because of its highly acidic nature (pKa  -1.3), $HNO_3$ is assumed to dissociate into $H^+$ and $NO_3^-$ immediately after entering the aqueous phase. The model equations are solved using the finite element method implemented in Comsol Multiphysics version 4.4.[1]

When the “discharge region" is discussed, it is in reference to the gas region roughly demarcated in the z direction by the jet outlet/needle anode and the water surface, and in the radial direction by the radius of the jet channel/needle, where the plasma is visible during experiments. The reader is reminded that the plasma and its electrodynamics are not explicitly modeled in section 2.1; the focus of this investigation is the qualitative behavior of momentum, heat, and neutral species mass transport in convective systems. The qualitative conclusions drawn here are equally applicable to atmospheric jets or streamer systems in which the ionic wind plays a key role in momentum transport. Moreover, we postulate that the steep gradients in highly reactive neutral species concentrations at the gas-liquid interface shown in the proceeding section are a universal phenomena of atmospheric plasma-liquid systems, regardless of whether they are convective or diffusive; this is consistent with the recent research by [22] as well as the greater reservoir of biochemistry literature [89]. Morever, these gradients have important implications for plasma medicine, mainly that cellular responses must be induced through secondary as opposed to directly generated plasma reactivity.

Table 2.1 Species included in the model

| | |
|---|---|
| Gas phase species | OH, $H_2O_2$, NO, $NO_2$, $N_2O_4$, $HNO_2$, $HNO_3$, $H_2O$ |
| Liquid phase species | OH, $H_2O_2$, NO, $NO_2$, $N_2O_4$, $HNO_2$, $NO_2^-$, $NO_3^-$, ONOOH, $H^+$, $OH^-$ |

Table 2.2 General model inputs

| | |
|---|---|
| Needle diamater | 1.2 mm |
| Petri dish diameter | 6 cm |
| Gap distance | 6.5 mm |
| Water volume | 10 mL |
| Jet channel length | 2 cm |
| Maximum axial velocity of ionic wind/jet | 7.75 m/s |

[1]A copy of the model is freely available upon request.

Table 2.3 The Henry's constant for a number of molecules[1].

| Molecule | $H_i$ [unitless] |
|---|---|
| OH | $6.92 \cdot 10^{2}$ |
| $H_2O_2$ | $1.92 \cdot 10^{6}$ |
| NO | $4.4 \cdot 10^{-2}$ |
| $NO_2$ | $2.8 \cdot 10^{-1}$ |
| $N_2O_4$ | $3.69 \cdot 10^{1}$ |
| $HNO_2$ | $1.15 \cdot 10^{3}$ |
| $HNO_3$ | $4.8 \cdot 10^{6}$ |
| HOONO | $4.8 \cdot 10^{6}$ |

Table 2.4 The diffusion coefficients for a number of molecules at 300 K. See text for implementation of temperature dependence.

| Molecule | D [$m^2$ $s^{-1}$] | reference |
|---|---|---|
| OH(g) | $4 \cdot 10^{-5}$ | [90] |
| $H_2O_2$(g) | $2 \cdot 10^{-5}$ | [90] |
| NO(g) | $2 \cdot 10^{-5}$ | [90] |
| $NO_2$(g) | $1.7 \cdot 10^{-5}$ | [90] |
| $N_2O_4$(g) | $1 \cdot 10^{-5}$ | [90] |
| $HNO_2$(g) | $2.1 \cdot 10^{-5}$ | [90] |
| $HNO_3$(g) | $2.1 \cdot 10^{-5}$ | [90] |
| $H_2O$(g) | $2.3 \cdot 10^{-5}$ | [90] |
| OH(aq) | $2.8 \cdot 10^{-9}[310K]$ | [91] |
| $H_2O_2$(aq) | $1.7 \cdot 10^{-9}$ | [92] |
| NO(aq) | $2.2 \cdot 10^{-9}$ | [93] |
| $NO_2$(aq) | $1.85 \cdot 10^{-9}[296K]$ | [94] |
| $N_2O_4$(aq) | $1.5 \cdot 10^{-9}$ | Estimate |
| $HNO_2$(aq) | $2.5 \cdot 10^{-9}$ | By analogy with nitric acid |
| $HNO_3$(aq) | $2.5 \cdot 10^{-9}$ | [95] |
| ONOOH(aq) | $2.5 \cdot 10^{-9}$ | By analogy with nitric acid |
| $NO_2^-$(aq) | $1.7 \cdot 10^{-9}$ | [96] |
| $NO_3^-$(aq) | $1.7 \cdot 10^{-9}$ | [96] |
| $H^+$(aq) | $7 \cdot 10^{-9}$ | [97] |
| $OH^-$(aq) | $5.29 \cdot 10^{-9}$ | [98] |

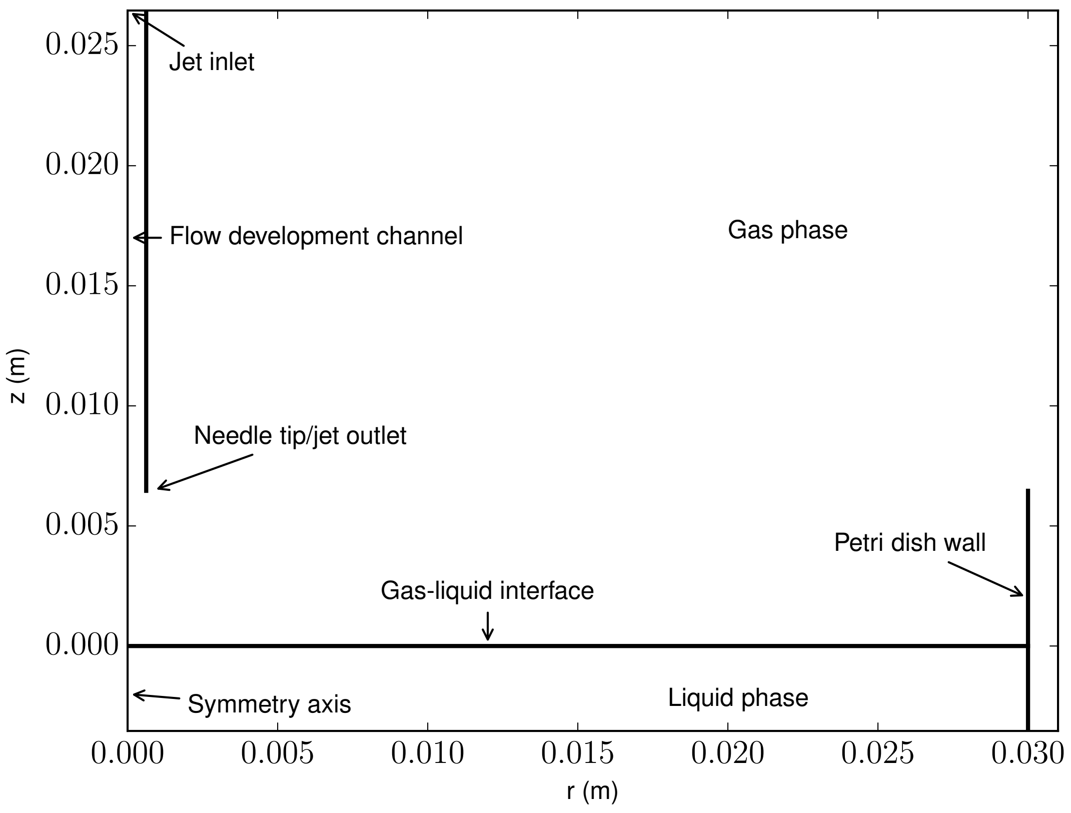


Figure 2.2 Experimental set-up for pulsed streamer-liquid system. Axis units are meters. The Python script used to create this figure, as well as the scripts used to create all subsequent figures can be found at [3]

Table 2.6 Reactions considered in model

| **Reaction** | **Rate coefficient** (Units of $s^{-1}$, $m^3$ $mol^{-1}$ $s^{-1}$, or $m^6$ $mol^{-2}$ $s^{-1}$. Temperature in K. Concentration of M in gas and $H_2O$ in liquid are lumped into rate coefficient) | **Reference** |
|---|---|---|
| Gas phase reactions | | |

Table 2.6 Continued

| **Reaction** | **Rate coefficient** (Units of $s^{-1}$, $m^3$ $mol^{-1}$ $s^{-1}$, or $m^6$ $mol^{-2}$ $s^{-1}$. Temperature in K. Concentration of M in gas and $H_2O$ in liquid are lumped into rate coefficient) | **Reference** |
|---|---|---|
| 1. $2NO_2 \rightarrow N_2O_4$ | $6.02 \cdot 10^5 \cdot (300/T)^{3.8}$ | [90] |
| 2. $N_2O_4 \rightarrow 2NO_2$ | $4.4 \cdot 10^6 \cdot exp(-4952/T)$ | [90] |
| 3. $NO + OH + M \rightarrow HNO_2 + M$ | $1.1 \cdot 10^7 \cdot (300/T)^{2.4}$ | [90] |
| 4. $NO + NO_2 + H_2O \rightarrow 2HNO_2$ | 22 | [99] |
| 5. $2OH + M \rightarrow H_2O_2 + M$ | $1.0 \cdot 10^7 \cdot (T/300)^{-0.8}$ | [90] |
| 6. $NO_2 + OH + M \rightarrow HNO_3 + M$ | $3.2 \cdot 10^7 \cdot (300/T)^{2.9}$ | [90] |
| 7. $OH + HNO_2 \rightarrow NO_2 + H_2O$ | $3.0 \cdot 10^6 \cdot exp(-390/T)$ | [90] |
| 8. $2HNO_2 \rightarrow NO + NO_2 + H_2O$ | $6.0 \cdot 10^{-3}$ | [90] |
| 9. $HNO_2 + HNO_3 \rightarrow 2NO_2 + H_2O$ | 9.6 | [90] |
| 10. $HNO_3 + NO \rightarrow HNO_2 + NO_2$ | $4.4 \cdot 10^{-3}$ | [100] |
| Liquid phase reactions | | |
| 11. $N_2O_4 + H_2O \rightarrow NO_2^- + NO_3^- + 2H^+$ | 1000 | [101] |
| 12. $2NO_2 + H_2O \rightarrow NO_2^- + NO_3^- + 2H^+$ | $8.4 \cdot 10^4 \cdot exp(-0.033 \cdot (T-295))$ | [102] |
| 13. $NO + NO_2 + H_2O \rightarrow 2NO_2^- + 2H^+$ | $1.6 \cdot 10^5$ | [102] |
| 14. $NO_2^- + H_2O_2 + H^+ \rightarrow ONOOH + H_2O$ | $1.1 \cdot 10^{-3}$ | [16] |
| 15. $ONOOH \rightarrow .7(NO_3^- + H^+) + .3(NO_2 + OH)$ | .8 | [101] |
| 16. $NO_2 + OH \rightarrow ONOOH$ | $5.3 \cdot 10^6$ | [16, 103] |
| 17. $2OH + M \rightarrow H_2O_2 + M$ | $1 \cdot 10^7 \cdot exp(-450 \cdot (1/T - 1/298))$ | [104] |

Table 2.6 Continued

| **Reaction** | **Rate coefficient** (Units of $s^{-1}$, $m^3$ $mol^{-1}$ $s^{-1}$, or $m^6$ $mol^{-2}$ $s^{-1}$. Temperature in K. Concentration of M in gas and $H_2O$ in liquid are lumped into rate coefficient) | **Reference** |
|---|---|---|
| 18. $HNO_2 \rightarrow H^+ + NO_2^-$ | $3.51 \cdot 10^6$ | From reaction 19 and equilibrium constant |
| 19. $H^+ + NO_2^- \rightarrow HNO_2$ | $7 \cdot 10^6$ | Assumes diffusion limited |
| 20. $H^+ + OH^- \rightarrow H_2O$ | $7 \cdot 10^6$ | Assumes diffusion limited |
| 21. $H_2O \rightarrow H^+ + OH^-$ | $7 \cdot 10^{-2}$ | From reaction 20 and equilibrium constant |
| 22. $NO + OH \rightarrow HNO_2$ | $2 \cdot 10^7$ | [1] |
| 23. $2HNO_2 \rightarrow NO + NO_2$ | $1.34 \cdot 10^{-2} \cdot exp(.106 \cdot (T - 295))$ | [102] |
| Tyrosine reactions | | |
| 24. $NO_2$ + Tyr $\rightarrow NO_2^-$ + Tyr(radical) | $2.90 \cdot 10^4$ | [105] |
| 25. Tyr(radical) + NO $\rightarrow$ Tyr-NO | $1.00 \cdot 10^6$ | [105] |
| 26. Tyr-NO $\rightarrow$ Products | 2 | [105] |
| 27. Tyr-NO $\rightarrow$ Tyr(radical) + NO | $1 \cdot 10^3$ | [105] |
| 28. Tyr(radical) + Tyr(radical) $\rightarrow$ diTyr | $2.25 \cdot 10^5$ | [105] |
| 29. $NO_2$ + Tyr(radical) $\rightarrow$ Tyr-$NO_2$ | $1.30 \cdot 10^6$ | [105] |

Table 2.6 Continued

| **Reaction** | **Rate coefficient** (Units of $s^{-1}$, $m^3$ $mol^{-1}$ $s^{-1}$, or $m^6$ $mol^{-2}$ $s^{-1}$. Temperature in K. Concentration of M in gas and $H_2O$ in liquid are lumped into rate coefficient) | **Reference** |
|---|---|---|
| 30. $NO_2$ + Tyr(radical) → Other products | $1.70 \cdot 10^6$ | [105] |
| 31. Tyr + OH → TyrOHo | $7 \cdot 10^6$ | [106] |
| 32. Tyr + OH → TyrOHm | $5 \cdot 10^6$ | [106] |
| 33. Tyr + OH → Tyr(radical) | $6 \cdot 10^5$ | [106] |
| 34. TyrOHm + TyrOHm → products | $1 \cdot 10^6$ | [106] |
| 35. TyrOHo + TyrOHo → products | $1.5 \cdot 10^5$ | [106] |
| 36. TyrOHo → Tyr(radical) + $H_2O$ | $1.8 \cdot 10^4$ | [106] |

### 2.1.2 Results and Discussion

The steady state velocity field is shown in figure 2.3. The recirculating pattern observed in the gas phase is consistent with the corona discharge modelling results reported in [83]. A similar recirculation pattern is observed in the liquid phase, induced by shear stresses between the phases at the gas-liquid interface. The maximum liquid phase velocity, .1 m/s, occurs along the interface approximately .8 mm away from the stagnation point ($r = 0$).

The temperature and water vapor profiles at the end of the simulation ($t = 1000$ seconds) are shown in figures 2.4 and 2.5. Notably, the temperature in the bulk liquid has fallen close to 10 K from its initial value of 300 K. This drop in temperature in the bulk liquid is an example of the classical wet-bulb/dry-bulb problem found in chemical engineering texts. Water vapor present in the gas above the interface is whisked away by convection. In order to maintain the equilibrium vapor pressure required by Antoine's equation, liquid water must be evaporated, consuming heat

Table 2.5 Gaseous species inlet concentrations. [1] See text for discussion

| Molecule | Molecule inlet concentration [$m^{-3}$] |
|---|---|
| $OH$ | $1.3 \cdot 10^{18}$ |
| $H_2O_2$ | $1.6 \cdot 10^{17}$ |
| $NO$ | $8 \cdot 10^{18}$ |
| $NO_2$ | $5 \cdot 10^{16}$ |
| $N_2O_4$ | 0 |
| $HNO_2$ | $8 \cdot 10^{17}$ |
| $HNO_3$ | $9 \cdot 10^{16}$ |
| $H_2O$ | 0 |

and lowering the temperature at the interface. Heat then migrates from the bulk liquid to the surface, leading to bulk liquid cooling. In this problem we have assumed that the impinging gas is at room temperature, and subsequently convection-induced evaporation leads to cooling of the bulk liquid below room ambient. However, if there is significant plasma heating of the gas, it is possible that heating of the liquid above the initial temperature will be observed experimentally. The important point is that the natural coupling between heat transport and evaporation leads to significant spatial variation in temperature, particularly at the interface, and a cooling of the bulk liquid relative to the gas. An example of the steep temperature gradients is shown in figure 2.6 where the gas phase temperature changes by 10 K over the span of 200 $\mu$m, immediately above the liquid surface. Because reaction rates typically exhibit Arrhenius dependence on temperature, physical factors that introduce steep temperature gradients should be included in any model that wishes to accurately predict plasma-liquid chemistry. Though not shown here, a simulation was conducted in which temperature and water-vapor transport were de-coupled and the interfacial temperature gradient removed. Compared to the coupled case, concentrations of long-lived aqueous species like $H_2O_2$, $NO_2^-$, and $NO_3^-$ differed by as large as factors of two at the end of the simulation, demonstrating the importance of accounting for evaporation-induced temperature gradients.

Another instance of transport coupling that may impact plasma-liquid chemistry is the significant radial gradients in the concentration of water vapor between the needle tip/jet outlet and gas-liquid interface. An example of these gradients induced by convective forces are shown in figure 2.7. Examining the dotted curve, which gives the radial distribution of water vapor half-way between the jet outlet/needle tip and liquid surface, one can see that the water vapor concentration drops precipitously in the region of the discharge (r < 2 mm). In the center

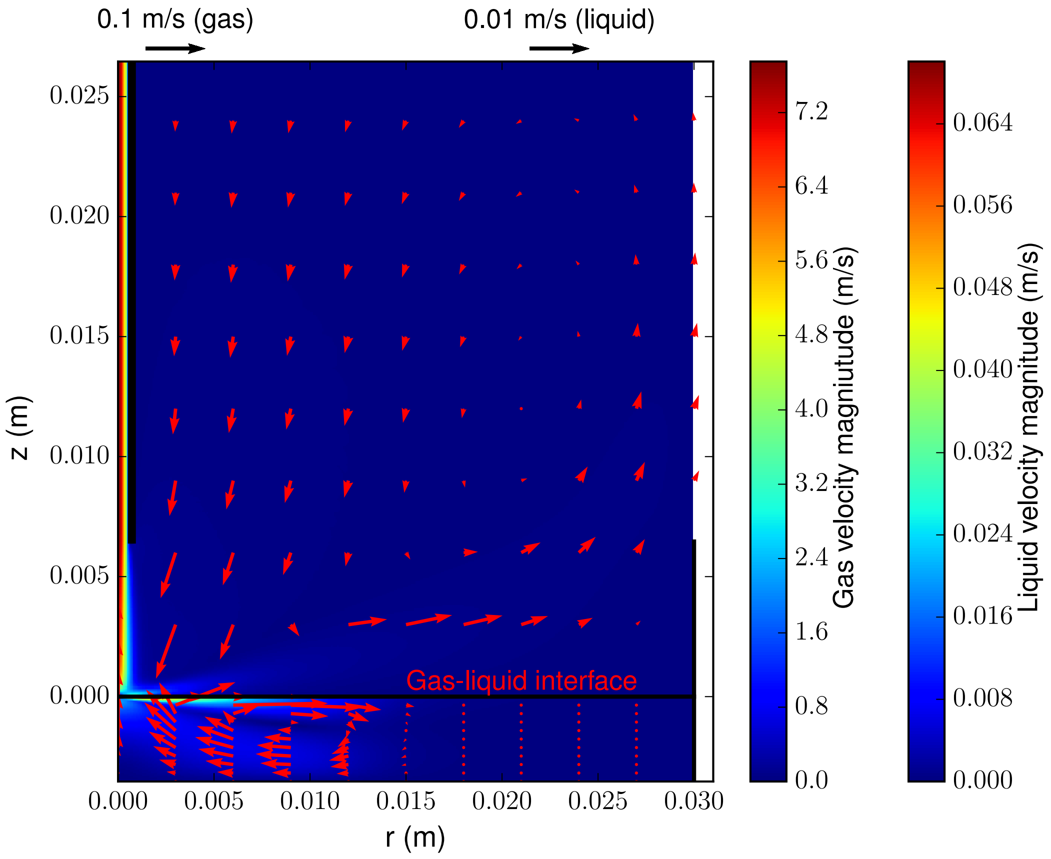


Figure 2.3 Velocity magnitude and direction. Axis units are meters. Interface is at z = 0. Axis of symmetry is at r = 0. Selected velocity vectors are overlaid on color scale indicating velocity magnitude. Red color represents higher values and blue color represents lower values. Velocity vector arrows are scaled 12.5 times larger in the aqueous phase.

of the discharge (r = 0) the water vapor concentration is essentially zero. This large drop in the concentration of water vapor in the active discharge region due to convection suggests that the concentrations of plasma species which rely on $H_2O$ as a precursor will be reduced at increasing distances from the liquid surface, relative to discharges where diffusion is the dominant mechanism of mass transport.

This model also addresses the role that convection plays in dissolution rates of different gaseous species. For this analysis, reactions are turned off, reducing mass transport to only convection and diffusion. As one might intuitively expect, the induced convective flow in the liquid significantly

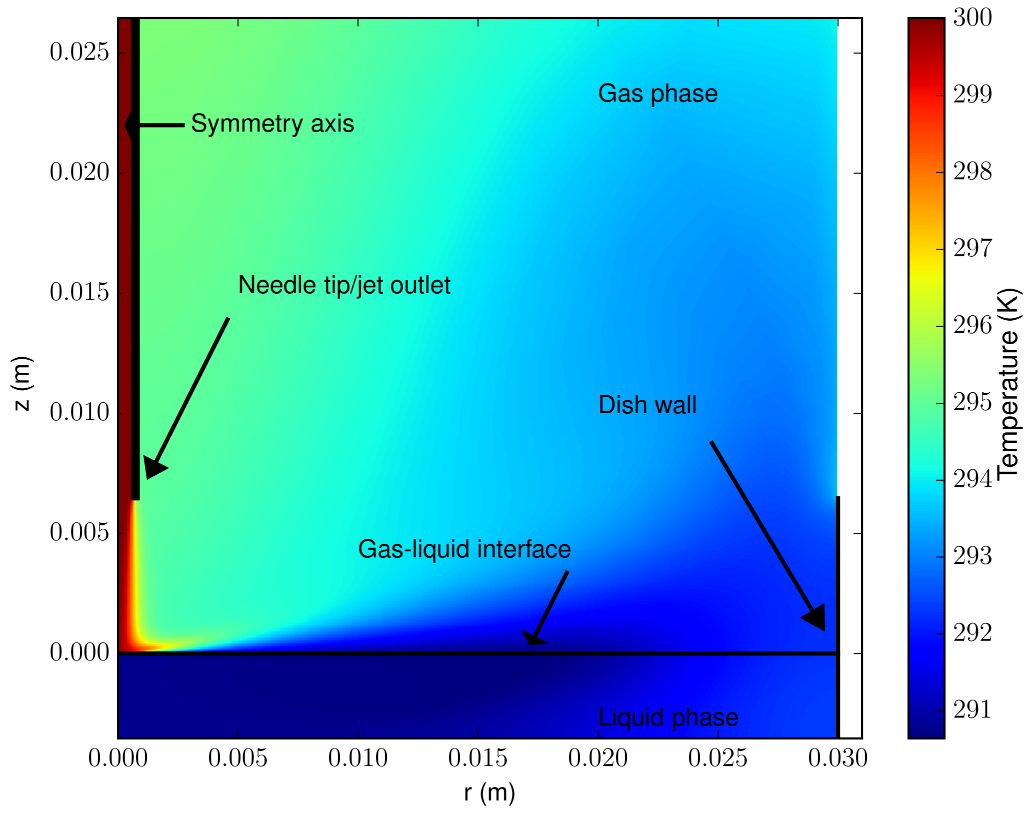


Figure 2.4 2D temperature profile at t = 1000 seconds. Red color represents higher values; blue color lower values. Inlet temperature is 300 K. Temperature in the bulk liquid has cooled by approximately 10 K because of convection-induced evaporative cooling.

changes the spatial distribution of aqueous species relative to a diffusion only case, as demonstrated for $HNO_3$ in figures 2.8a and 2.8b (note that gas convection is present in both figures). However, what is perhaps not intuitive is that though the $HNO_3$ spatial distribution changes dramatically depending on whether convection is present in the liquid, the volume-averaged uptake of $HNO_3$ does not change from diffusion-dominated to convection-dominated cases as shown in figure 2.9a. If a hydrophobic specie like NO is examined instead of a hydrophilic specie like $HNO_3$, it is observed that the presence of liquid convection increases volume-averaged uptake significantly, as illustrated in figure 2.9b. This fundamental difference in behavior between hydrophilic and hydrophobic species can be explained in terms of lumped mass transfer resistances. For a hydrophilic specie, the dominant resistance to interfacial transfer is in the

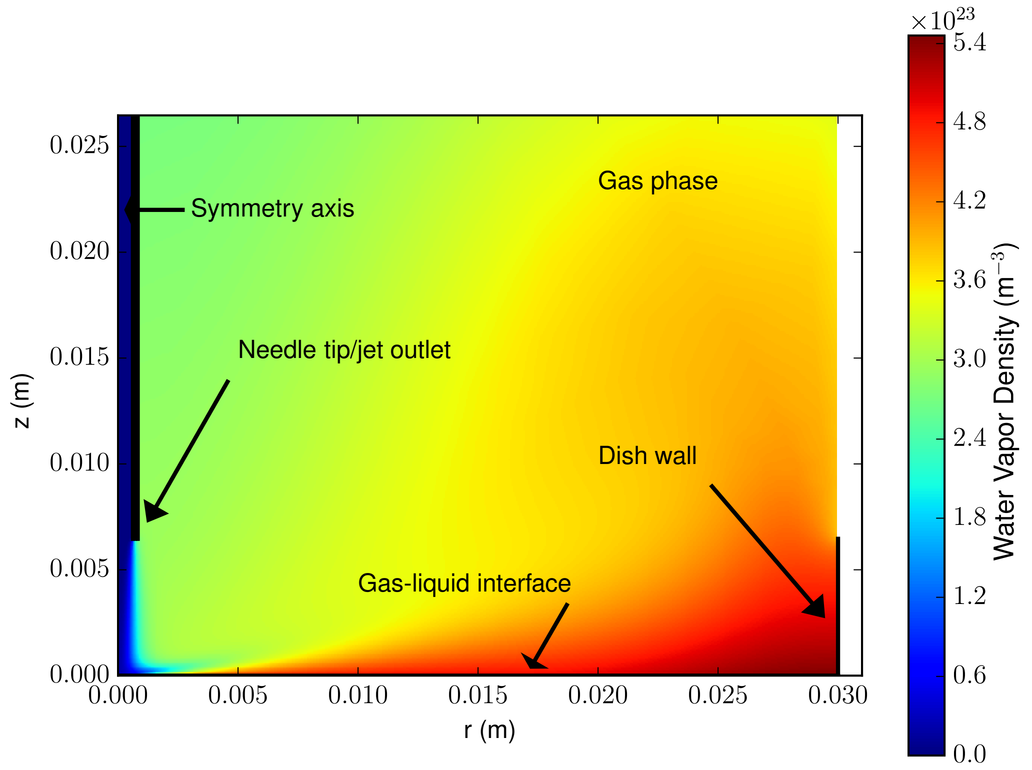


Figure 2.5 2-D water vapor profile at t = 1000 seconds. Red color represents higher values of water vapor concentration; blue color lower values. As implemented through Antoine's equation, water vapor concentration at the interface is highest where the temperature is highest. Role of convection in water vapor profile is evident in decreased concentration near the streamer/jet.

gas-phase, whereas for a hydrophobic specie the dominant resistance to transfer occurs in the liquid phase. [107, p. 249] Consequently, when convection is added to the liquid phase, effectively reducing the liquid-side mass transfer resistance, the overall resistance to mass transfer decreases and the volume-averaged uptake increases significantly for hydrophobic species like NO whereas the change in overall resistance is miniscule for hydrophilic species like $HNO_3$.

When the full reaction set is considered, large gradients in reactive species concentrations emerge at the interface. Of particular interest for biomedical or pollutant degradation applications is the distribution of hydroxyl radical in the aqueous phase. Figure 2.10 shows a 3D plot of the base

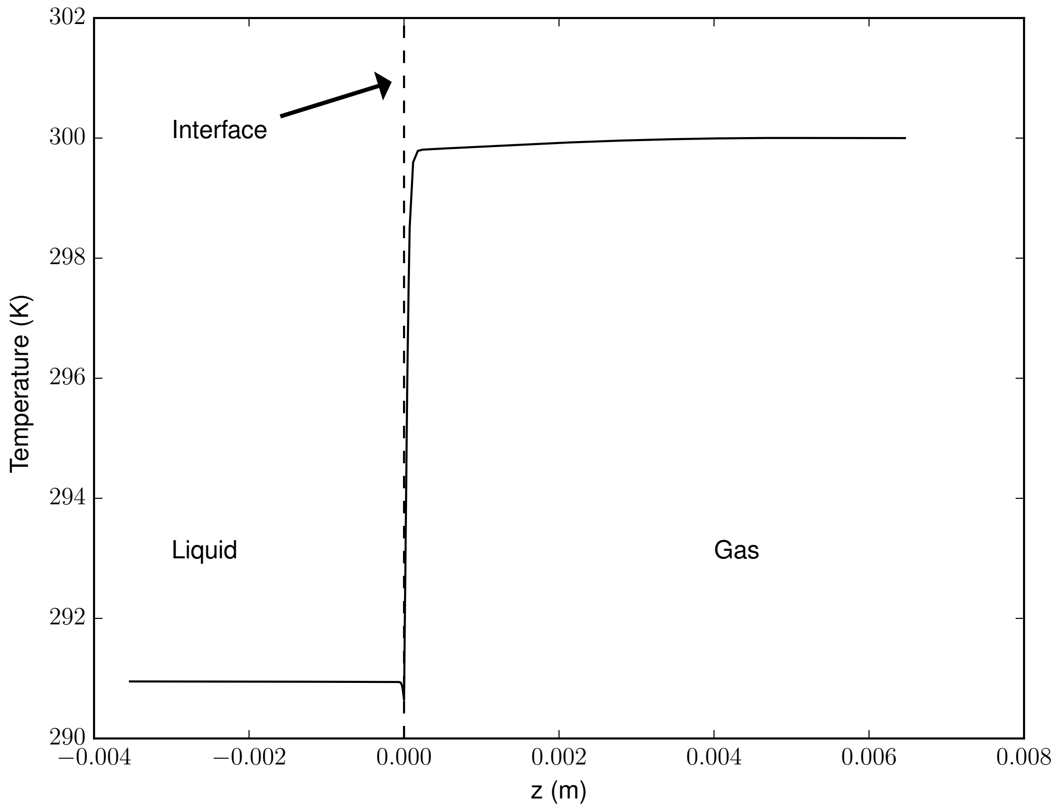


Figure 2.6 Temperature along z-axis. Illustrates the large temperature gradient that exists at the gas-liquid interface and the resulting difference in bulk gas and liquid temperatures.

10 log of OH(aq) concentration versus radial and axial position for t = 1000 seconds. Interfacial gradients in OH concentration are prominent in the region where the discharge impinges on the water surface. At the stagnation point (r=0) the OH concentration drops by approximately 9 orders of magnitude over the span of 50 $\mu$m in the axial direction. Moving away from where the streamer/jet touches the surface, the interfacial gradients become less pronounced. Even so, the largest OH concentration away from the surface and in the bulk solution is 3-4 orders of magnitude lower than the peak OH concentration which occurs at the interface and at the center of the impinging streamer/jet. The presence of the liquid phase convective loop is evident from the OH concentration hole in the center of Figure 2.10. The effect is also seen in the center of Figure 2.11 for ONOOH.

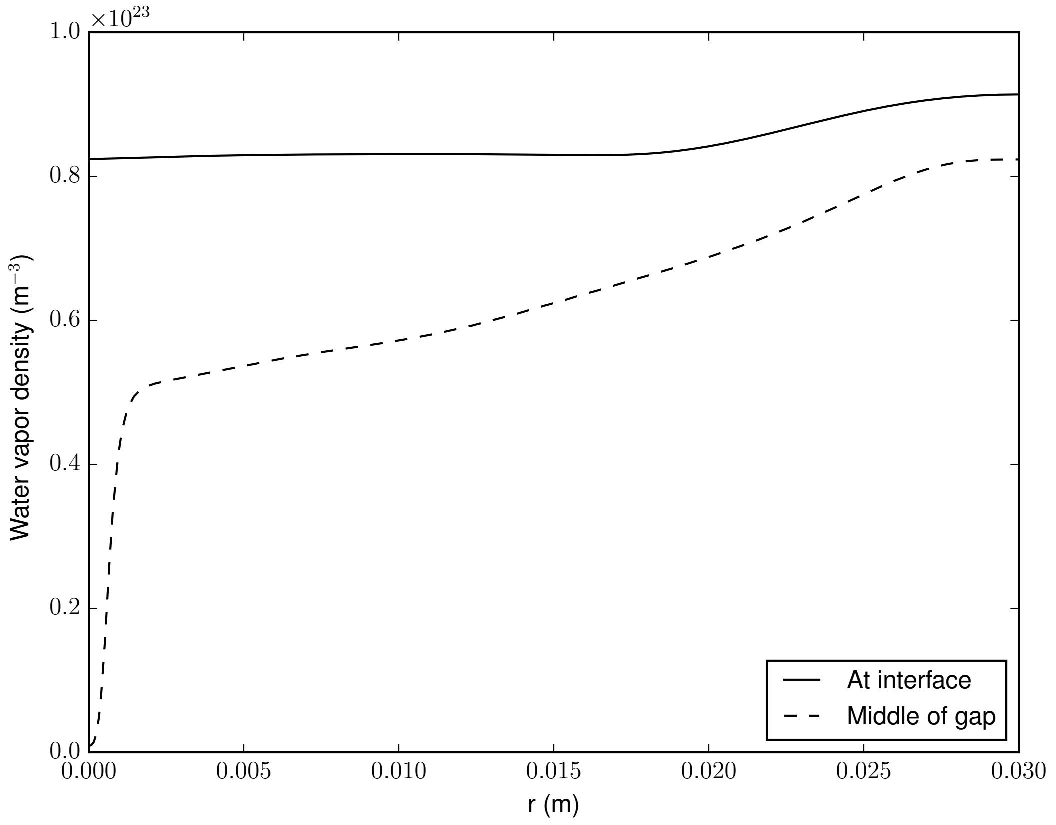
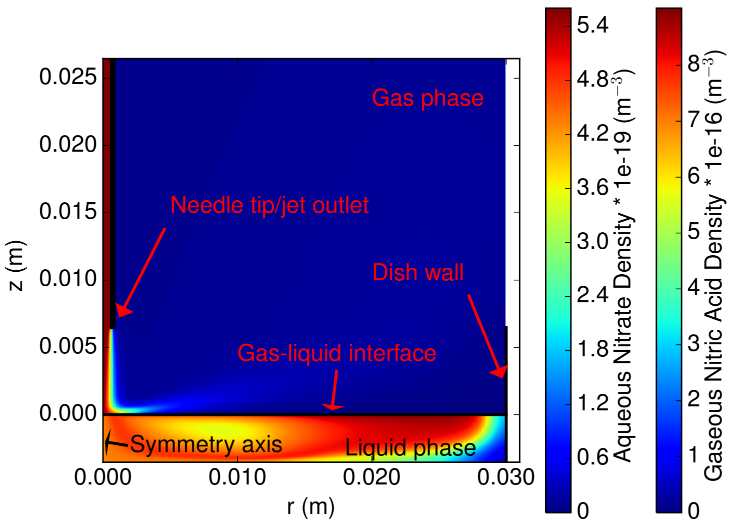
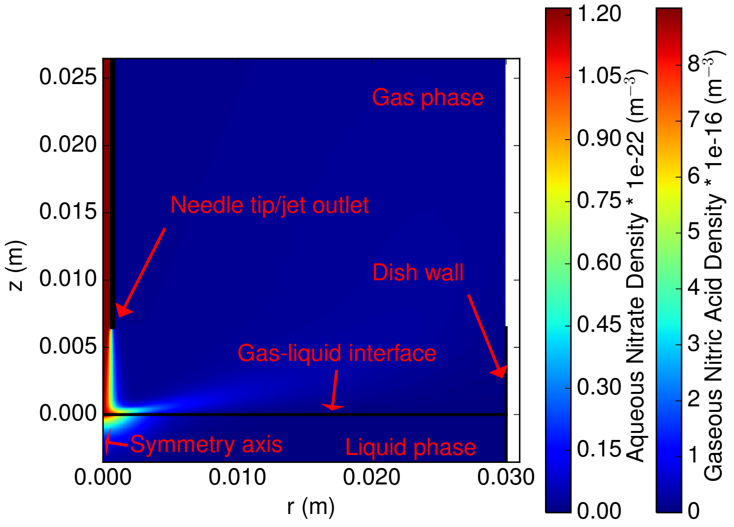


Figure 2.7 Radial water vapor profiles. Horizontal axis is the radial coordinate. The top curve (solid) corresponds to the water vapor concentration at the interface. The bottom curve (dashed) shows the strong water vapor radial dependence in the middle of the streamer/jet gap. Gradient is largest near and inside the discharge region.

Another plasma-generated specie which has received much attention in the literature is peroxynitrous acid (ONOOH) and its conjugate base peroxynitrite ($ONOO^-$). In the current model, ONOOH exists only in the liquid phase and can be created through two mechanisms: reactions 14 and 16 in table 2.6. Reaction 14 was the topic of an excellent study in [16] and is hypothesized to be a key player in the long-term anti-bacterial efficacy of plasma activated water. Figure 2.11 shows the base 10 log of ONOOH concentration as a function of r and z in the aqueous phase. As with OH, there are large interfacial concentration gradients. At r=0, the ONOOH concentration drops by 5 orders of magnitude over 30 $\mu$m in the axial direction. Away from the stagnation point, especially where the convective current flows away from the interface,

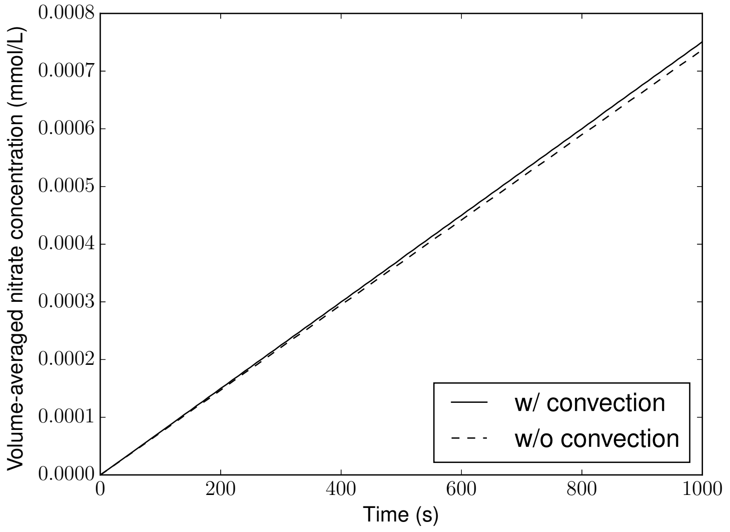


(a) Distribution of $HNO_3$ with liquid convection turned on. t = 1000 s

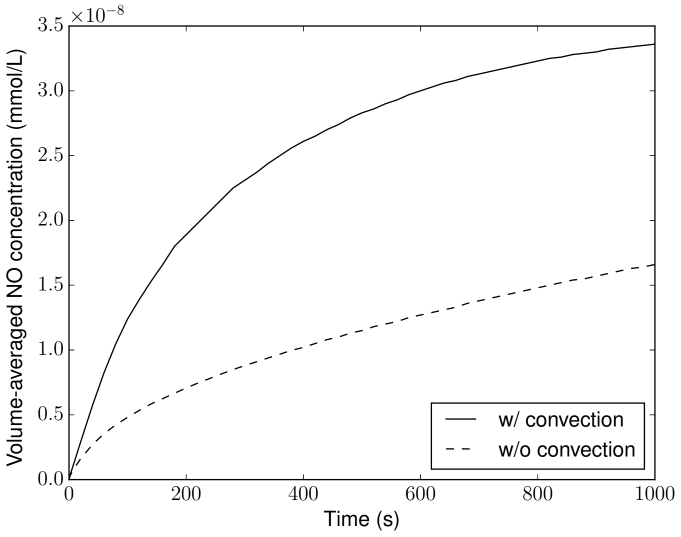


(b) Distribution of $HNO_3$ with liquid convection turned off. t = 1000 s

Figure 2.8 Liquid convection vs. liquid diffusion-only spatial profiles for $HNO_3$



(a) Comparison of volume averaged uptake of nitrate ($HNO_3$ before dissolution) with liquid convection toggled on or off. Very little difference between the two cases.



(b) Comparison of volume averaged uptake of NO with liquid convection toggled on or off. The presence of convection increases the volume-averaged concentration by roughly a factor of 2 over the course of the simulation. Note that the vertical scale is much smaller for NO than for $HNO_3$ in 2.9a because of NO's hydrophobicity

Figure 2.9 Liquid convection vs. liquid diffusion-only volume-averaged uptake of NO

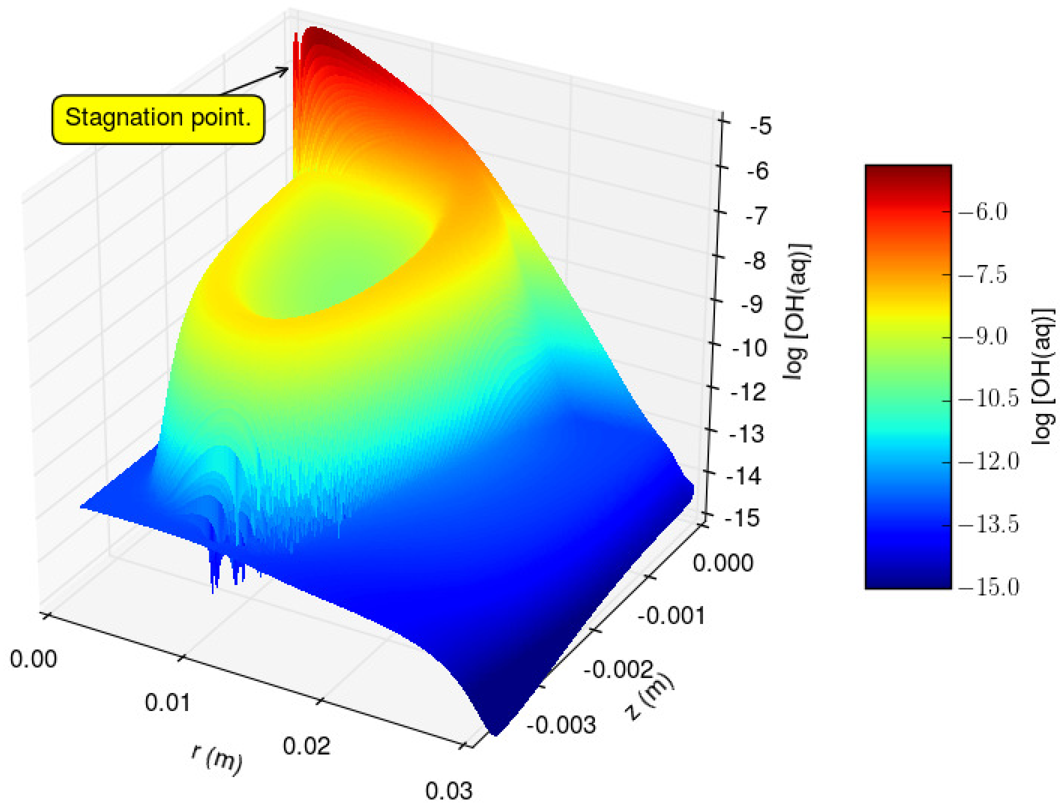


Figure 2.10 3D plot of $\log_{10}(OH(aq))$ as a function of position for t = 1000 seconds. Large interfacial gradients are evident, particularly at the stagnation point (r=z=0). Note the effect of convection in the hole in OH concentration in the center of the plot. Some numerical noise is observed at very low OH concentrations towards the bottom of the dish. Note that the r and z axes have different scales

the interfacial gradient is much smaller but again as with OH, the highest bulk concentration is orders of magnitude lower than the peak surface concentration. The reason for these large ONOOH gradients is the relative dominance of reaction 16 over reaction 14 for the given model inputs. Over the course of the simulation, almost 7000 times more ONOOH is produced through the reaction of OH and $NO_2$ than through the reaction of $H^+$, $H_2O_2$, and $NO_2^-$. As shown in figure 2.10, hydroxyl does not penetrate any more than a few tens of microns into the liquid phase, so consequently all ONOOH produced through OH is produced within a few tens of microns of the liquid surface. If we examine the production of ONOOH through reaction 14, it is observed to be much more uniform as illustrated in figure 2.12. This is because of the relative uniformity in concentration of $H^+$, $NO_2^-$, and $H_2O_2$, although there is a peak in their concentrations and consequently in their production of ONOOH in the vicinity of the impinging streamer/jet. Once the streamer is turned off and surface fluxes of ON and $NO_2$ are removed, production of ONOOH(aq) will proceed almost exclusively through reaction 14, resulting in a mostly homogeneous distribution. This is relevent for applications of PAW since ONOOH is the long-term intermediary for production of OH through reaction 15.

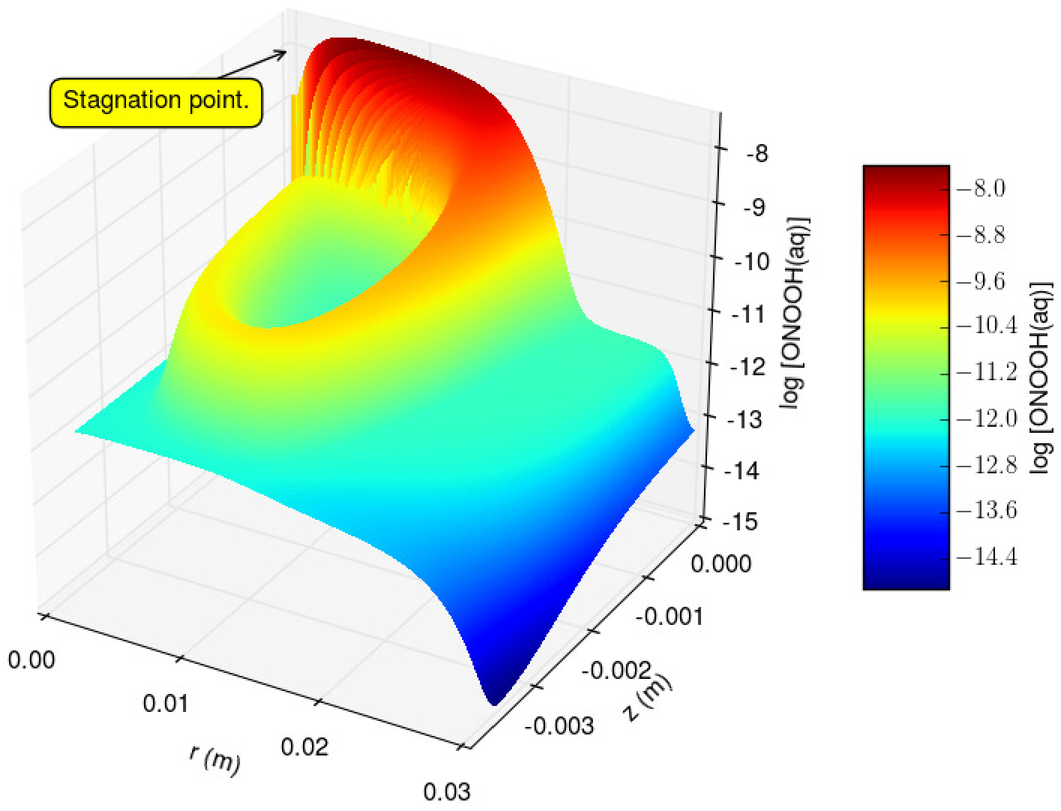


Figure 2.11 3D plot of $\log_{10}$(ONOOH(aq)) as a function of position for t = 1000 seconds. As with OH, large interfacial gradients are evident as is the effect of the liquid convection loop. Note that the r and z axes have different scales

The results presented here suggest two different regimes of activity in the solution: surface and bulk. When the discharge is on, reactivity in the form of OH is confined almost entirely to the

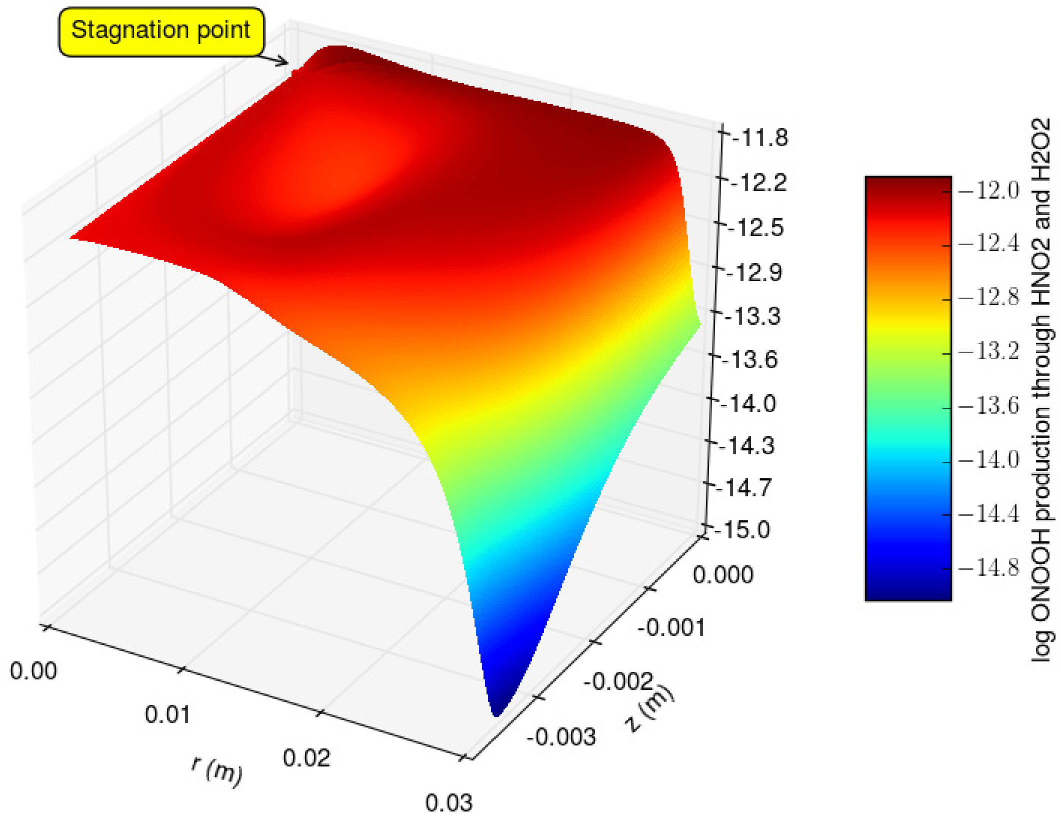


Figure 2.12 3D plot of the base 10 logarithm of the producation rate of ONOOH through reaction 14 listed in table 2.6. Because of the relative uniformity in the distribution of $H^+$, $H_2O_2$, and $NO_2^-$, the production of ONOOH through reaction 14 is much more uniform than the overall concentration profile of ONOOH. Note that the r and z axes have different scales.

plasma-liquid interface. Bulk solution concentrations of OH are several orders of magnitude lower than the surface concentration. This is also true of other potentially biologically important and highly reactive reagents such as NO and $NO_2$ radicals. Less reactive plasma-generated RONS like $H_2O_2$ and $NO_2^-$ have significantly longer lifetimes and can be transported into the bulk solution where they can react and form ONOOH, a precursor to OH and $NO_2$. However, for the model inputs used here, the bulk reactivity generated through $H_2O_2$ and $NO_2^-$ precursors is orders of magnitude less on a per unit time scale than the surface reactivity coming from direct fluxes of plasma generated radicals. When the discharge is turned off, generation of reactive species through $H_2O_2$ and $NO_2^-$ will persist for a while in the bulk. The results in [16] show a nitrite half-life of 2-3 hours in acidic solution; Traylor et. al. [17] observed anti-bacterial efficacy of PAW, which they attributed to ONOOH and its products, for several days after plasma treatment. If one assumes that every mole of $H_2O_2$ present in solution at the conclusion of our simulation reacts over time to form ONOOH and 30% of ONOOH dissociation creates OH, then the amount of OH coming from this long-term bulk reaction is 32% less than the amount of OH that comes from surface fluxes while the “discharge" is on. This comparison suggests that if one is willing to wait many hours or several days, bulk reactivity can approach surface reactivity on an order of magnitude scale. However, if an application demands speed and efficient utilization of plasma generated reactivity, the target must be placed right at the plasma-liquid interface. If the target is removed from the plasma by millimeters or even hundreds of $\mu$m of aqueous solution, its rate of treatment will rely on a serial process: the rate of transport of longer-lived species like $H_2O_2$ and $NO_2^-$ through the bulk fluid followed by the rate of formation of peroxynitrous acid chemistry.

It should be noted that aqueous phase specie concentrations will be a function of plasma conditions such as electric field distribution and as previously mentioned, the water vapor concentration in the plasma region. Water vapor concentration will play a large role in determining the gas phase concentrations of species such as OH and $H_2O_2$. For a geometric configuration similar to that modelled here, Yagi, et. al. [20] showed relative humidity in the vicinity of the plasma increasing from less than 5% to 90% as the flow rate was decreased from 1 to 0.05 liters per minute (Lpm). Decreased water vapor content led to a significant reduction in gas phase OH for flow rates as low as 0.3 Lpm, which is comparable to the flow rate in this study. Since the gas phase inputs for the model were taken from a DBD configuration, it seemed prudent to consider whether the likely reduced fluxes of water vapor delivered to the gas-liquid interface in a jet or streamer-type system would affect one of the principal qualitative conclusions of this study: that aqueous bulk concentrations of highly reactive radicals can be orders of magnitude less than their interfacial concentrations.

To study this, we constructed a 1D model of just the liquid phase, neglecting liquid-phase convection (mostly in the radial direction at the interface) and considering only diffusion and reaction. Steady-state interfacial concentrations of OH, NO, $NO_2$, and $N_2O_4$ from the full 2D axisymmetric simulation were scaled up by a factor of ten and used as boundary conditions in the 1D liquid model. The scaling in the 1D model is meant to produce conditions more comparable to those in experiments. Using the described boundary conditions, the penetration distances of OH and NO into the bulk solution (defined to be the distance required for a log reduction in concentration) were 5.8 and 4.7 $\mu$m respectively. Another simulation was conducted in which the interfacial concentration of OH was reduced by a factor of 10 to reflect the potential reduction in OH fluxes to the interface for convective discharge systems. Under this scenario, OH and NO penetration distances were both 16$\mu$m. In both simulations, $NO_2$ concentrations fell sharply in the first tens of microns but then displayed a second local concentration maximum around 100 $\mu$m. This second concentration maximum arises from ONOOH dissociation rates outweighing the loss of $NO_2$ through radical reactions; as already stated, two of $NO_2$'s principal reaction partners, NO and OH, are depleted within the first tens of microns. The concentration of $NO_2$ at this second maximum is an order of magnitude less than the interfacial NO concentration and two orders of magnitude less than the interfacial OH concentration for the diluted convective case. If the OH concentration were diluted quite a bit further (which may not be physically realistic), it is conceivable that NO penetration as predicted by our model could approach hundreds of microns or even millimeters. However, the physical conditions that would promote a very low interfacial OH concentration may also promote higher NOx radical interfacial concentrations than those used in our model. NOx radical reaction rate constants are on the same order of magnitude as OH reactions ( 1e10 moles/L); consequently one could envision a case in which interfacial OH radical concentrations are very low, but the penetration distance of radical NOx species is still on the order of microns because of self-reactions.

On a related note, the 1D model can be used to analyze the effect of the uniform dilution factor used in the 2D-axisymmetric model on the penetration distance of radicals. From the species mass conservation equations we expect penetration distances to decrease as input concentrations are scaled up since the only non-linear terms are reaction sinks. More precisely, the dimensionless Damkohler number, $Da = \frac{kCL^2}{D}$, for a second-order reaction-diffusion problem shows that the penetration distance will scale like $\frac{1}{\sqrt{C}}$. [108, p. 55] This hypothesis is confirmed by the 1D model. When the interfacial concentrations from the 2D-axisymmetric model are input directly into the 1D model, OH and NO penetration distances are 18 and 15 microns respectively. These are almost exactly a factor of $\sqrt{10}$ greater than the 5.8 and 4.7 $\mu$m penetration distances seen when the concentration inputs are scaled up by 10. If the concentrations are increased more, the penetration distances are even further reduced, following the scaling of $\frac{1}{\sqrt{C}}$. These analyses of the radical concentration profiles support one of our main arguments: though the amount and type of species present at the plasma-liquid interface may vary between discharge types, most of the plasma generated reactivity will likely lie within a small interfacial region, microns to a couple hundred microns, regardless of the experimental configuration.

Returning to the 2D model, we can introduce a dissolved marker like Tyrosine into solution and observe its degradation via reactive plasma species and predict product formation for potential future validation with experiments. Tyrosine reactions are detailed at the end of table 2.6. Figure 2.14 shows the concentration profile of tyrosine after fifteen minutes of exposure to reactive plasma species. The origin represents the stagnation point or the point at which the center of the plasma column impinges on the liquid surface. Z = 0 represents the water surface. After treatment, the concentration of tyrosine at the surface has been roughly cut in half. The presence of bulk liquid advection is evident in the decreased tyrosine concentration surrounding a local maximum at roughly (.01, -.0015). We can also look at the volume-averaged concentrations of tyrosine products in fig. 2.13. TyrDot is tyrosine radical, diTyr is dityrosine, TyrNO is tyrosine with an added NO functional group, TyrNO2 has an added $NO_2$ functional group, and TyrOHo and TurOHm are tyrosine with an additional OH functional group at the ortho- and meta- positions respectively. Our model predicts that dityrosine will be the dominant terminal specie with TyrOHm and TyrDot representing important reactive intermediates.

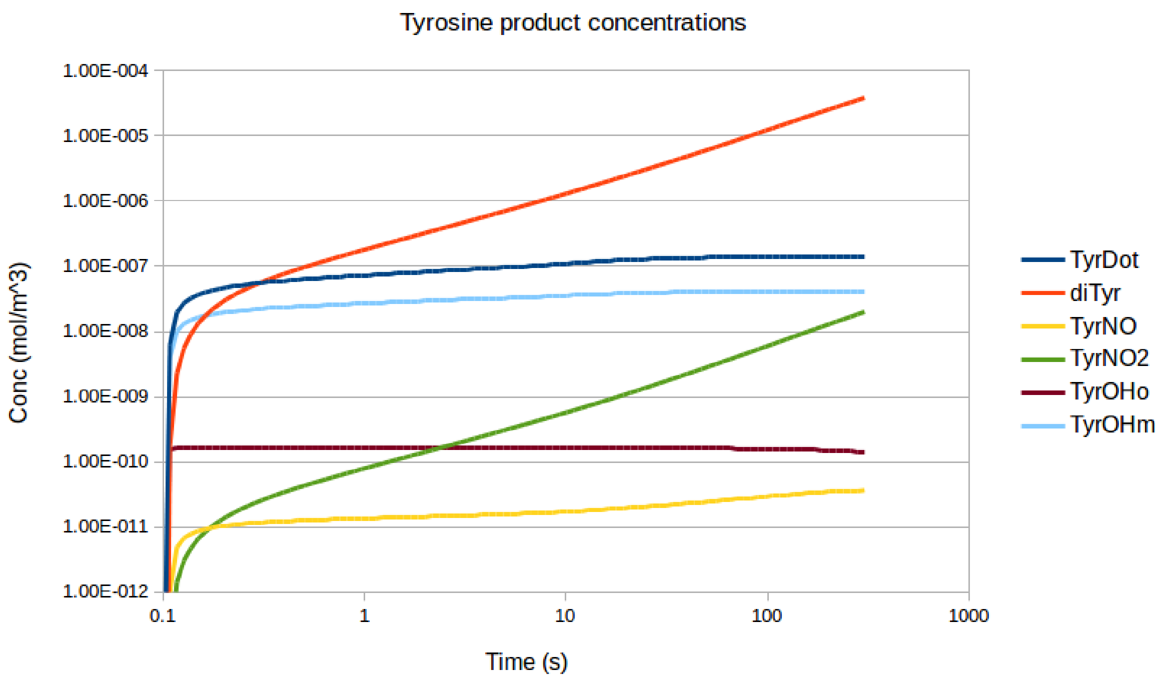


Figure 2.13 Plot vs. time showing the growth of different tyrosine products. The dominant product formed is dityrosine.

Before concluding, it is worth touching on the assumption of a flat interface. This work discusses

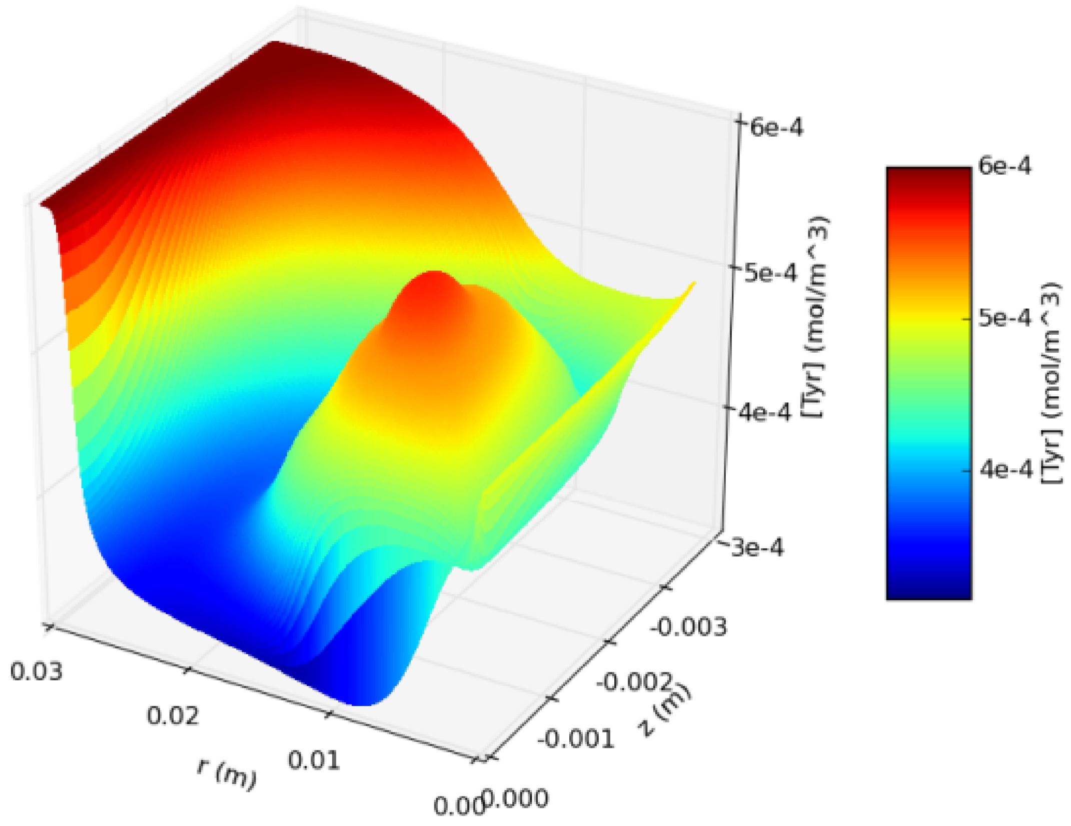


Figure 2.14 Plot of tyrosine concentration at the end of 15 minutes of reactive species exposure. Concentration is significantly depleted near the liquid surface as well as within the advective re-circulating loop.

the role gas phase convection plays in determining momentum, heat, and mass transport across a gas-liquid interface and into the bulk liquid. Species uptake is used as the criteria for determining whether to include self-consistent deformation of the interface by the gas flow. To make this determination, the shape of the interface deformation was observed experimentally and the deformation was introduced into the model geometry as shown in figure 2.15. The depth of the deformation was supported by a calculation balancing gravitational ($\rho$gh) and convective ($\frac{1}{2}\rho v^2$) stresses. The fluid flow simulation was then run until it reached stead-state, and then the heat and mass transfer equations were solved. In this way, though the interface deformation was not self-consistently determined, its influence on variables of interest could be analyzed. Shown in

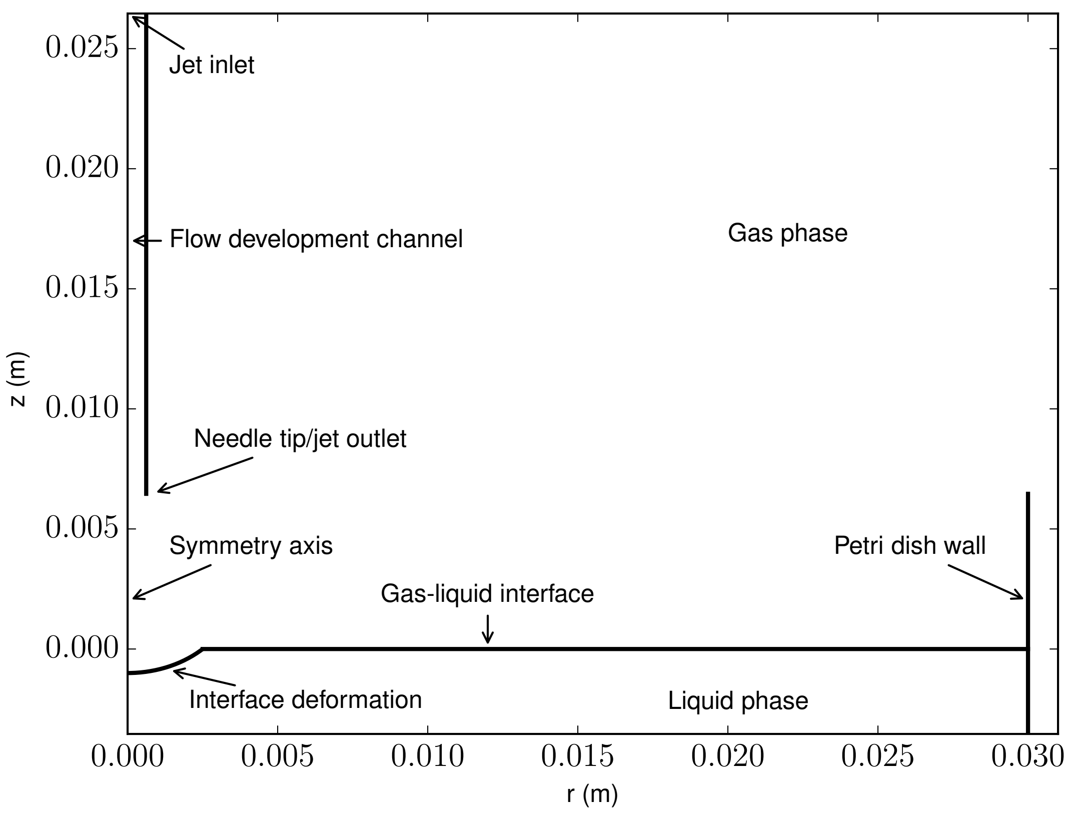


Figure 2.15 Geometry for deformed interface simulations.

2.16 is the effect of the interface deformation on total solution uptake of NO(aq). As can be seen in the figure, the effect is minute. Because of the characteristics shown in Figure 2.9, the effect of the interface deformation on hydrophilic species transport is expected to be even less. Subsequently, for the purpose of this work, the interface deformation can be safely neglected.

Though comprehensive in many respects, the model in this section is not near complete without simulation of plasma discharge physics. Incorporating discharge physics requires changing simulation platforms from Comsol to Zapdos, a highly customizable code built by the authors where near total control over solution of the highly-nonlinear plasma equations is maintained. Zapdos is described in detail in chapter 3. The 1D simulation effort in section 2.2 are the authors' first foray into self-consistent discharge-liquid modelling. Along with exploring questions about interfacial parameters, the work in section 2.2 is absolutely necessary for extending the work in

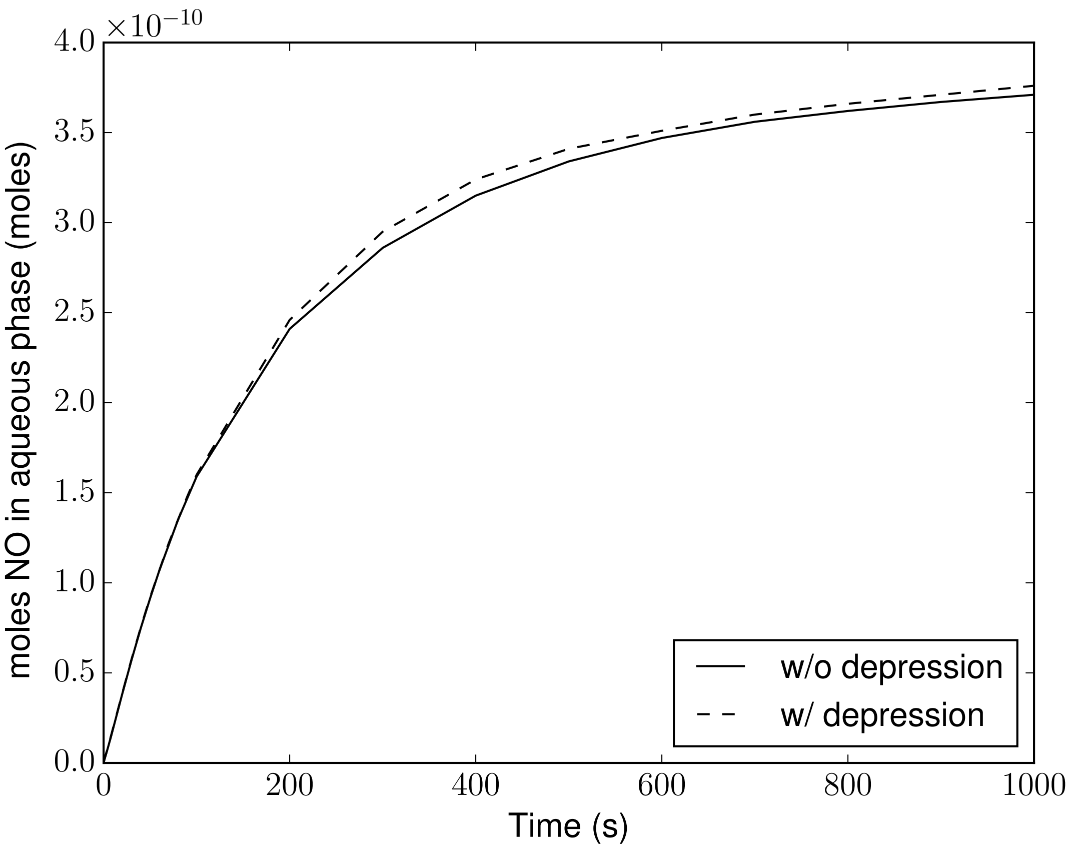


Figure 2.16 Comparison of total NO(aq) uptake as a function of time for cases in which interface deformation is included and not included. As shown in the figure, the interface deformation has very little effect on the macrosocpic NO uptake. The effect for hydrophilic species is expected to be even less

this section. Combining the two modelling efforts is definitely on the roadmap for future work.

## 2.2 Fully Coupled Simulation of the Plasma Liquid Interface and Interfacial Coefficient Effects

### 2.2.1 Model Description

As suggested at the end of section 2.1.2, truly grasping the nature of interactions between plasma and liquid phases requires a fully coupled model that combines the aqueous phase with discharge physics. To this end we have created Zapdos, a physics application built on top of the MOOSE framework for simulating plasma-liquids. A description of the code, which is open source and free to use [36], is given in chapter 3. In this section we consider a DC atmospheric pressure argon discharge in one dimensions impinging on a very thin water layer. The powered electrode is biased negatively, making it the cathode. From the plasma's perspective, the water surface is the anode. Only elastic collisions, ground state ionization, and ground state excitation are considered. The model governing equations are described below. Continuity equations based on the drift-diffusion approximation are solved for the electrons and ions:

$$\frac{\partial n_i}{\partial t} + \nabla \cdot \vec{\Gamma_i} = S_{iz} \tag{2.9}$$

$$\frac{\partial n_e}{\partial t} + \nabla \cdot \vec{\Gamma_e} = S_{iz} \tag{2.10}$$

$$\vec{\Gamma_i} = \mu_i \vec{E} n_i - D_i \nabla n_i \tag{2.11}$$

$$\vec{\Gamma_e} = \mu_e \vec{E} n_e - D_e \nabla n_e \tag{2.12}$$

$$S_{iz} = \alpha_{iz} |\vec{\Gamma_e}| \tag{2.13}$$

where $\mu$ is the mobility, D the diffusivity, $\alpha_{iz}$ the Townsend ionization coefficient, $\Gamma$ the species flux, S$_{iz}$ the ionization source term, n the species density, and $\vec{E}$ the electric field, equal to $\nabla V$ where V is the potential. Equation (2.13) utilizes a Townsend formulation as opposed to a rate coefficient formulation to describe the rate of ionization. In the Townsend formulation, the ionization rate is proportional to the electron flux; in a rate coefficient formulation, the ionization rate is proportional to the electron density. Hagelaar et. al. [109] recommend using a Townsend formulation in cases where the electron flux is field driven and in particular in the cathode region of DC discharges where the drift-diffusion approximation can lead to significant errors in the electron density but hardly affect the electron flux.

Poisson's equation is solved for the potential:

$$-\nabla^2 V = \frac{e\left(n_i - n_e\right)}{\epsilon_p} \tag{2.14}$$

where e is the Coulombic charge and $\epsilon_p$ is the permittivity of the plasma, set equal to the permittivity of free space for our model. The equation for the electron energy is:

$$\frac{\partial\left(n_e\epsilon\right)}{\partial t} + \nabla\cdot\vec{\Gamma_\epsilon} = -e\vec{\Gamma_e}\cdot\vec{E} - |\vec{\Gamma_e}|\left(\alpha_{iz}\epsilon_{iz} + \alpha_{ex}\epsilon_{ex} + 3\frac{m_e}{m_i}\alpha_{el}T_e\right) \tag{2.15}$$

$$\vec{\Gamma_\epsilon} = \frac{5}{3}\epsilon\vec{\Gamma_e} - \frac{5}{3}n_e D_e \nabla\epsilon \tag{2.16}$$

where $\epsilon$ is the mean electron energy, $\epsilon_{iz}$ the electron energy lost in an ionization collision, $\alpha_{ex}$ the Townsend excitation coefficient, $\epsilon_{ex}$ the electron energy lost in an excitation collision, $m_i$ and $m_e$ the ion and electron masses respectively, $\alpha_{el}$ the Townsend elastic collision coefficient, and $T_e$ the electron temperature, equal to $\frac{2}{3}\epsilon$.

Plasma boundary conditions at the cathode are based on the work in [110] and [30]. For ions, electrons, and the electron energy, the conditions are respectively:

$$\vec{\Gamma_i}\cdot\vec{n} = \frac{1-r_i}{1+r_i}\left(\left(2a_i - 1\right)\mu_i\vec{E}\cdot\vec{n}n_i + \frac{1}{2}v_{th,i}n_i\right) \tag{2.17}$$

$$\vec{\Gamma_e}\cdot\vec{n} = \frac{1-r_{dens}}{1+r_{dens}}\left(-\left(2a_e - 1\right)\mu_e\vec{E}\cdot\vec{n}\left(n_e - n_\gamma\right) + \frac{1}{2}v_{th,e}\left(n_e - n_\gamma\right)\right) - \left(1 - a_e\right)\gamma_p\vec{\Gamma_p}\cdot\vec{n} \tag{2.18}$$

$$\vec{\Gamma_\epsilon}\cdot\vec{n} = \frac{1-r_{en}}{1+r_{en}}\left(-\left(2a_e - 1\right)\frac{5}{3}\mu_e\vec{E}\cdot\vec{n}\left(n_e\epsilon - n_\gamma\epsilon_\gamma\right) + \frac{5}{6}v_{th,e}\left(n_e\epsilon - n_\gamma\epsilon_\gamma\right)\right) - \frac{5}{3}\epsilon_\gamma\left(1 - a_e\right)\gamma_p\vec{\Gamma_p}\cdot\vec{n} \tag{2.19}$$

where $r_i$, $r_{dens}$, $r_{en}$ are the boundary reflection coefficients for ions, electrons, and electron energy respectively (more discussion on $r_{en}$ shortly), $\gamma_p$ is the secondary electron emission coefficient, $\epsilon_\gamma$ is the energy of the secondary electrons, $\vec{n}$ is the outward facing normal vector, and:

$$a_k = \begin{cases} 1, & sgn_k\mu_k\vec{E}\cdot\vec{n} > 0 \\ 0, & sgn_k\mu_k\vec{E}\cdot\vec{n} \leq 0 \end{cases} \tag{2.20}$$

$$v_{th,k} = \sqrt{\frac{8T_k}{\pi m_k}} \tag{2.21}$$

$$n_\gamma = (1 - a_e)\,\frac{\gamma_p \vec{\Gamma_p} \cdot \vec{n}}{\mu_e \vec{E} \cdot \vec{n}} \tag{2.22}$$

where $v_{th,k}$ is the thermal velocity of species $k$ and $n_\gamma$ is the density of secondary electrons. All $r_k$'s are set to zero at the cathode. At the interface of the plasma with the liquid phase, the ion boundary condition is the same as for the cathode with $r_i = 0$. For electrons in the gas phase two formulations are considered. The first is the kinetic formulation given by eq. (2.18) where $r_{dens}$ is variable. The second is a thermodynamic formulation analogous to Henry's law where the ratio of the liquid phase electron density to the gas phase electron density is specified by a variable H (equivalent to a Henry's Law coefficient):

$$Hn_{e,g} = n_{e,l} \tag{2.23}$$

The electron energy interfacial condition is the kinetic one, see eq. (2.19). Though $r_{dens}$ (or $H$ for the thermodynamic electron BC) at the interface is varied in the results that follow, $r_{en}$ is held constant at 0 for most simulations. This is done for the following physical reasoning. Electrons can either pass freely into the liquid phase, carrying their energy with them, or they can be reflected. If they are reflected, then it is reasonable to expect these electrons to lose their energy in surface collisions such as vibrational excitation of $H_2O$ until they are incorporated into the liquid. Thus though some electrons coming from the bulk may be reflected, it may be reasonable to assume that all the electron energy coming from the bulk is absorbed by the interface. However, in the interest of covering all realms of possibility (perhaps most electron collisions at the interface are low-loss elastic collisions for example), a study is conducted in which the amount of energy absorbed/reflected by the interface is varied. This is done by changing $\gamma_{en}$. Note that in the plots and discussion to follow, the surface loss coefficients $\gamma_{dens}$ and $\gamma_{en}$ will often be used instead of the reflection coefficients $r_{dens}$ and $r_{en}$. The relationship between surface loss and reflection coefficients is simply $\gamma_k = 1 - r_k$.

The liquid phase electron density interfacial condition is given simply by the continuity of flux. At the bottom of the liquid, electrons are assumed to recombine or flow out at a rate equivalent to the advective flux.

For potential conditions, V is set to zero at the end of the liquid domain. At the cathode, Kirchoff's voltage law for a circuit including a ballast resistor yields:

$$V_{source} + V_{cathode} = \left(e\vec{\Gamma_i} - e\vec{\Gamma_e}\right) AR \tag{2.24}$$

where A is the cross-sectional area of the plasma and R is the ballast resistance.

Gas phase electron coefficients were calculated in the following way: Argon ionizization, excitation, and elastic collision cross sections were taken from the Phelps database [111] at [112]. Then using the open source Boltzmann solver Bolos [113] based on the work of Hagelaar [109] electron energy distribution functions were calculated for 200 electric field points between $10^3$ and $10^7$ V/m. Then for each distribution function, $\mu_e$, D$_e$, $\epsilon$, and the necessary electron collision rate coefficients were calculated as defined by [109]. Transport and rate cofficients were tabulated against the mean energy. These lookup-tables were then referenced during solution of the fluid equations. The details of the inputs for the fluid simulations are given in tables 2.7 and 2.8 and figure 2.17. Mesh sizes for the simulations were typically around 200 elements with most elements located in the cathode and interfacial regions. Each individual simulation took between 12 and 60 seconds to run.

Table 2.7 Plasma liquid simulation input parameters

| Parameter | Value |
|---|---|
| Gas | Argon |
| Pressure | 1 atm |
| $\gamma_p$ | 0.15 |
| A | $5.02 \cdot 10^{-7} m^2$ |
| R | $10^6 \Omega$ |
| V$_{source}$ | 1.25 kV |
| Gas Domain | 1 mm |
| Liquid Domain | 100 nm |
| $\epsilon_\gamma$ | 3 eV |
| T$_i$ | 300 K |

Table 2.8 Plasma liquid simulation input parameters

| Coefficient | Value | Source |
|---|---|---|
| $\mu_e$ | Variable | [113] |
| $D_e$ | Variable | [113] |
| $\mu_i$ | $3.52 \cdot 10^{-4} m^2 s^{-1} V^{-1}$ | [114] |
| $D_i$ | $5.26 \cdot 10^{-6} m^2 s^{-1}$ | [114] |
| $\alpha_{iz}$ | Variable | [113] |
| $\alpha_{ex}$ | Variable | [113] |
| $\alpha_{el}$ | Variable | [113] |
| $\epsilon_{iz}$ | 15.76 eV | [112] |
| $\epsilon_{ex}$ | 11.5 eV | [112] |

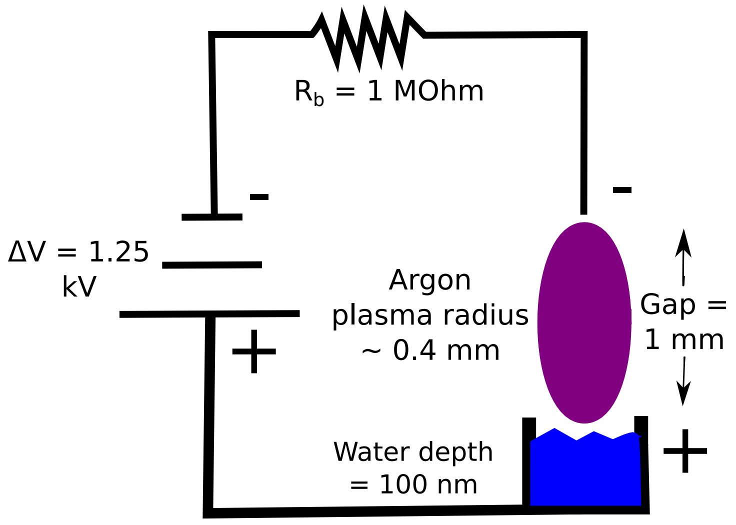


Figure 2.17 Circuit schematic of coupled plasma liquid system. Note that diagram is not to scale

### 2.2.2 Results and Discussion

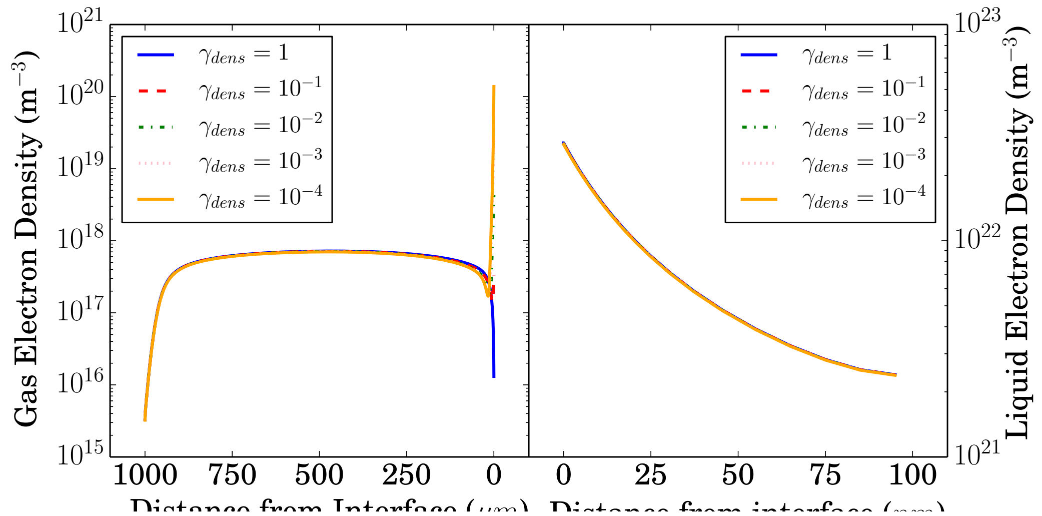


Figure 2.18 Electron density as a function of the interfacial surface loss coefficient

Figure 2.18 shows the electron density in both the gas and liquid phases as a function of the interfacial surface loss coefficient. The cathode and bulk profiles are unaffected by changing $\gamma_{dens}$. However, as one might expect, decreasing the surface loss coefficient leads to a build-up of electrons on the gas phase side of the interface, seen more clearly in fig. 2.19. Similar behavior can be achieved by decreasing the $H$ coefficient in eq. (2.23) and fig. 2.21. In order to observe anode characteristics akin to those for a plasma in contact with a metallic electrode ($\gamma_{dens} = 1$), $H$ must be on the order of $10^6$. This is on the same order of magnitude as Henry's Law coefficients for $H_2O_2$ and $HNO_3$, both very hydrophilic species. If $H$ is reduced to $10^4$, the gas phase electron density near the interface increases by an order of magnitude. If $H$ is further reduced to $10^2$, only slightly less hydrophilic than OH, then the gas phase interfacial density rockets up to three orders of magnitude greater than the metallic anode base case. Decreasing $H$ further only continues the trend.

Despite the dramatic functional dependence of the gas phase electron density in the anode, the liquid phase electron density profile remains unchanged as $\gamma_{dens}$ is varied. The reason for this can be seen by looking at fig. 2.22. Like the liquid phase electron density profile, the potential drop across the plasma-liquid system is unaffected by changing $\gamma_{dens}$. This means that the

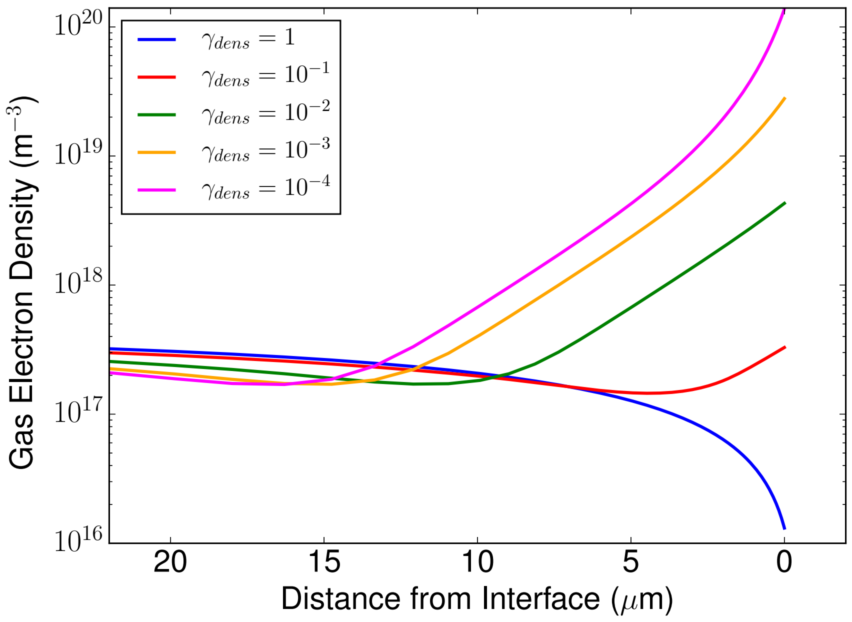


Figure 2.19 Electron density as a function of the interfacial surface loss coefficient. Final 20 $\mu$m of the gas phase before the interface.

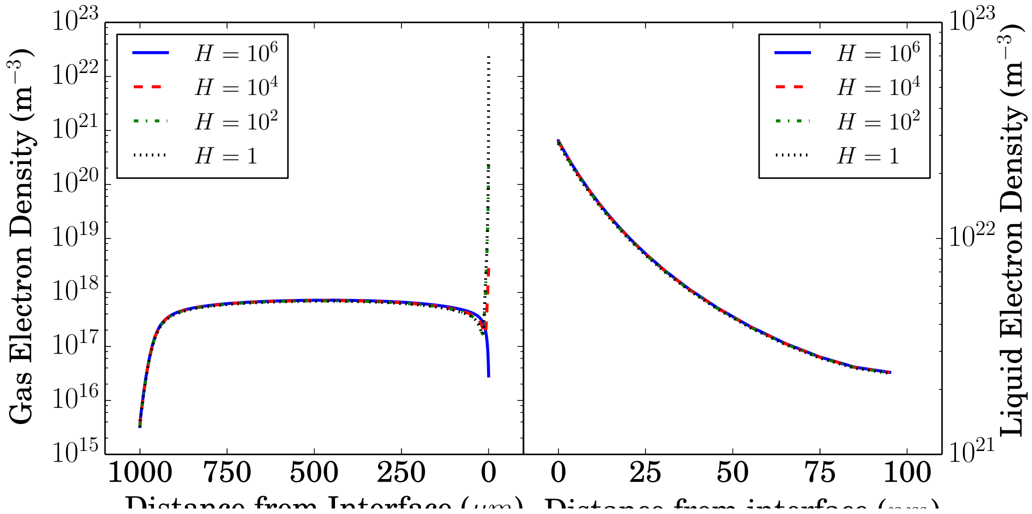


Figure 2.20 Electron density as a function of $H$ using the thermodynamic boundary condition. Shows same trend as fig. 2.18

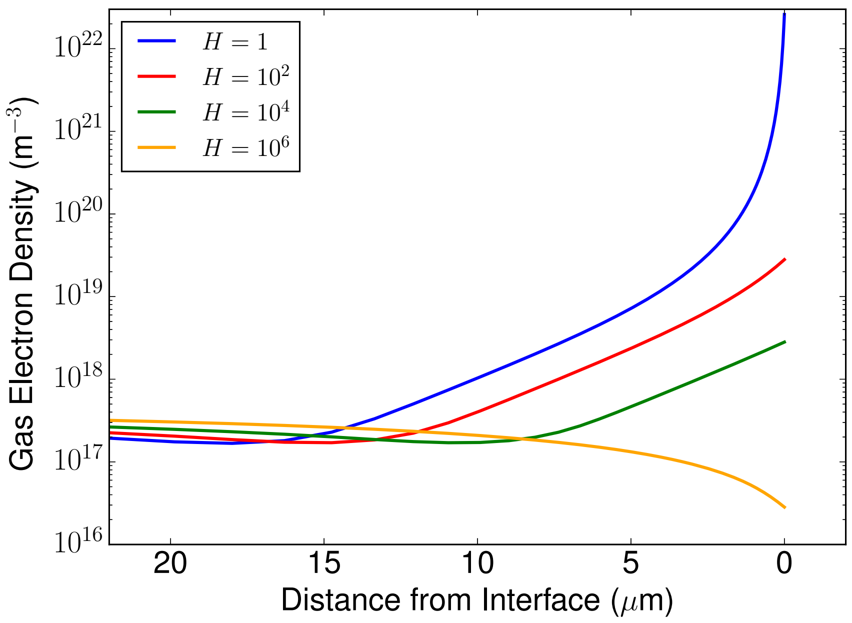


Figure 2.21 Electron density as a function of $H$ over the last 20 $\mu$m of the gas phase. Shows same trend as fig. 2.19

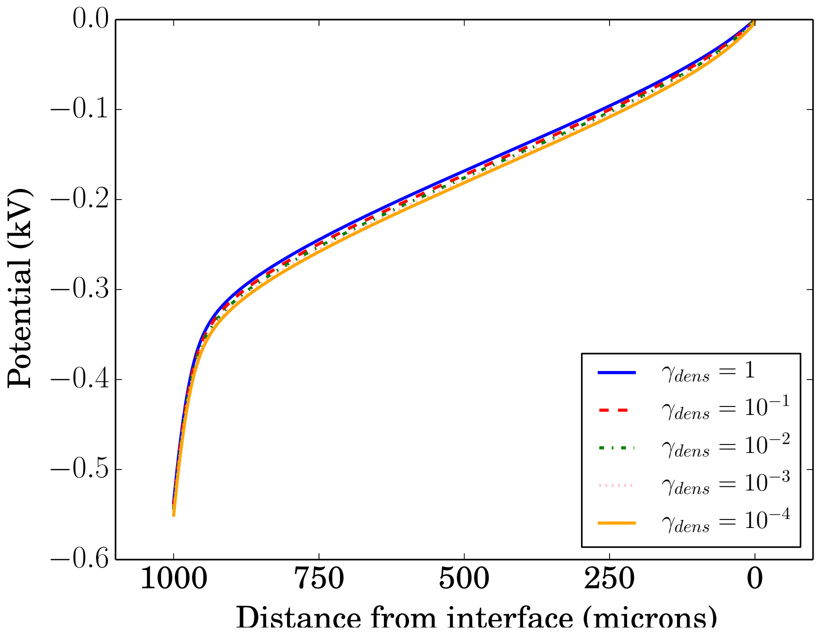


Figure 2.22 Potential as a function of the interfacial surface loss coefficient

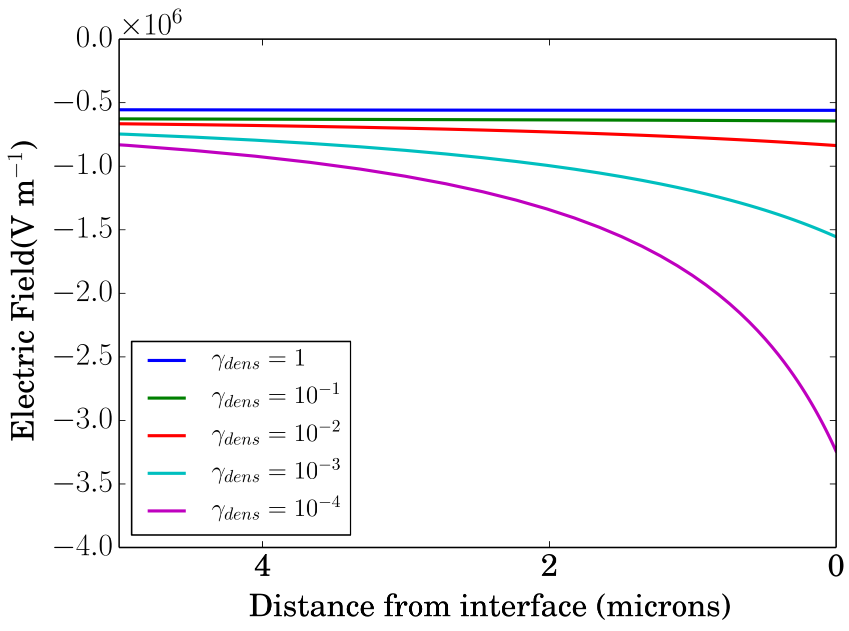


Figure 2.23 Electric field near the interface as a function of the interfacial surface loss coefficient

system DC current is also unaffected, roughly 1000 A m$^{-2}$ for all simulation cases. Away from the cathode, all the current is carried by electrons, thus the electron current at the interface between the gas and liquid must also remain unchanged as $\gamma_{dens}$ is varied. With the liquid phase electron input thus unaffected by $\gamma_{dens}$, the liquid phase electron density profile remains constant. Varying $\gamma_{dens}$ does change the potential and electric field profiles near the interface; this is shown in fig. 2.23. From the low reflection to high reflection extremes, the interfacial electric field increases by about a factor of seven.

As with the electron density, the cathode and bulk electron temperature profiles in fig. 2.25 do not change as $\gamma_{dens}$ is varied. However, there is major variation in the anode. This variation arises from the assumption described in the model description section that electrons coming from the bulk either carry their energy into the liquid phase upon absorption or else if reflected lose their energy through interfacial surface collisions. The greater the reflection, the lower the average energy of electrons near the interface because of non-recombinatory surface collisions. This is what is observed in fig. 2.25. This trend in electron energy also explains the slight variation in anode ion density profiles seen in fig. 2.24. Lower electron mean energy near the interface means a smaller fraction of electrons with sufficient energy to create ionization and

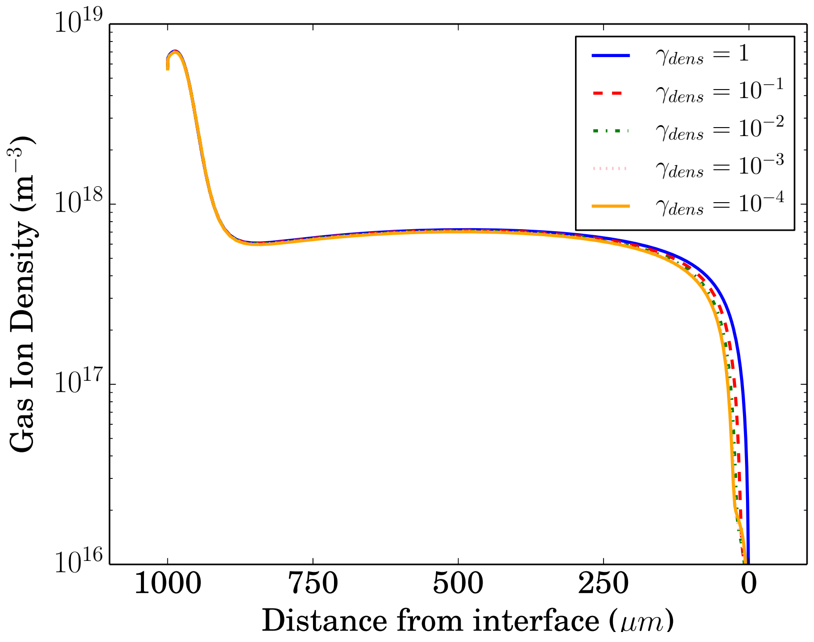


Figure 2.24 Ion density as a function of the interfacial surface loss coefficient

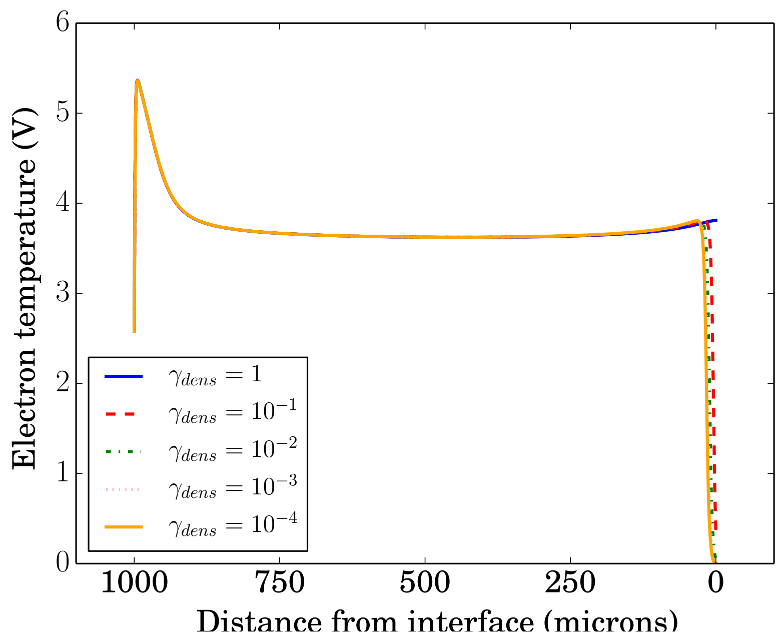


Figure 2.25 Electron temperature as a function of the interfacial surface loss coefficient

a smaller Townsend ionization coefficient. Because in this model ionization is proportional to the electron flux magnitude and because the electron flux magnitude is constant with respecto to $\gamma_{dens}$, the decrease in $\alpha_{iz}$ corresponds to a decrease in the rate of ionization. Hence the ion density rises to its bulk value farther from the anode for decreasing $\gamma_{dens}$.

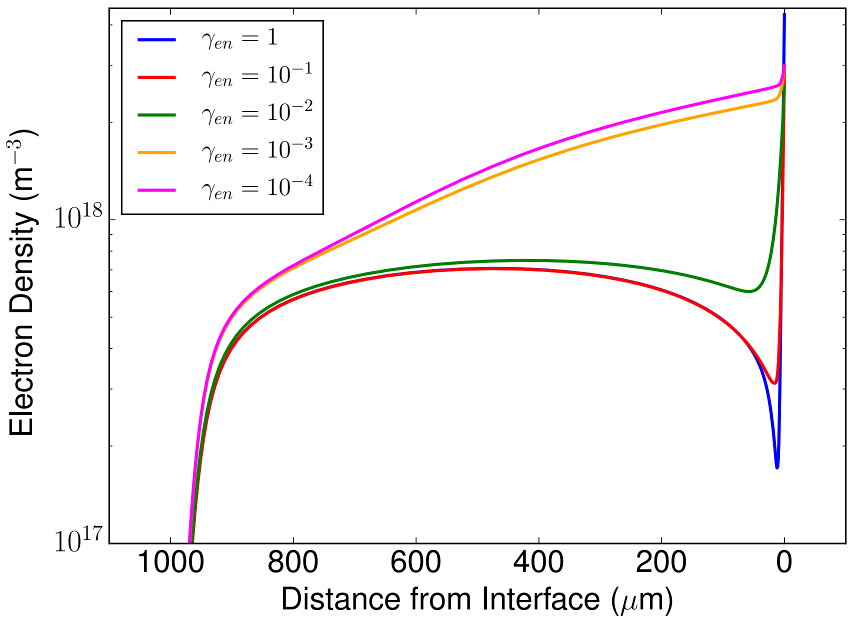


Figure 2.26 Gas phase electron density as a function of the electron energy interfacial surface loss coefficient. ($\gamma_{dens} = 10^{-2}$ for all cases)

The physically correct boundary condition for the electron energy at the interface is unknown. However, we can vary the amount of electron energy that is absorbed/reflected at the interface and see whether that affects the most important result of the above figures: that interfacial electron density increases significantly as the electron surface loss coefficient is decreased. Figure 2.26 shows the effect of varying the amount of energy lost at the interface when $\gamma_{dens}$ is kept constant at $10^{-2}$. A couple of trends are notable. The first is that as the energy reflection is increased, e.g. as $\gamma_{en}$ is decreased, the bulk electron density increases; moreover, instead of retaining a flat profile through the bulk, the electron density increases almost linearly moving from cathode to anode. Additionally, as $\gamma_{en}$ decreases the jump in electron density at the anode/interface decreases. The combination of these effects results in anodic electron densities that differ by less

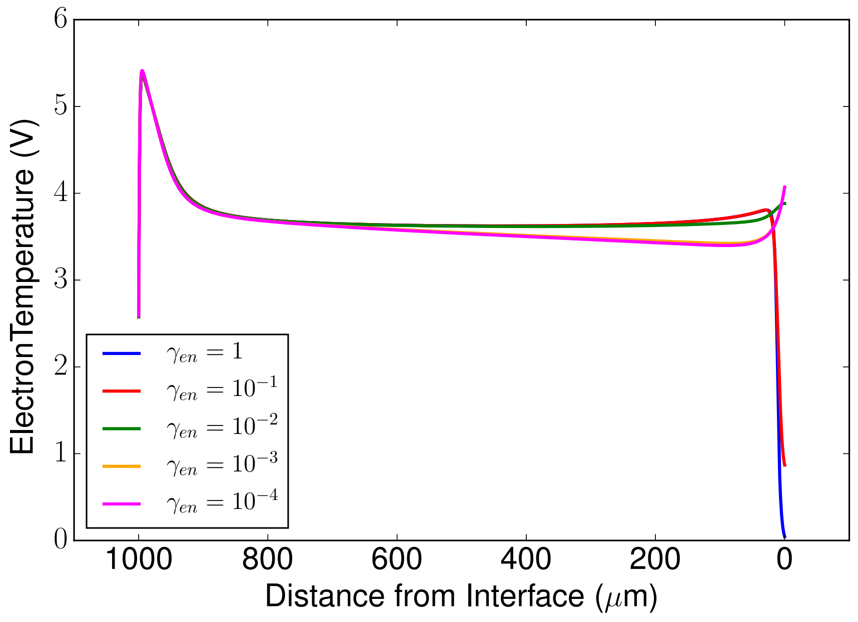


Figure 2.27 Gas phase electron temperature as a function of the electron energy interfacial surface loss coefficient. ($\gamma_{dens} = 10^{-2}$ for all cases)

than a factor of two over values of $\gamma_{en}$ that span four orders of magnitude. Moreover, no matter the value of $\gamma_{en}$, the anodic electron density with $\gamma_{dens} = 10^{-2}$ is over an order of magnitude higher than if the surface loss coefficients for electrons is set to unity. Thus, we conclude that the important result of increasing anodic electron density with decreasing $\gamma_{dens}$ is relatively insensitive to the choice of $\gamma_{en}$; e.g. without knowing how to properly handle the electron energy boundary condition at the interface, we can still reasonably conclude that a decreasing surface loss coefficient will significantly increase the density of gas phase electron at the interface. The effect of varying $\gamma_{en}$ on the electron temperature gas phase profile is shown in fig. 2.27. Changes in the cathode and bulk profiles are minimal. However, as one might intuitively expect, increasing energy reflection increases the anodic electron temperature. An increase in electron temperature from the bulk to the anode (observed for $\gamma_{en} = 10^{-4}$) is more consistent with high current atmospheric argon PIC simulations. [115]

These trends in the anode electron density and electron temperature at the anode could play an important role in more complex models that consider evaporation of $H_2O$ and dilute aqueous species. The rates of reactions of electrons with these species will depend strongly on the electron density and the electron energy distribution. Different energy distributions might favor vibrational excitation of $H_2O$ or dissociative attachment and the production of electronegative plasma species like $O^-$ and $OH^-$. The near interface gas chemistry will of course couple back into the liquid phase chemistry. Future work with more complex models will investigate how changing $\gamma_{dens}$ and $\gamma_{en}$ affects plasma and liquid chemistry. However, in order to limit the scope of possible results and increase the predictive capability of such models, there must be more certainty in interfacial parameters like $\gamma_{dens}$ and in the interfacial energy dynamics (represented in this work by $\gamma_{en}$. Determination of such characteristics will likely require finer scale simulations (molecular dynamics for instance) and/or new experimental diagnostics that are capable of probing near-interface gas dynamics.

Figure 2.28 shows the power deposited into electrons from joule heating over the last 100 $\mu$m of the gas domain. It is evident that there is significantly more heating of the electrons for the higher reflection cases. This is almost entirely attributable to the enhanced electric field in the region (fig. 2.23) that develops because of the build-up in negative charge density (fig. 2.29) from electron reflection. The electron current itself does not change with reflection as evidenced in fig. 2.32.

Power deposition into both electrons and ions for the cases of $\gamma_{dens} = 1$ and $\gamma_{dens} = 10^{-4}$ are shown in fig. 2.30. Ion power deposition profiles are identical between the different reflection cases. As expected for a DC discharge, all of the ion power deposition occurs in the cathode fall where both the electric field and ion current are significantly higher than in the bulk. Electron heating occurs throughout the discharge with local maxima in both the cathode and anode regions.

Figure 2.31 shows the rate of elastic, excitation, and ionization collisions in the discharge. Elastic collision rates show an inverse relationship to the electron temperature: in the cathode where the electron temperature peaks, the elastic collision rate crashes; in the anode for the high reflection case ($\gamma_{dens} = 10^{-4}$) the elastic collision rate peaks sharply because of the sharp fall in the electron temperature. Excitation collision rates are uniform through the bulk of the discharge for both high and low reflecion cases. Ionization rates peak in the bulk-sheath interface regions where the electron temperature is highest. Both excitation and ionization rates collapse in the cathode where electrons have not had sufficient time to gain energy; for high electron reflection the excitation and ionization rates also collapse in the anode.

Figure 2.32 is a somewhat busy figure. It illustrates the currents carried by different species in gas and liquid phases as well as the total current. Dotted lines indicate the total current,

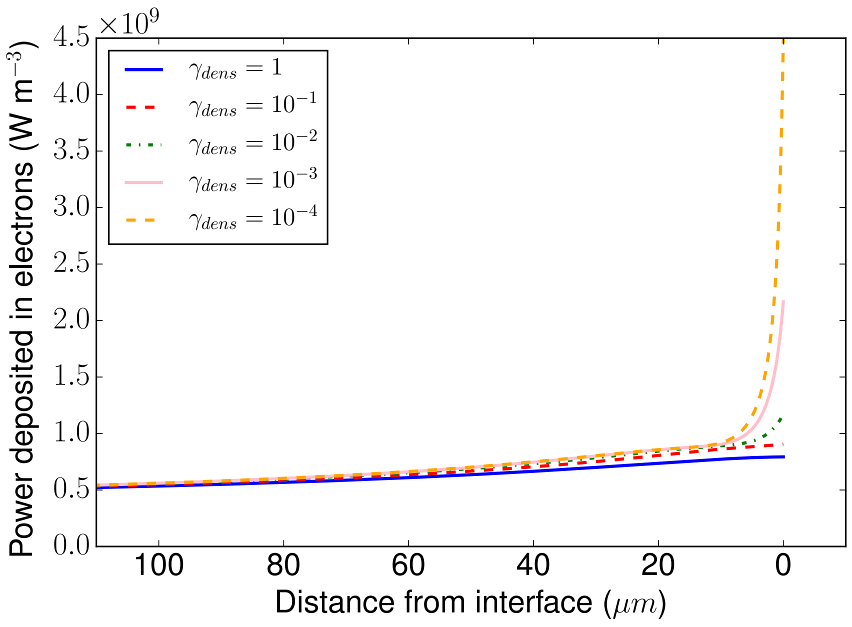


Figure 2.28 Power deposited in electrons near the interface as a function of the interfacial surface loss coefficient

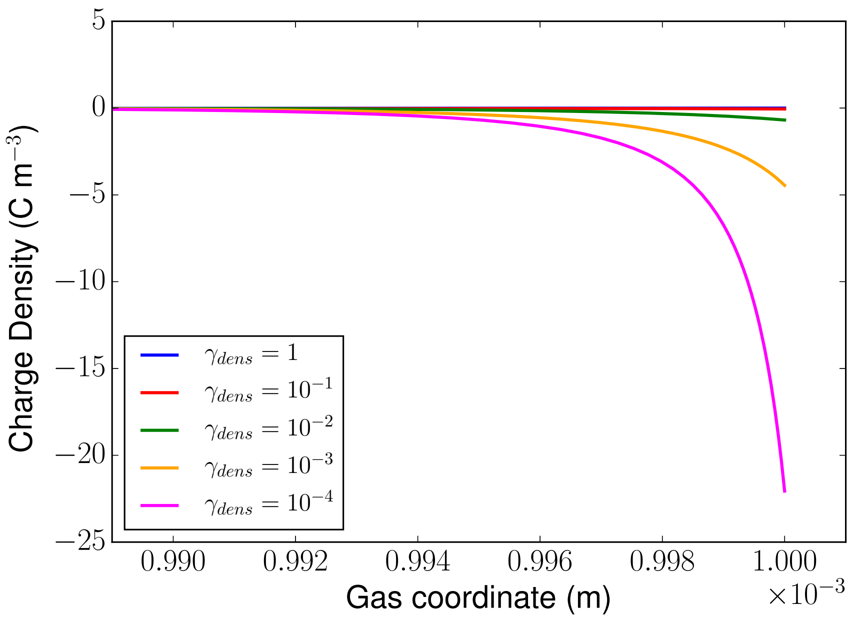


Figure 2.29 Charge density near the interface as a function of the interfacial surface loss coefficient

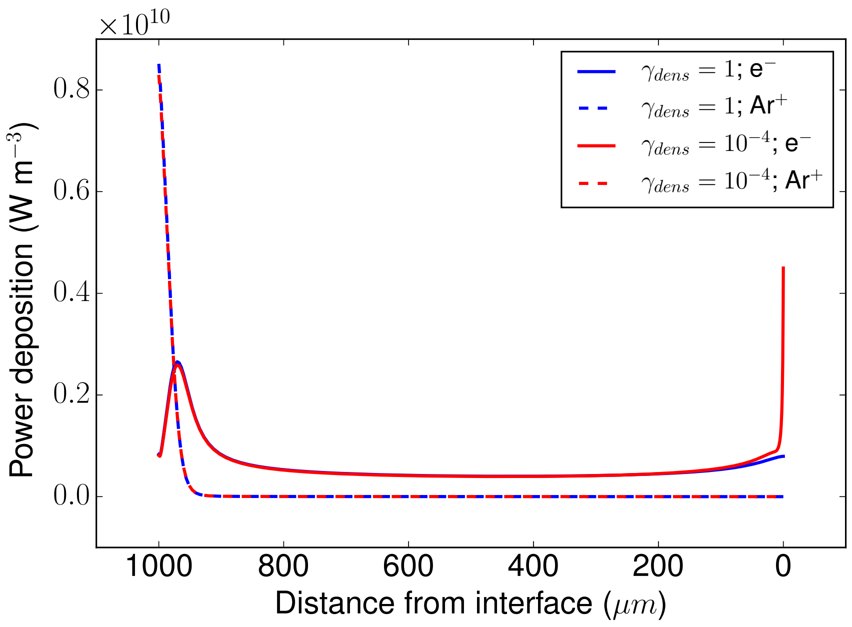


Figure 2.30 Power deposited in ions and electrons over the whole gas domain for low and high reflection cases

solid lines the electron current, and dashed lines the argon and hydroxyl currents in the gas and liquid phases respectively. The curves do not show a strong functional dependence on the surface loss coefficient $\gamma_{dens}$. Current is carried by argon ions in the cathode, transitioning to electrons in the gas bulk and through the gas-liquid interface. As electrons react with water molecules, $OH^-$ becomes the dominant liquid current carrier away from the interface.

Electron total, advective, and diffusive fluxes are illustrated in fig. 2.33. Consistent with fig. 2.32, the total electron current rises from zero at the cathode to a constant value that it maintains through the gas bulk to the interface. The total current declines in the liquid phase because of electron reactions with $H_2O$. Advection and diffusion flux profiles are very similar between high and low reflection cases with the exception of the behavior very near the gas-liquid interface. Advective flux builds in the cathode until reaching a peak at the sheath-bulk interface. The advective flux then gradually declines in the bulk because of a gradual decrease in the bulk electric field until plummeting sharply in the anode as the electron density plummets; however, for the high reflection case very near the gas-liquid interface the advective flux rebounds sharply. This counter-acts the very strong leftward diffusive flux that arises from the large build-up in

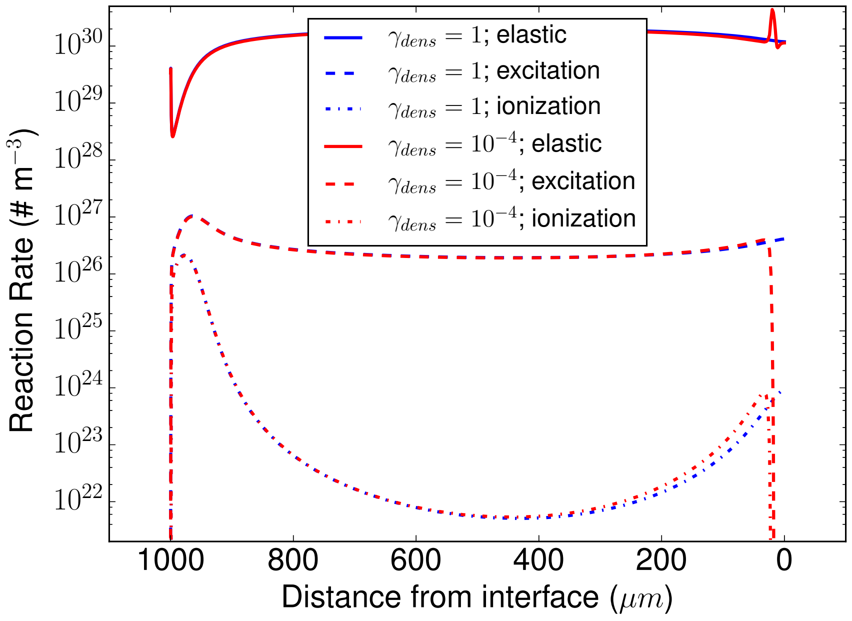


Figure 2.31 Rate of ionization, excitation, and elastic collisions in the plasma for low and high reflection cases

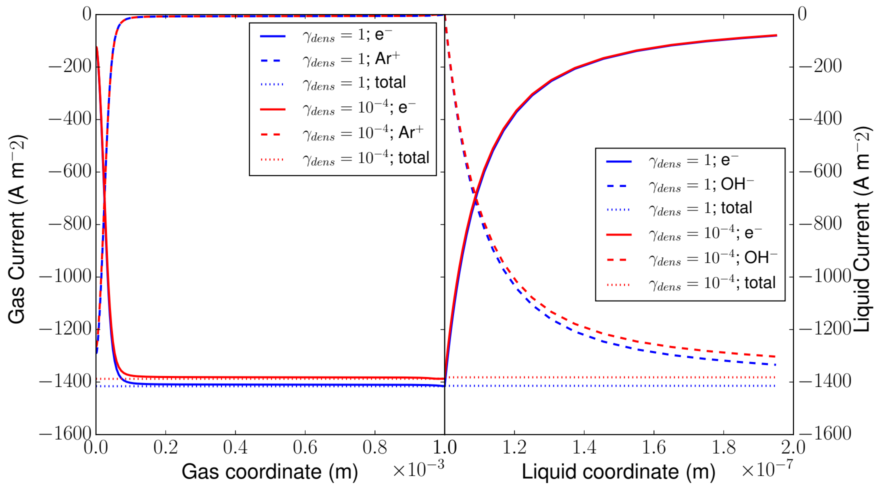


Figure 2.32 Plot of specie and total currents in gas and liquid phases for low and high reflection cases

electron concentration. The net effect is that the total electron flux remains constant.

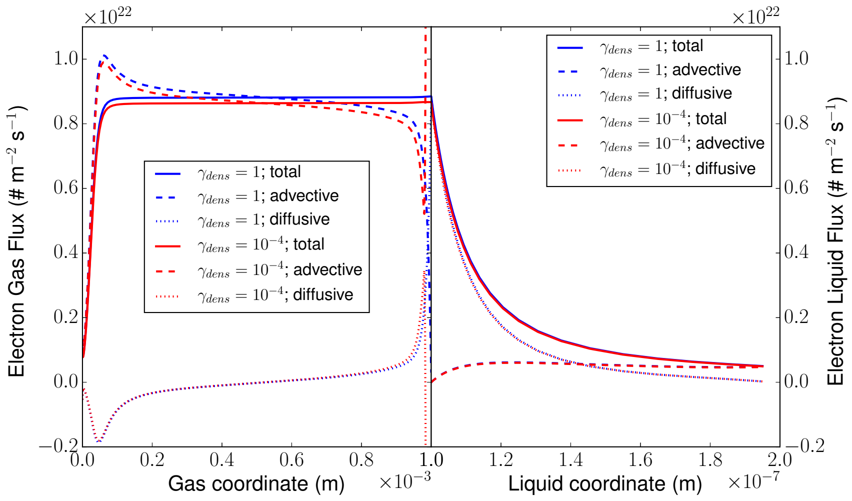


Figure 2.33 Breakdown of electron diffusive and advective fluxes in both domains for low and high reflection cases

The Zapdos plasma fluid model can be compared and tested against PIC calculations conducted by Emi Kawamura. [115] A comparison of ion densities and electron densities for the Zapdos electron energy equation (EEE) model (eqs. (2.9) to (2.16)), a Zapdos local field approximation (LFA) model, and Emi's PIC calculations is shown in figures 2.34 and 2.35. Agreement between the LFA and EEE fluid models is very good. However, the fluid models fail to capture the magnitude or shape of the density profiles in the cathode region predicted by PIC. Disparities in PIC and fluid calculations in the cathode region of atmospheric DC discharges have been observed in previous simulation work.[116, 117, 118] Additionally, the results here are consistent with the knowledge that in the cathode region the drift-diffusion approximation can lead to large errors in density without too seriously affecting the rest of the discharge.[109, 118] Consequently, the difference in PIC and fluid model predictions for the densities in the cathode is not surprising. A difference of a factor of two or less in the bulk densities is not too great and the differences decrease moving away from the cathode.

Comparison electric field profiles computed by EEE, LFA, and PIC models is shown in fig. 2.36. The electric field at the cathode boundary shows excellent agreement among the three models. However, the larger ion density seen in the PIC model in fig. 2.34 leads to a greater decrease in

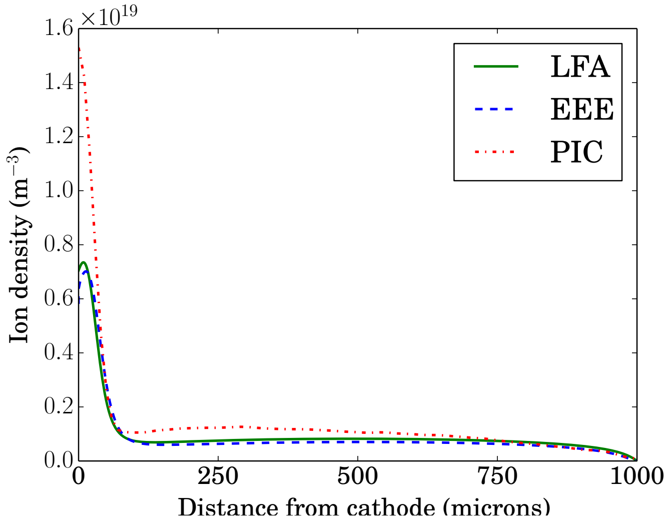


Figure 2.34 Comparison of ion densites computed by Zapdos local field approximation (LFA), Zapdos electron energy equation (EEE), and PIC models

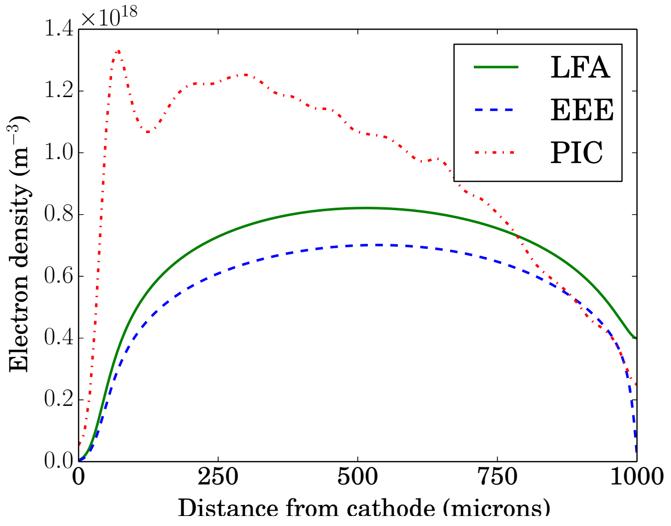


Figure 2.35 Comparison of electron densites computed by Zapdos local field approximation (LFA), Zapdos electron energy equation (EEE), and PIC models

the electric field magnitude as we move through the cathode sheath. Consequently, the bulk electric field magnitude is roughly a factor of two less in the PIC model than in the EEE and LFA fluid models.

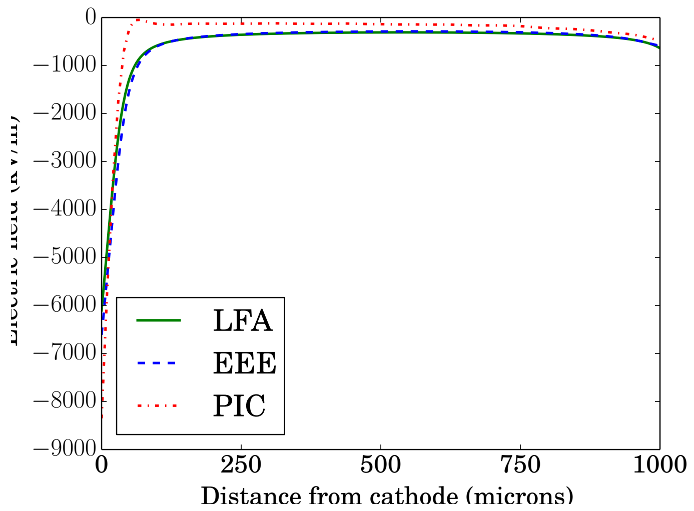


Figure 2.36 Comparison of electric field profiles computed by Zapdos local field approximation (LFA), Zapdos electron energy equation (EEE), and PIC models

Figure 2.37 shows the electron temperature profiles computed by EEE and PIC models. Bulk and anode temperature profiles are in excellent agreement. Cathode profiles differ but this can again be attributed to the inability of the fluid model to accurately capture electron kinetics near the boundary. In summary, as shown by figs. 2.34 to 2.37 the Zapdos fluid models qualitatively reproduce the plasma dynamics predicted by PIC. There are some discrepancies between the models in the cathode region because of the more accurate kinetic treatment provided by PIC. Future work described in more detail in chapter 6 could lead to a hybrid model implemented in Zapdos in which electron behavior in the cathode and perhaps near the water interface is described kinetically while bulk behavior is described with the current EEE fluid model.

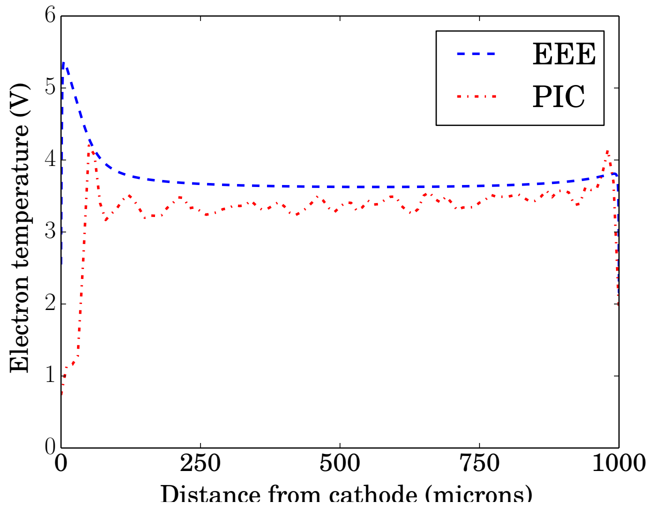


Figure 2.37 Comparison of electron temperature profiles computed by Zapdos electron energy equation (EEE) and PIC models

## 2.3 Summary

This chapter addresses a couple of fundamental questions regarding plasma-liquids. A qualitative study of the momentum, heat, and mass transport in a convective plasma-liquid system has been conducted in section 2.1. Several interesting results were found. Convective flow of water vapor away from the interface leads to a sharp temperature gradient between the bulk gas and liquid phases. This sharp gradient is important for accurately determining temperature-dependent rate coefficients in both the gas and liquid phases. Additionally, convection drives water vapor away from the discharge region except immediately above the liquid surface; this could have important consequences for gas-phase generation of reactive species that depend on water as a precursor. Induced convection in the liquid phase substantially changes the spatial distribution of aqueous species and increases the volume-averaged uptake of hydrophobic species but interestingly has little effect on the volume-averaged uptake of hydrophilic species. This phenomena occurs because the majority of resistance to interfacial transfer is in the gas phase for hydrophilic species and in the liquid phase for hydrophobic species; consequently, decreasing the liquid-phase mass transfer resistance by adding liquid convection has a significant impact for hydrophobic but not for hydrophilic molecules.

Perhaps the most interesting result of the study in section 2.1 is the sharp distinction between reactivity at the surface and in the liquid bulk. Though the limited penetration (tens of $\mu$m or less) of reactive neutral radicals is well publicized in prominent biochemistry texts like [89], this phenomena has yet to receive significant attention in the plasma-liquid literature with the exception of [22]. Concentrations of species of interest for plasma-medicine and other applications (e.g. OH, NO, ONOOH) fall by as many as 9 orders of magnitude within 50 $\mu$m of the interface. In a relatively pure aqueous solution as is modeled here, the process responsible for conveying plasma reactivity to a target may be transport of $H_2O_2$ and acidified nitrite followed by a close-proximity reaction to form ONOOH and then OH and $NO_2$. For targets in aqueous biological systems, the conveyors of reactivity may be proteins modified by primary plasma species. What can be stated from the results presented here and in [22] is that these conveyors of plasma reactivity are probably not neutral radicals generated directly by the plasma; rather they are likely species generated by secondary reactions in the aqueous medium or more stable compounds like $H_2O_2$ and nitrite. To address this problem with even greater precision and detail, future research will involve developing a configuration-specific plasma and electrodynamics model that can be self-consistently coupled to the momentum, heat, and neutral species mass transport presented in this study.

The latter half of the chapter explores the fully self-consistent coupling of discharge physics with the liquid phase. The effects of varying unknown interfacial coefficients like $\gamma_{dens}$ and $\gamma_{en}$ are investigated. It is found that varying these parameters within reason can lead to interfacial gas-phase electron densities varying by orders of magnitude. The interfacial electron energy is also a very sensitive function of the interfacial parameters. This variation of electron density and energy could have significant impacts on both gas and liquid phase chemistry, thus our work motivates future studies to accurately determine the interfacial parameters. It is noteworthy that the overall potential drop and total current are relatively unaffected by changing gas-liquid interfacial parameters.

Self-consistently coupling the liquid domain to the highly non-linear plasma discharge equations as is done for section 2.2 requires an efficient multi-physics code. Comsol, the package used for the simulations in the first half of chapter 2, has typically struggled to efficiently solve plasma problems. Thus a new code, named Zapdos, was built on top of the MOOSE framework for the express purpose of plasma-liquid simulations (although the highly modular nature of the code makes it easily extensible to other low-temperature plasma problems). The structure of Zapdos is the subject of chapter 3.

CHAPTER

3

# ZAPDOS: A TOOL FOR FULLY COUPLED MODELLING OF PLASMA-LIQUID SYSTEMS

In chapter 2 we introduced two models addressing various aspects of plasma-liquids. In section 2.1, we addressed momentum, heat, and neutral species transport using a 2D-axisymmetric model without explicity simulating the plasma discharge. In section 2.2 we examined the discharge physics coupled to the liquid phase in one dimension. The former model was created with the commercially available multi-physics package Comsol. However, efficient solution of the discharge governing equations coupled with liquid phase required creation of a novel plasma-simulation code, Zapdos, based on top of the Multiphysics Object-Oriented Simulation Environment (MOOSE). Creation of the fully-coupled plasma-liquid simulation environment is detailed in this chapter. In section 3.1 we describe Zapdos, the application physics code. In section 3.2 we detail changes implemented in the MOOSE framework allowing physics coupling across domains like that encountered in plasma-liquids.

## 3.1 Zapdos Code

### 3.1.1 Zapdos Intro

It is perhaps prudent to warn the reader that the following subsection contains an overview of a lot of mathematics. We feel that this overview is worthwhile, however, for a couple of reasons. Firstly, the MOOSE application programmer principally responsible for providing two functions in his code: residuals and Jacobians. The following overview defines the meaning and importance of residuals and Jacobians in a physics simulation context. Secondly, the overview should give the reader an idea of the stunning level of control available to the MOOSE application programmer as he assembles and solves his physical simulation. We do not see it this way, but the high degree of control can be regarded as a two-edged sword. Building an accurate and efficient physical simulation on top of the MOOSE framework demands a level of understanding of the underlying mathematics well beyond that of a user of commercial multi-physics software. A minimum of a few months is required before acquiring the knowledge needed to achieve any interesting modelling results. However, once the developer has invested the requisite time and effort, the return is substantial as hopefully demonstrated in section 2.2 and in the exciting possibilities for future research (chapter 6).

Zapdos is built on top of the MOOSE [35] and libMesh [119] codes. MOOSE employs finite element methods (FEM), either Continuous Galerkin, Discontinuous Galerkin, or a combination, to solve fully coupled systems of partial differential equations (PDEs). Physics may also be segregated using the MOOSE multi-app system. After using FEM to discretize the governing equations, MOOSE interfaces with the code PetSc [120] to solve the non-linear or linear system of algebraic equations, $\vec{F}(\vec{u})$, via some form of Newton's method, where $\vec{F}$ is the residual vector and $\vec{u}$ is the vector of unknowns. A block diagram showing the interplay between Zapdos, MOOSE, libMesh, PetSc, as well as meshing and visualization packages (discussed more in sections 3.1.7 and 3.1.8) is shown in fig. 3.1. We will outline briefly the concept of forming the residual in the context of the finite element method. Let us consider the following differential equation:

$$\nabla \cdot -D\nabla u = s(\vec{r}) \tag{3.1}$$

where diffusion of species $u$ is balanced by a source term $s(\vec{r})$. We refer to eq. (3.1) as the strong form of our reaction-diffusion example problem. For FEM we convert the problem to what is known as the weak form by multiplying by a test function $\psi_i$ and integrating over the whole domain. We also move all terms to the left-hand side (LHS) of the equation such that the right-hand side (RHS) becomes 0. Our problem now becomes:

$$\int_\Omega \psi_i \nabla \cdot -D\nabla u \, dV - \int_\Omega \psi_i s(x) \, dV = 0 \tag{3.2}$$

where $\Omega$ represents the extent of the domain. The first term is then integrated by parts to gives us:

$$\int_\Omega \nabla\psi_i \cdot D\nabla u \, dV - \int_{\partial\Omega} \psi_i D\nabla u \cdot \vec{n} \, dS - \int_\Omega \psi_i s(x) \, dV = 0 \tag{3.3}$$

where $\partial\Omega$ represents the domain boundaries. Integrals over the domain, e.g. terms 1 and 3 in eq. (3.3), are written in Zapdos and MOOSE as kernel class objects. Integrals over domain boundaries are written as boundary condition class objects. We will discuss Zapdos class objects more shortly. To continue our development of FEM, we must discretize the problem, or in other words convert our integral-differential equation into an algebraic equation. We accomplish this by dividing the domain volume into individual elements and associating basis functions with different geometric aspects of the discretization. Division of the domain into elements is also referred to as meshing. An example of a two-dimensional mesh is shown in fig. 3.2, where the domain has been sub-divided into triangular elements. For a two-dimensional domain, basis functions can be associated with mesh elements, edges, or vertices; in three-dimensions: elements, faces, edges, and vertices. Most commonly, Zapdos uses a linear basis associated with mesh vertices; these basis functions are commonly referred to as linear Lagrange. We construct a basis function $\phi_j(x, y, z)$ for each vertex that exists on the mesh (j = 1, 2, ..., N where N is the number of vertices in the domain). The function $\phi_j$ has these important properties: it is equal to one on vertex $j$ with coordinates $(x_j, y_j, z_j)$ and zero on all other vertices. This is illustrated in fig. 3.3 for a vertex sitting at the intersection of four two-dimensional triangular elements.

The dependent variable $u$ is obtained by summing over the whole basis:

$$u = \sum_{j=1}^{N} u_j \phi_j(x, y, z) \tag{3.4}$$

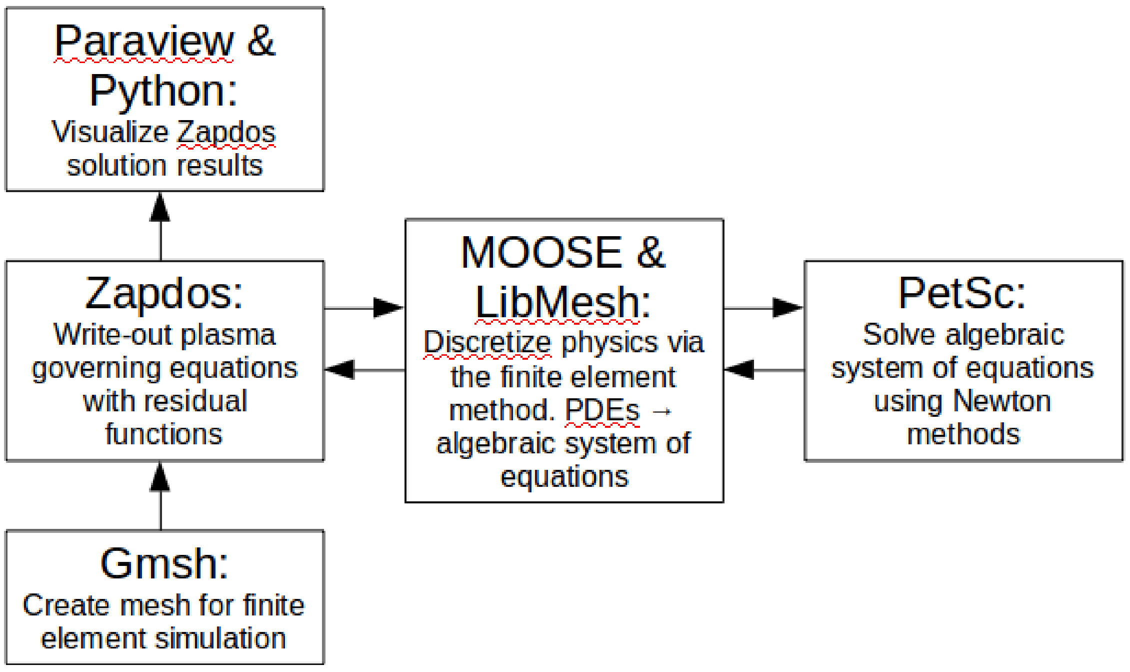

Paraview & Python:
Visualize Zapdos solution results
Zapdos:
Write-out plasma governing equations with residual functions
MOOSE & LibMesh:
Discretize physics via the finite element method. PDEs → algebraic system of equations
PetSc:
Solve algebraic system of equations using Newton methods
Gmsh:
Create mesh for finite element simulation

Figure 3.1 Block-diagram laying out simulation work-flow

where $u_j$ is the value of u at vertex $j$. The set of $u_j's$ represent the degrees of freedom or the quantities to be solved for. It is worthwhile to note that in Galerkin FEM, which is the only method used in Zapdos, the following relationship exists between basis and test functions: $\phi_k = \psi_k$ for k = 1, 2, ..., N. Substitution of eq. (3.4) into eq. (3.3) yields

$$\int_\Omega \nabla\psi_i \cdot D\nabla \sum_{j=1}^{N} u_j\phi_j(x,y,z)\,dV - \int_{\partial\Omega} \psi_i D\nabla \sum_{j=1}^{N} u_j\phi_j(x,y,z)\cdot\vec{n}\,dS - \int_\Omega \psi_i s(x)\,dV = R_i \approx 0 \tag{3.5}$$

where we have introduced the idea that our governing equations form residual statements ($R_i$) that we hope to eventually drive to zero. The first and second terms of eq. (3.5) arising from diffusion can be analytically integrated. To complete the transformation of eq. (3.5) into an algebraic equation, the source term integral is computed with numerical quadrature. MOOSE employs Guassian quadrature, which for a general integrand s(x), can be written as:

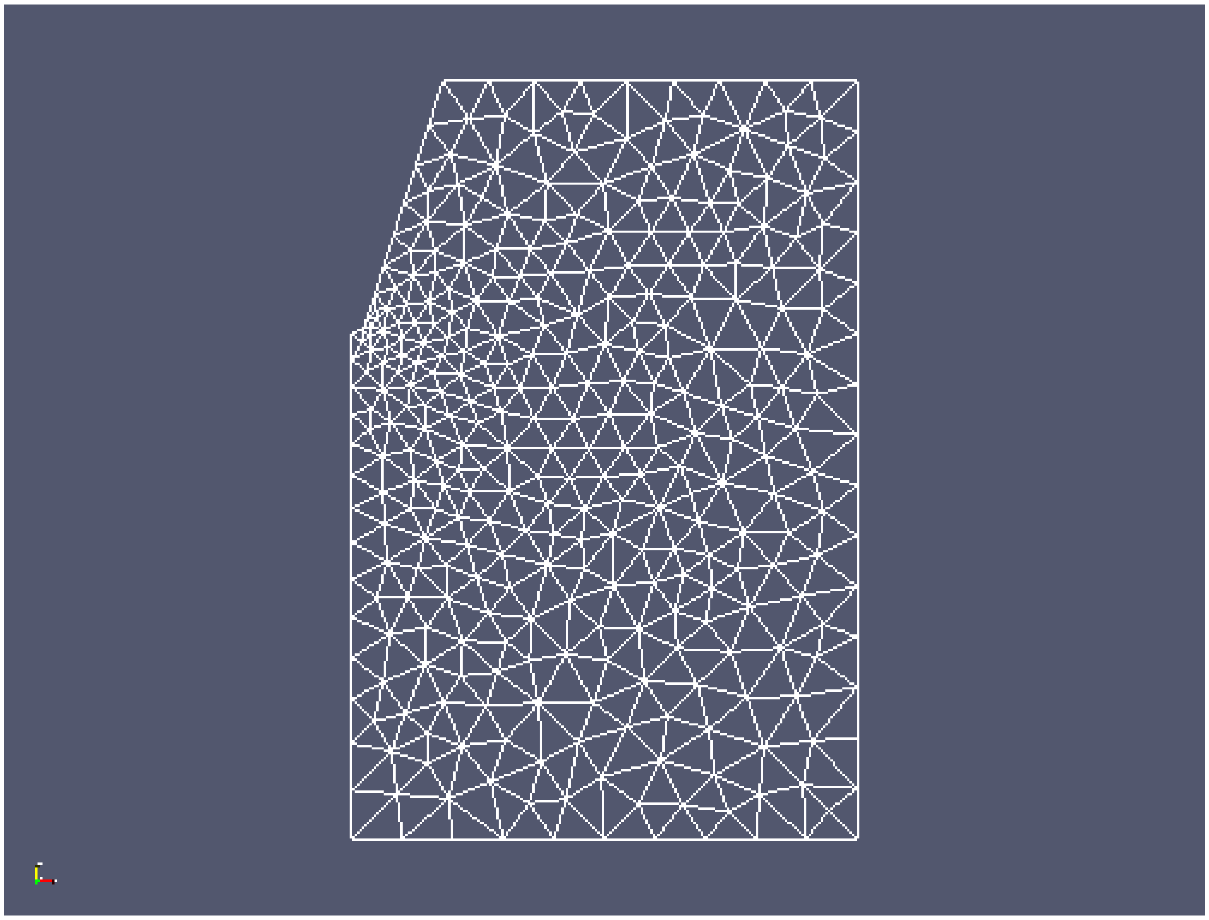


Figure 3.2 An example two-dimensional mesh. The domain is sub-divided into triangular elements.

$$\int_a^b s(x)\,dx = \frac{b-a}{2}\sum_{i=1}^{n} w_i s\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right) \tag{3.6}$$

where the $w_i's$ and $x_i's$ are the quadrature weights and points respectively. Details of calculation of $w_i$ and $x_i$ are not important to our purpose here; it is enough to know the concept of how a general integral is converted into an algebraic expression. After numericall integrating eq. (3.5), the resulting algebraic expression can be evaluated using an initial guess for the $u_j's$. This results in the initial residual, with the residual essentially representing how close the $u_j's$ are to satisfying the governing equations; a residual of zero would represent a numerically perfect solution. In practice, the initial guess may be far from the actual solution. To rectify this, Zapdos, through

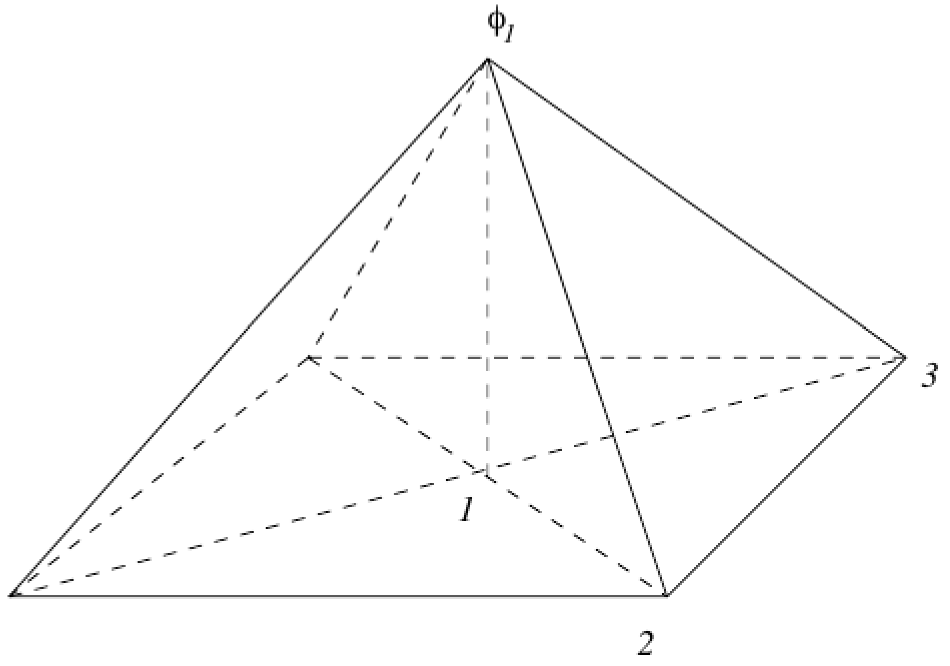


Figure 3.3 Basis function $\phi_1$ associated with vertex one. Vertex one sits at the intersection of four triangular two-dimensional elements. Note that $\phi_1$ equals unity at $(x_1, y_1)$ and zero at all other vertices. Illustration taken from [4]

MOOSE's interface with PetSc, uses Newton's method to iterate the solution vector $\vec{u}$ towards the true solution. Newton's method derives from a Taylor expansion: [121]:

$$\vec{R}(\vec{u}_{k+1}) = \vec{R}(\vec{u}_k) + \tilde{R}'(\vec{u}_k)(\vec{u}_{k+1} - \vec{u}_k) + \text{higher-order terms} \tag{3.7}$$

where in our case $\vec{R}$ is the residual vector of length N and $\vec{u}$ is the solution vector, also of length N where N in our example problem is equal to the number of mesh vertices as well as the number of basis functions. The index $k$ denotes the $k^{th}$ iteration of the Newton solve. In a more general case we may be solving for multiple fields, e.g. in addition to a concentration variable $u$ we may be solving for a temperature variable $T$. In this more general case (still limiting ourselves to a linear Lagrange discretization), $\vec{R}$ and $\vec{u}$ will have dimension equal to the # of mesh vertices times the # of field variables with $\vec{u}$ comprised of $T_j's$ and $u_j's$. (Indices $j$ and $k$ should not be confused; $u_j$ is one scalar element of the solution vector $\vec{u}$; $\vec{u}_k$ is the $k^{th}$ iterate of the solution vector computed during the Newton solve.) Returning to eq. (3.7), if we set the left-hand side to zero (our goal is always to drive $\vec{R}_{k+1}$ to zero) and neglect the higher-order terms we reproduce the strict Newton method:

$$\tilde{J}(\vec{u}_k)\vec{\delta u}_k = -\vec{R}(\vec{u}_k) \tag{3.8}$$

$$\vec{u}_{k+1} = \vec{u}_k + \vec{\delta u}_k \tag{3.9}$$

where $\tilde{J}$ is the Jacobian matrix formed by taking the derivatives of the residual vector with respect to the solution vector (equivalent to $\tilde{R}'$ in eq. (3.7)).[121] We iterate with eqs. (3.8) and (3.9) until the norm of $\vec{R}$ is below some user-defined tolerance. Strict Newton is known to have locally quadratic convergence; e.g. if the initial guess is relatively close to the solution, Newton's method will converge quickly and with ease. [122] In practice, however, the initial guess may not be close the solution. In this case a globalization method is used to increase the rate of convergence. Line search and trust region methods are the most common techniques for globalization. The default globalization in MOOSE applications is a line search with cubic backtracking. In a line search, eq. (3.9) is modified by simply inserting a multiplicative factor $\lambda$ in front of $\vec{\delta u}_k$ as shown in eq. (3.10).

$$\vec{u}_{k+1} = \vec{u}_k + \lambda\vec{\delta u}_k \tag{3.10}$$

Determining $\lambda$ is the realm of the particular line search technique chosen. The cubic backtracking technique for chosing $\lambda$ that is used in all Zapdos simulations is described in detail in [122]. We will give a short summary here. For simplicity, let us consider minimizing the residual $R$ of one nonlinear equation with one degree of freedom, $u$. We use the index $k$ to denote the $k^{th}$ Newton iteration and we introduce the index $l$ to denote the $l^{th}$ attempt to select a good value for $\lambda$. Since Newton's method displays exceptional convergence when $u_k$ is sufficiently close to the true solution, the zeroth guess for $\lambda$ is always $\lambda_0 = 1$. If $\lambda_0$ does not yield a desirable result (see eq. (3.13), then $\lambda_1$ is determined through quadratic backtracking (for details see [122]). If $\lambda_1$ also fails, the line search algorithm has sufficient information to perform a cubic backtrack. Cubic backtracking has the benefit over quadratic backtracking of being able to minimize $R$ when $R$ has negative curvature in the vicinity of $u_k$. Cubic backtracking minimizes the equation:

$$m_{cu}(\lambda_l) = a\lambda_l^3 + b\lambda_l^2 + J(u_k)\delta u_k\lambda_l + R(u_k) \tag{3.11}$$

where J is our one-dimensional Jacobian equal to $\frac{\partial R}{\partial u}$ and $\delta u_k$ is the full Newton step determined from eq. (3.8); $a$ and $b$ are given by: [122]

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\lambda_{l-1} - \lambda_{l-2}} \begin{bmatrix} \frac{1}{\lambda_{l-1}^2} & \frac{-1}{\lambda_{l-2}^2} \\ \frac{-\lambda_{l-2}}{\lambda_{l-1}^2} & \frac{\lambda_{l-1}}{\lambda_{l-2}^2} \end{bmatrix} \begin{bmatrix} R(u_k + \lambda_{l-1}\delta u_k) - R(u_k) - J(u_k)\delta u_k \lambda_{l-1} \\ R(u_k + \lambda_{l-2}\delta u_k) - R(u_k) - J(u_k)\delta u_k \lambda_{l-2} \end{bmatrix} \tag{3.12}$$

The stopping criterion used for $\lambda$ is: [122]

$$R(u_k + \lambda_l \delta u_k) \leq R(u_k) + \alpha \lambda_l J(u_k)\delta u_k \tag{3.13}$$

with $\alpha$ chosen to be some value between 0 and 1; the default in PetSc is $\alpha = 10^{-4}$. When the criterion of eq. (3.13) is met, then the solution $u$ is updated through eq. (3.10). Equation (3.13) essentially requires that the average rate of decrease in R when moving from from $u_k$ to $u_k + \lambda_l \delta u_k$ be some fraction of the initial rate of decrease in that direction. The combination of eqs. (3.8) and (3.10) is properly called a quasi-Newton method since the full Newton step is not necessarily used to update the solution vector at each iteration. However, since a globalization method like the line search described above is always combined with eq. (3.8) in Zapdos simulations and any other practical Newton implentation, we will drop the quasi- prefix for the remainder of this discussion.

The linear problem for determining $\delta u_k$, Equation (3.8), may be solved through either direct or iterative methods. The details of these methods are not critical to understanding the work in this dissertation, so we will give only a brief summary of them here. A direct method like lower-upper decomposition works well for relatively small problems. However, larger problems, particularly problems in three dimensions, require iterative Krylov subspace methods to be feasible. [26] The most common Krylov technique and the one used by default in Zapdos is the generalized minimum residual method (GMRES) with restart. GMRES is an Arnoldi-based method; its algorithm is given in [123]. The option to restart the GMRES algorithm prevents the inherent problem that as the number of linear iterations $k$ increases, the number of multiplications required to solve the linear system increases by $\frac{1}{2}k^2 N$ where N denotes the number of degrees of freedom being solved for. [123] The GMRES algorithm only requires knowledge of matrix vector products to complete the iteration, e.g. the Jacobian solution vector product on the LHS of eq. (3.8), as opposed to individual elements of the matrix. [121] This makes GMRES conducive for a class of methods known as Jacobian-free Newton-Krylov (JFNK) methods, which will be discussed more shortly. Convergence of iterative methods like GMRES tends to improve as the condition number of the matrix $\tilde{A}$ in the equation $\tilde{A}\vec{x} = \vec{b}$ decreases. The condition number can be decreased through a process called preconditioning. Right preconditioning solves the new problem: $\tilde{A}\tilde{P}^{-1}\tilde{P}\vec{x} = \vec{b}$; left preconditioning solves problem $\tilde{P}^{-1}(\tilde{A}\vec{x} - \vec{b}) = 0$. The preconditioning matrix $\tilde{P}$ is usually based on the matrix $\tilde{A}$ or in our case the Jacobian matrix $\tilde{J} = \tilde{A}$. In creating $\tilde{P}$, there is a balance to strike between the degree of conditioning and the computational cost of applying the preconditioner. Choosing $\tilde{P} = \tilde{J}$ results in a conditioned matrix $\tilde{P}^{-1}\tilde{J} = \tilde{I}$ with an optimal condition number of one; solving the resulting conditioned system will take only one iteration to converge. However, the cost of applying the preconditioner is at a maximum. In practice, $\tilde{P}$ is much more sparse than $\tilde{J}$ and can be formed through a variety of techniques. The default serial preconditioning method for iterative linear solutions with Zapdos is incomplete lower-upper factorization which is described in detail in [124] and [125]. Other preconditioning techniques are more parallelizable including block-Jacobi, which only fills the diagonal elements of the preconditioning matrix; [125] the domain decomposition technique, additive Schwarz, which operates under the principle of divide and conquer; [125] and multi-grid methods that rely on mesh restriction and prolongation operators. [121] We recommend consulting this passage's references for more details on the mentioned preconditioning techniques.

When analytic calculation of Jacobian elements becomes either impossible or impractical, the property that GMRES only requires knowledge of matrix-vector products as opposed to individual matrix elements becomes very useful. It allows the modeller to implement the Jacobian-free approximation: [121]

$$\tilde{J}\vec{v} \approx \frac{\vec{R}(\vec{u} + \epsilon\vec{v}) - vecR(\vec{u})}{\epsilon} \tag{3.14}$$

where $\epsilon$ is a finite-differencing parameter chosen by the modeller. Using this approximation, the elements of the Jacobian matrix are never explicitly formed, giving the algorithm its "Jacobian-free" moniker. Using JFNK, one achieves Newton-like nonlinear convergence without the need for computing or storing the true Jacobian. [121] Through finite differencing of the residual, JFNK feels out the true Jacobian, generally giving better convergence than if a Newton-Krylov method is used in conjunction with an incomplete or incorrect user-provided Jacobian matrix. Although JFNK methods are sometimes referred to as "matrix-free" methods, this is somewhat of a misnomer. In practice JFNK almost always involves forming a preconditioning matrix since the linear solve is intrinsically iterative.

In the precending section, we have hopefully illustrated the generation and importance of residual and Jacobian statements in simulating our physical system. To summarize, residual statements are pieces of the physical governing equations cast in the weak form and discretized with the finite element method. The Jacobian matrix $\tilde{J}$ is composed of derivatives of the residuals with respect to the solution vector $\vec{u}$. A maximally efficient application code in terms of computational time will have a complete and correct set of Jacobian statements and will employ a Newton method globalized with a line search. Small problems with accurate Jacobians can be solved directly, via lower-upper factorization for example; however, higher-dimensional problems may require a preconditioned iterative method to solve the linear system in eq. (3.8). If developer time is at a premium or some residual derivatives cannot be computed analytically, some Jacobian statements can be omitted and the Jacobian-Free approximation of eq. (3.14) can be used. While still efficient, this method displays slower convergence than if a fully analytic and accurate Jacobian is supplied. Thus, the low-temperature plasma application Zapdos supplies complete and correct Jacobian statements wherever ot can. As new physics are introduced into Zapdos, newly coded analytical Jacobians are compared against PetSc Jacobians formed through finite differencing of the residual statements to ensure accuracy. The one-dimensional simulations run in section 2.2 featured fully accurate Jacobians; as a result, Newton's method combined with a direct solve of the linear system displayed the best rate of convergence. New 2-D axisymmetric simulations not reported in this work feature boundary condition residuals with non-local integrals. The coding infrastructure required to implement the corresponding Jacobian functions does not currently exist in MOOSE. Since the Jacobian is not completely accurate in this case, direct Newton and preconditioned JFNK display comparable rates of convergence.

Having given an overview of the mathematics that Zapdos relies on, we will turn to its coding implementation. Zapdos partitions governing equation terms into individual pieces called kernels. Each kernel contains the residual and the corresponding Jacobian statements. Recall our diffusion-reaction example from eqs. (3.1) to (3.3) and (3.5). Let us consider the first term in eq. (3.3); the corresponding Zapdos code looks like:

```
Real
CoeffDiffusionLin::computeQpResidual()
{
  // Computes the residual
  return -_diffusivity[_qp] * _grad_u[_qp] * _r_units * -_grad_test[_i][_qp] *
```

```
    _r_units;
}

Real
CoeffDiffusionLin::computeQpJacobian()
{
  // Computes the Jacobian
  return -_diffusivity[_qp] * _grad_phi[_j][_qp] * _r_units * -_grad_test[_i][_qp
    ] * _r_units;
}
```

Listing 3.1 Example of residual and Jacobian function definitions

where _diffusivity is the diffusion coefficient, _u is our solution variable, _phi and _test represent the shape and test functions that we introduced above, and _qp represent the positions of quadrature points used for numerical integration. By splitting governing equations in this way into individual terms/kernels, code reproduction is kept at a minimum; analagous terms can be used in many different settings, e.g. a “diffusion” term has the exact same mathematical form as a “conduction” or “viscosity” term and so the same kernel code can be used for all three physics cases. Material properties like mobilty and diffusivity are defined in a materials file separated from the kernel code. Material properties can be defined as constants, as functions of the solution variables, or as properties to be read from look-up tables. Through MOOSE, Zapdos provides an interface for linear, bilinear, and spline interpolation of material properties. Boundary conditions are available in “Nodal” and “Integrated” flavors. Nodal boundary conditions are dirichlet like conditions that are enforced strongly. Integrated boundary conditions are cast in the weak form and often arise from performing integration by parts on divergence terms in the governing equations.

As of commit f74bad6, Zapdos has the necessary kernels and boundary conditions for solving gas phase DC discharge fluid models as well as conventional convection-diffusion-reaction equations for dilute species in a fluid. (Zapdos uses the software “git” for version control. The commit “hash” f74bad6 is essentially a version number.) Another student is working on implementing RF plasma simulation capabilities (for capacitively coupled plasmas this will only require slight modification of some boundary conditions; inductively coupled plasmas will require a little more work). These efforts will be detailed further in chapter 6.

Zapdos solutions are output to an exodus file by default, although MOOSE provides varying levels of support for some other output file formats (including full support for simple CSV).

Exodus files are a file type developed at Sandia National Lab designed specifically for storing and retrieving finite element data. [126] These exodus files are most commonly viewed graphically with either of the open source packages Visit or Paraview. For users more programatically inclined, Paraview provides python tools that enable the user to directly read the exodus file and create publication level plots in MatPlotLib with a single script (as is done for a lot of figures in this dissertation). For transient simulations, results for any solution or auxiliary variable can be viewed while the calculation is on-line. Results are also not lost if a solve is cancelled for any reason. These features enable quick convergence debugging of a failing or failed solve.

Another feature of Zapdos is the adaptive mesh refinement inherited from MOOSE. The user can choose from several different indicators, including the jump in a solution gradient or laplacian between elements, for determing where mesh refinement should take place. Figures 3.4 and 3.5 show the results of an advection-diffusion simulation for temperature in which a pressure difference between the left and right ends of the domain induce bulk flow from left to right. The effective mobility of the temperature is specified to be greater in the bottom half of the domain, resulting in faster temperature flow in the bottom half. Note that the mesh is refined at the head of the temperature flow where numerical instabilities are more likely to occur; once the temperature front has passed the mesh is coarsened to reduce computational expense. This feature can be incredibly useful when trying to track ionization bullets (fig. 3.6) or similar phenomena.

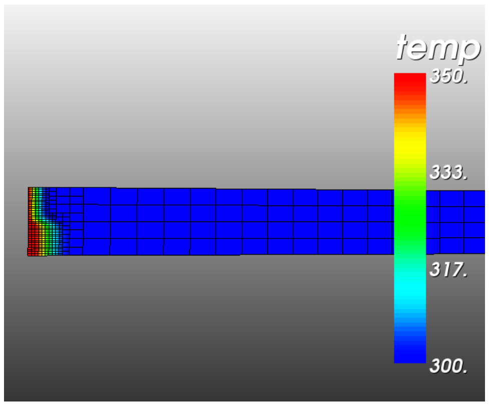


Figure 3.4 Propagating front. Time step 15. Note how the mesh is fine around the solution gradients and coarse elsewhere.

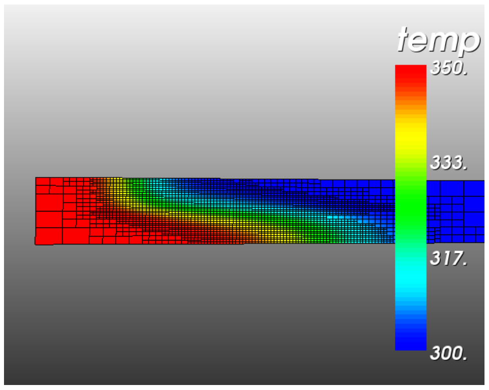


Figure 3.5 Propagating front. Time step 49. Note how the mesh is fine around the solution gradients and coarse elsewhere.

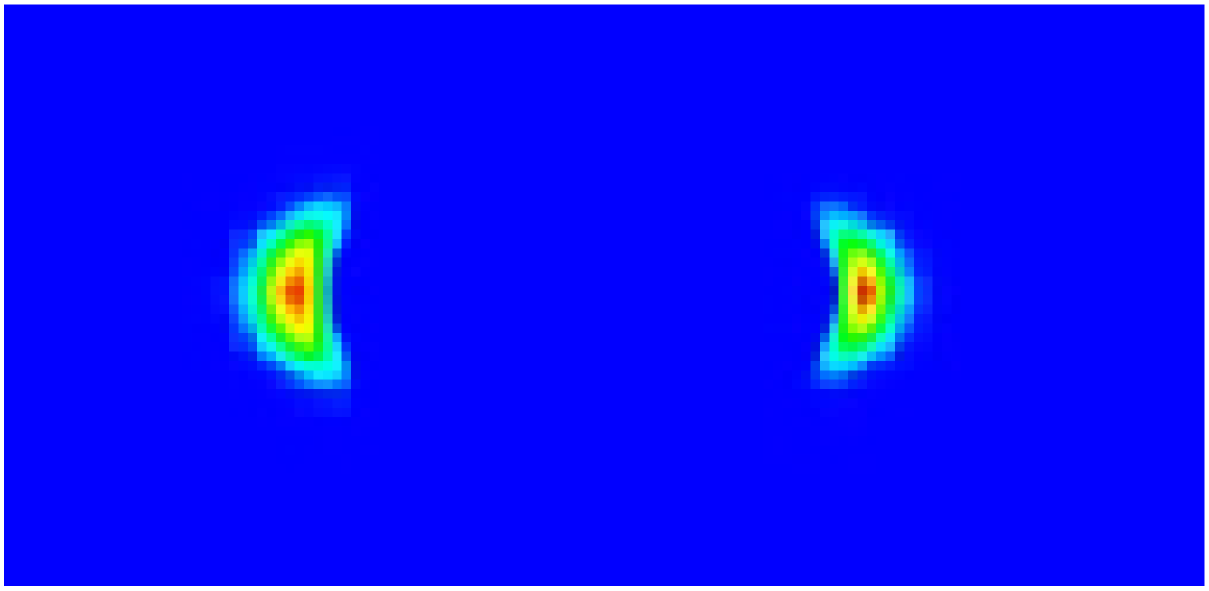


Figure 3.6 Ionization bullets simulated with Zapdos. Mesh adaptivity is used to follow their propagation.

### 3.1.2 Zapdos Kernels

As mentioned in the introductory section, Zapdos takes each term in a governing equation and casts that term as a class with methods for computing the residual and Jacobian. These governing equation term classes are called kernels. As of commit f74bad6, Zapdos has 77 kernels. However, not all of these have utility, e.g. one kernel may have been re-cast as a new class without the old class being removed from the kernel directory. The most important kernels, e.g. the ones actively being used for physics and engineering research, are enumerated below. An important feature of Zapdos is the option to cast concentration or density variables in a logarithmic form, e.g. $N_k = \ln(n_k)$ where $N_k$ is the logarithmic variable representation of the density and $n_k$ is the true physical density. This is done for the modelling studies presented in section 2.2. The advantage of the logarithmic casting is that it prevents the true concentration from ever becoming negative. Negative concentrations can be a product of and contribute to numerical instabilities. Negative concentrations can cause source terms to become sink terms and visa versa, thus it is advantageous to avoid them if possible.

Also for all the simulations described in section 2.2, it is the logarithm of the *product* of the electron density and mean energy that is a solution variable as opposed to simply the logarithm of the mean energy. Thus for the simplest plasma discharge simulation, there are four solution variables:

$$N_i = \ln n_i \tag{3.15}$$

$$N_e = \ln n_e \tag{3.16}$$

$$E_n = \ln(n_e \epsilon) \tag{3.17}$$

$$V = V \tag{3.18}$$

where n$_i$ and n$_e$ are the physical ion and electron densities respectively, $\epsilon$ is the mean electron energy, and V is the potential. Anywhere that $n_i$ exists in the governing equations, it is replaced with $e^{N_i}$; $n_e$ is replaced with $e^{N_e}$; the product of $n_e$ and $\epsilon$ is replaced with $e^{E_n}$. Whatever units are used for the original variables are retained by their replacement expressions, e.g. if $n_e$ has units of #/m$^3$ then the expression $e^{N_e}$ has units of #/m$^3$. While the choice to use the product of n$_e$ and $\epsilon$ simplifies some parts of the governing equation, it complicates others. In particular the electron transport and Townsend coefficients that are functions of the mean electron energy become functions of two solution variables, $\mathrm{N}_e$ and $\mathrm{E}_n$, as opposed to just one. Thus residual/governing equation terms that involve electron transport or Townsend coefficients must include Jacobian contributions from both $\mathrm{N}_e$ and $\mathrm{E}_n$.

In order to improve conditioning of the Jacobian, Zapdos provides the user options for unit scaling. The potential can either be cast in units of volts or kilovolts (this will be made even more flexible in the future). This choice is made in the GlobalParams block of the Zapdos input file, by specifying $potential\_units = V$ or $potential\_units = kV$. The length units can also be scaled. In the coupled plasma-liquid simulations described in section 2.2, the plasma domain length is 1 mm, whereas the water domain length is 100 nm. Thus at the beginning of the input file, in order to scale closer to unity, we specify $dom0scale = 1e-3$ and $dom1scale = 1e-7$. The values of $dom0Scale$ and $dom1Scale$ can then be accessed using the syntax, $\$\{\}$. [127] Thus for any kernel or boundary condition that contains a gradient term, we specify the parameter: $position\_units = \$\{dom0Scale\}$ or $position\_units = \$\{dom1Scale\}$ depending on whether the kernel or boundary condition is acting in the gas or liquid phase. Unlike the $potential\_units$ parameter which must be a string equal to $V$ or $kV$, the $position\_units$ parameter can be set to any real number. The length scaling is represented in table 3.1 as the symbol $l_c$ where $l_c$ is actually equal to $1/position\_units$; the potential scaling is represented by $V_c$ where $V_c = 1000$ if $potential\_units = kV$ else $V_c = 1$ if $potential\_units = V$.

Zapdos tries to be as generic and modular as possible in the formulation of its kernels, e.g. a diffusion kernel code should be as applicable to the diffusion of Argon ions in the gas phase as it is to neutral OH radicals in the liquid phase. However, advection-diffusion-reaction (ADR) terms for electrons in the plasma typically have to have their own kernels because the transport and Townsend coefficients are allowed to be functions of the mean electron energy as opposed to constants. In this case Jacobian terms must be provided for the coefficient functional dependence on both $\mathrm{N}_e$ and $\mathrm{E}_N$. This functional dependence does not exist when the coefficients are constant as in the cases where we are modelling transport of heavy ions and neutrals, thus those kernels can be completely generic. For generic kernels, the solution variable is denoted by $u$ in table 3.1; a coupled solution variable is denoted by $v$. Note that the kernel residuals/governing equation terms are cast in the weak form, i.e. each term in the governing equation is multiplied by a test function $\psi_i$ where $i$ denotes the $i^{th}$ shape function. Divergence terms, e.g. flux terms, are integrated by parts to produce both a volumetric kernel term and a surface boundary condition term (or surface discontinuous Galerkin term if a discontinuous Galerkin discretization is used). $\mu$ is the mobility of the species the kernel is acting on, $D_k$ is the diffusivity, $l_c$ is the characteristic length defined in the paragraph above, $\alpha_{iz}$, $\alpha_{ex}$, and $\alpha_{el}$ are the Townsend ionization, excitation, and elastic collision coefficients respectively, $sgn(q)$ is the charge sign, $k$ is used to generally represent reaction coefficients, $N_A$ is Avogadro's number, and $e$ is the Coulombic charge equal to $1.6x10^{-19}$ C. All other symbols should be defined in their corresponding table entry.

Table 3.1 Kernels in Zapdos used for simulations presented in section 2.2

| **Kernel Name** | **Governing Eqn. Term**[a] | **Description** |
|---|---|---|
| ElectronTimeDerivative | $\psi_i e^u \frac{\partial u}{\partial t}$ | Generic accumulation term |
| EFieldAdvectionElectrons | $-\nabla\psi_i \mu_e(N_e, E_n) e^{N_e} \nabla V / l_c^2$ | Electron specific electric field driven advection term |
| CoeffDiffusionElectons | $\nabla\psi_i D_e(N_e, E_n) e^{N_e} \nabla N_e / l_c^2$ | Electron specific diffusion term |
| ElectronsFromIonization | $-\psi_i \alpha_{iz}(E_n, N_e)\lvert\vec{\Gamma_e}\rvert$ | Rate of production of electrons from ionization |
| LogStabilizationMoles | $-\psi_i e^{-(b+u)}$ | Kernel stabilizes solution variable $u$ in places where $u \rightarrow 0$; $b$ is the offset value specified by the user. A typical value for $b$ is 20. |
| EFieldAdvection | $-\nabla\psi_i \mu\, \mathrm{sgn}(q) e^u \cdot -\nabla V / l_c^2$ | Generic electric field driven advection term |
| CoeffDiffusion | $-\nabla\psi_i - D_e e^u \nabla u / l_c^2$ | Generic diffusion term |
| ReactantFirstOrderRxn | $\psi_i k e^u$ | Generic first order reaction sink term for $u$ ($u$ is the reactant); $k$ is the reaction rate coefficient |

Table 3.1 Continued

| **Kernel Name** | **Governing Eqn. Term**[a] | **Description** |
|---|---|---|
| ReactantAARxn | $2\psi_i k e^{2u}$ | Generic second order reaction sink term for $u$ in which two molecules of $u$ are consumed |
| CoeffDiffusionLin | $-\nabla\psi_i \cdot -D\nabla u/l_c^2$ | Generic *linear* diffusion term, e.g. this is a diffusion term for solution variables *not* cast in a logarithmic form |
| ChargeSourceMoles_KV | $\frac{-\psi_i e\,\mathrm{sgn}(q) N_A e^v}{V_c}$ | Used for adding charged sources to Poisson's equation; $e^v$ represents the charged particle density of species $v$. This kernel assumes that densities are measured in units of mol/volume as opposed to #/volume. |
| IonsFromIonization | $-\psi_i \alpha_{iz}(E_n, N_e)\lvert\vec{\Gamma}_e\rvert$ | Same governing term/residual as ElectronsFromIonization; however, the Jacobian structure is different. $\frac{\partial R_i}{\partial N_e}$ will be on-diagonal for ElectronsFromIonization and off-diagonal for IonsFromIonization |

Table 3.1 Continued

| **Kernel Name** | **Governing Eqn. Term**[a] | **Description** |
|---|---|---|
| ProductFirstOrderRxn | $-\psi_i k e^{v}$ | Generic first order reaction source term for $u$ ($v$ is the reactant) |
| ProductAABBRxn | $-2\psi_i k e^{2v}$ | Generic second order reaction source term in which two molecules of $v$ are produced from two molecules of $u$ |
| EFieldAdvectionEnergy | $-\nabla\psi_i \mu_\epsilon(N_e, E_n) e^{E_n} \cdot \nabla V/l_c^2$ | Electron energy specific electric field driven advection term |
| CoeffDiffusionEnergy | $-\nabla\psi_i \cdot -D_\epsilon(N_e, E_n) e^{E_n} \nabla E_n/l_c^2$ | Electron energy specific diffusion term |
| JouleHeating | $-\psi_i \nabla V V_c/l_c \cdot \vec{\Gamma_e}$ | Joule heating term for electrons |
| ElectronEnergyLossFromIonization | $\psi_i \alpha_{iz}(N_e, E_n)\lvert\vec{\Gamma_e}\rvert E_{iz}$ | Electron energy loss term for inelastic ionization collisions; $E_{iz}$ is the energy lost in Volts in a single ionization collision |
| ElectronEnergyLossFromExcitation | $\psi_i \alpha_{ex}(N_e, E_n)\lvert\vec{\Gamma_e}\rvert E_{ex}$ | Electron energy loss term for inelastic excitation collisions; $E_{ex}$ is the energy lost in Volts in a single excitation collision |

Table 3.1 Continued

| **Kernel Name** | **Governing Eqn. Term**[a] | **Description** |
|---|---|---|
| ElectronEnergyLossFromElastic | $\psi_i \alpha_{el}(N_e, E_n) \lvert \vec{\Gamma_e} \rvert \frac{3 m_e T_e}{m_n}$ | Electron energy loss term for elastic collisions. $\alpha_{el}$ is the elastic Townsend coefficient; $m_e$ is the electron mass; $m_n$ is the mass of the neutral background gas; $T_e = \frac{2\epsilon}{3}$ is the electron temperature |

[a] $\vec{\Gamma_e} = \mu_e(N_e, E_n) \nabla V l_c e^{N_e} - D_e(N_e, E_n) e^{N_e} \nabla N_e l_c$

### 3.1.3 Zapdos Auxiliary Kernels

Zapdos implements a variety of auxiliary kernels that, while not essential to the solve, are important for visualizing and understanding the plasma-liquid physics. The auxiliary kernels that are employed for the simulations in section 2.2 are outlined in table 3.2.

Table 3.2 AuxKernels in Zapdos used for visualization of simulation results described in section 2.2

| **AuxKernel Name** | **Expression** | **Description** |
|---|---|---|
| PowerDep | $\text{sgn}(q) e N_A \cdot (\text{sgn}(q)\mu \cdot -\nabla V e^{N_k} - D e^{N_k} \nabla N_k) \cdot -\nabla V V_c / l_c^2$ | Amount of power deposited into a user specified specie by Joule Heating |
| ProcRate | $N_A \cdot \left\lvert -\mu_e \cdot -\nabla V e^{N_e} - D_e e^{N_e} \nabla N_e \right\rvert \cdot \alpha / l_c$ | Reaction rate for electron impact collisions in units of $\frac{\#}{m^3 s}$. User can pass choice of elastic, excitation, or ionization |

Table 3.2 Continued

| **AuxKernel Name** | **Expression** | **Description** |
|---|---|---|
| ElectronTemperature | $\frac{2}{3}e^{E_n - N_e}$ | The electron temperature |
| Position | $xl_c$ | Produces an elemental auxiliary variable useful for plotting against other elemental auxiliary variables. Mesh points automatically output by Zapdos only work for plotting nodal variables. Since almost all auxiliary variables are elemental, this AuxKernel is very important. |
| Density | $e^{N_k}N_A$ | Returns physical densities in units of $\frac{\#}{m^3}$ |
| Efield | $-\nabla V/l_c$ | Returns the x-component of the electric field (only relevant component for 1-D simulations) |
| Current | $\text{sgn}(q_k)eN_A \cdot (\text{sgn}(q)\mu_k \cdot -\nabla V e^{N_k}/l_c - D_k e^{N_k}\nabla N_k/l_c)$ | Returns the electric current associated with the flux of species $k$ |
| EFieldAdvAux | $\text{sgn}(q_k)\mu_k e^{N_k} \cdot -\nabla V N_A/l_c$ | Returns the electric field driven advective flux of species $k$ |

Table 3.2 Continued

| **AuxKernel Name** | **Expression** | **Description** |
|---|---|---|
| DiffusiveFlux | $-D_k e^{N_k} \nabla N_k N_A / l_c$ | Returns the diffusive flux of species $k$ |

### 3.1.4 Zapdos Interface Kernels

Critical to fully coupling the plasma and liquid phase simulations are the conditions at the interface. Initially, it was not possible to create interfacial conditions in Zapdos because the capability did not exist in the MOOSE framework. However, as described in section 3.2, we were able to add the capability to the framework, and thus it is now possible to create interfacial conditions in Zapdos. A couple important ones that are used in mean_en.i (see appendix A.1) are described in table 3.3.

Table 3.3 Important InterfaceKernels in Zapdos

| **InterfaceKernel Name** | **Expression** | **Description** |
|---|---|---|
| InterfaceAdvection | $-\psi_{i,el}\mu_{k,n}\,\mathrm{sgn}(q_k)e^{N_{k,n}}\nabla V_n \cdot \vec{n}/(l_{c,n}l_{c,el})$ | Used to include the electric field driven advective flux of species $k$ into or out of a neighboring subdomain. The subscript $el$ denotes the subdomain to which the InterfaceAdvection residual is being added. The subscript $n$ denotes the neighboring subdomain. Currently this interface kernel is specific to electrons because the transport coefficients are assumed to be a function of the mean electron energy. A generic interface kernel with constant transport coefficients will have a much simpler Jacobian |
| InterfaceLogDiffusionElectrons | $-\psi_{i,el}D_{k,n}e^{N_{k,n}}\nabla N_{k,n} \cdot \vec{n}/(l_{c,n}l_{c,el})$ | Used to include the diffusive flux of species $k$ into or out of a neighboring subdomain. Also currently specific to electrons. |

### 3.1.5 Zapdos Boundary Conditions

Zapdos boundary conditions at the cathode are based on the work in [110] and [30]. For ions, electrons, and the electron energy, the most commonly used conditions are respectively (in strong form and without scaling factors):

$$\vec{\Gamma_i} \cdot \vec{n} = \frac{1-r_i}{1+r_i}\left(\left(2a_i - 1\right)\mu_i \vec{E} \cdot \vec{n} n_i + \frac{1}{2} v_{th,i} n_i\right) \tag{3.19}$$

$$\vec{\Gamma_e} \cdot \vec{n} = \frac{1-r_{dens}}{1+r_{dens}}\left(-\left(2a_e - 1\right)\mu_e \vec{E} \cdot \vec{n}\left(n_e - n_\gamma\right) + \frac{1}{2} v_{th,e}\left(n_e - n_\gamma\right)\right) - \left(1 - a_e\right)\gamma_p \vec{\Gamma_p} \cdot \vec{n} \tag{3.20}$$

$$\vec{\Gamma_\epsilon} \cdot \vec{n} = \frac{1-r_{en}}{1+r_{en}}\left(-\left(2a_e - 1\right)\frac{5}{3}\mu_e \vec{E} \cdot \vec{n}\left(n_e \epsilon - n_\gamma \epsilon_\gamma\right) + \frac{5}{6} v_{th,e}\left(n_e \epsilon - n_\gamma \epsilon_\gamma\right)\right) - \frac{5}{3}\epsilon_\gamma \left(1 - a_e\right)\gamma_p \vec{\Gamma_p} \cdot \vec{n} \tag{3.21}$$

where $r_i$, $r_{dens}$, $r_{en}$ are the boundary reflection coefficients for ions, electrons, and electron energy respectively (more discussion on $r_{en}$ shortly), $\gamma_p$ is the secondary electron emission coefficient, $\epsilon_\gamma$ is the energy of the secondary electrons, $\vec{n}$ is the outward facing normal vector, and:

$$a_k = \begin{cases} 1, & sgn_k \mu_k \vec{E} \cdot \vec{n} > 0 \\ 0, & sgn_k \mu_k \vec{E} \cdot \vec{n} \leq 0 \end{cases} \tag{3.22}$$

$$v_{th,k} = \sqrt{\frac{8T_k}{\pi m_k}} \tag{3.23}$$

$$n_\gamma = \left(1 - a_e\right)\frac{\gamma_p \vec{\Gamma_p} \cdot \vec{n}}{\mu_e \vec{E} \cdot \vec{n}} \tag{3.24}$$

where $v_{th,k}$ is the thermal velocity of species $k$ and $n_\gamma$ is the density of secondary electrons. A thermodynamic interfacial condition is also available:

$$H n_{e,g} = n_{e,l} \tag{3.25}$$

For ions and electrons in the liquid phase, depending on the polarity of the discharge, a simple outflow BC is used at the counter electrode at the bottom of the liquid. Its strong form is:

$$\vec{\Gamma_k} \cdot \vec{n} = -a_k \mu_k \operatorname{sgn}(q_k) e^{N_k} \nabla V \cdot \vec{n} \tag{3.26}$$

For potential conditions, grounding is done using a DirichletBC class inherited from MOOSE. The other boundary condition incorporates an external ballast resistor:

$$V_{source} + V_{cathode} = \left(e\vec{\Gamma_i} - e\vec{\Gamma_e}\right) AR \tag{3.27}$$

A summary of the important boundary condition classes that Zapdos defines are summarized in table 3.4.

Table 3.4 Important BoundaryConditions defined by Zapdos

| **BoundaryCondition Name** | **Expression (strong form and without scaling factors)** | **Description** |
|---|---|---|
| HagelaarIonAdvectionBC | First parenthetical term in eq. (3.19) | Kinetic advective ion boundary condition |
| HagelaarIonDiffusionBC | Second parenthetical term in eq. (3.19) | Kinetic diffusive ion boundary condition |
| HagelaarElectronBC | eq. (3.20) | Kinetic electron boundary condition |
| HagelaarEnergyBC | eq. (3.21) | Kinetic electron energy boundary condition |
| DCIonBC | eq. (3.26) | Electric field driven outflow boundary condition |
| NeumannCircuitVoltageMoles_-KV | eq. (3.27) | Circuit boundary condition for potential |
| MatchedValueLogBC | eq. (3.25) | Henry's Law like thermodynamic boundary condition for specifying a specie concentration ratio at the gas-liquid interface |

### 3.1.6 Zapdos Materials

Transport, rate, and other properties are defined in class files in the materials directory of Zapdos. Gas properties (whether argon, air, etc.) are defined in the class file Gas.C. Aqueous solute properties are defined in the class file Water.C. In the gas phase, electron transport and Townsend rate coeffiecients are functions of the electron mean energy (or alternatively they can be functions of the local electric field). As shown in listing 3.2, this data is read in from a whitespace-delimited look-up table from a text input file. The look-up table data is parsed into interpolation objects. There are several interpolation types that MOOSE application developers can choose from; we have chosen to use a spline interpolator. Because a spline interpolator provides $C^2$ continuity, there are no derivative jumps with respect to the mean energy. This in turn makes for continuous Jacobian functions, leading to markedly improved convergence over a linear interpolator for example.

```
// Define path to look-up table
std::string tdPath = "/src/materials/td_argon_mean_en.txt";
std::string path = zapDir + tdPath;
const char *charPath = path.c_str();

// Create input file stream: myfile
std::ifstream myfile (charPath);
Real value;

if (myfile.is_open())
{
  // As long we haven't reached the end of file, read entries from the look-up
  table into respective data arrays
  while ( myfile >> value )
  {
    // Get mean energy values that Townsend and transport coefficients are a
  function of
    actual_mean_energy.push_back(value);
    myfile >> value;
    // Townsend ionization coefficient
    alpha.push_back(value);
    myfile >> value;
    // Townsend excitation coefficient
    alphaEx.push_back(value);
    myfile >> value;
    // Townsend elastic collision coefficient
    alphaEl.push_back(value);
```

```
      myfile >> value;
      // Electron mobility
      mu.push_back(value);
      myfile >> value;
      // Electron diffusivity
      diff.push_back(value);
    }
    myfile.close();
  }

  else std::cerr << "Unable to open file" << std::endl;

  // Create interpolation functions for Townsend and transport coefficients that
    depend on the mean energy
  _alpha_interpolation.setData(actual_mean_energy, alpha);
  _alphaEx_interpolation.setData(actual_mean_energy, alphaEx);
  _alphaEl_interpolation.setData(actual_mean_energy, alphaEl);
  _mu_interpolation.setData(actual_mean_energy, mu);
  _diff_interpolation.setData(actual_mean_energy, diff);
```

Listing 3.2 Code for reading in electron transport and Townsend coefficient data from a lookup table in a text file

Listing 3.3 shows definition of both constant material properties and solution variable-dependent properties. In this particular codeblock the code tests whether the user wants the electron mobility and diffusivity to be functions of the mean energy. If so, the interpolator **sample** method is called to retrieve the property at the corresponding value for the mean energy (recall that $\epsilon$ is a function of $E_n$ and $N_e$; see eq. (3.15)). In addition the interpolator's **sampleDerivative** method is also called; its returned value is used in the Jacobian methods of any kernels or boundary conditions that rely on the corresponding material property. If the user does not want to interpolate the transport coefficients, then they are set to some constant sane values. The code block also shows how the properties are scaled depending on the choice of potential units.

```
// Check whether user wants to interpolate transport coefficients as a function
   of mean energy, or just use constants
if (_interp_trans_coeffs) {
  // Get value for mobility
  _muem[_qp] = _mu_interpolation.sample(std::exp(_mean_en[_qp]-_em[_qp])) *
  _voltage_scaling;
  // Get derivative of mobility with respect to the mean energy. Used in
  Jacobian computations
```

```
    _d_muem_d_actual_mean_en[_qp] = _mu_interpolation.sampleDerivative(std::exp(
    _mean_en[_qp]-_em[_qp])) * _voltage_scaling;
    // Get value for diffusivity
    _diffem[_qp] = _diff_interpolation.sample(std::exp(_mean_en[_qp]-_em[_qp]));
    // Get derivative of diffusivity with respect to the mean energy. Used in
    Jacobian computations
    _d_diffem_d_actual_mean_en[_qp] = _diff_interpolation.sampleDerivative(std::
    exp(_mean_en[_qp]-_em[_qp]));
  }
  else {
    // From bolos at atmospheric pressure and an EField of 2e5 V/m
    _muem[_qp] = 0.0352103411399 * _voltage_scaling; // units of m^2/(kV*s) if
    _voltage_scaling = 1000
    // No functional dependence on mean energy if transport coefficients are
    constant
    _d_muem_d_actual_mean_en[_qp] = 0.0;
    _diffem[_qp] = 0.297951680159;
    _d_diffem_d_actual_mean_en[_qp] = 0.0;
  }
```

Listing 3.3 Material property definition

### 3.1.7 Meshing for Zapdos

Meshes required for Zapdos input files are generated using the program Gmsh. [128] An example Gmsh input file is shown in listing 3.4. The scaling used in the mesh input file is the inverse of the scaling used in the Zapdos input file, e.g. $dom0Mult = 1/dom0Scale$. For typical plasma simulations, the characteristic length of the mesh is much finer in the boundary regions than in the plasma bulk. For the input file below, which is representative of the meshes used for section 2.2, the characteristic length of the mesh is 2 nm at the cathode and 1 nm at the plasma-liquid interface with the mesh characteristic length peaking at 50 $\mu$m in the center of the discharge. In the liquid phase, the characteristic length in the bulk is 10 nm.

```
/* In this example we have chosen to scale the mesh to improve Jacobian
    conditioning */
dom0Mult = 1e3;
dom1Mult = 1e7;

/* We would comment the above two lines and uncomment the two lines below if we
    did not want to scale the mesh */
```

```
// dom0Mult = 1;
// dom1Mult = 1;

// Specify a 2 nm characteristic length at the left edge of the plasma
Point(1) = {0, 0, 0, 2e-9 * dom0Mult};

// 50 micron characteristic length in plasma center
Point(3) = {.5e-3 * dom0Mult, 0, 0, 50e-6 * dom0Mult};
Line(2) = {1,3};

// 1 nm characteristic at gas-liquid interface
Point(8) = {1e-3 * dom0Mult, 0, 0, 1e-9 * dom0Mult};
Line(7) = {3,8};

// 10 nm characteristics at liquid domain center and right edge
Point(9) = {1e-3 * dom0Mult + 50e-9 * dom1Mult, 0, 0, 10e-9 * dom1Mult};
Line(8) = {8,9};
Point(10) = {1e-3 * dom0Mult + 100e-9 * dom1Mult, 0, 0, 10e-9 * dom1Mult};
Line(9) = {9,10};

// Create physical mesh objects that will be recognized by Zapdos
Physical Line(0) = {2,7};
Physical Line(1) = {8,9};
```

Listing 3.4 Gmsh input file used to create plasma and liquid domains for simulations in section 2.2

### 3.1.8 Postprocessing Zapdos results

Zapdos by default outputs results to an exodus file. Our preferred tool for processing these results is the open-source application Paraview. [129] Paraview offers both a GUI as well as python modules for processing data in a variety of formats, including exodus. Using the python interface, we can script creation of Paraview readers and write data for specific time steps as well as perform many other functions. In the case where we are simulating to steady-state a DC discharge from some arbitrary initial state, solution data from the last time point are mostly what we care about. Using Paraview's CSV writer from within python, we write the data to a csv file, from which we can then load the data into numpy data arrays. These steps are achieved in a single python method that we developed; it is shown in listing A.2. Once nodal and elemental variables have been loaded into numpy arrays, python's matplotlib package can be used to render publication-quality figures. This is automated using generic figure scripts shown in listings A.3 and A.4.

## 3.2 Modifying the MOOSE Framework

Essential to the simulation of plasma-liquid systems is the ability to couple the physics of the two domains across their interface. Somewhat surprisingly for a framework used for very mature multi-physics application codes, when first starting to investigate simulation of plasma-liquids, MOOSE lacked a straightforward way to interface physics across domains. E.g. one could not set the flux of a specie A on the domain 0 side of an interface equal to the flux of a specie B on the domain 1 side of the interface. If one wanted to ensure continuity of a species flux across an interface, he had no choice but to use the same variable on both domains, which has the unfortunate side-effect for a Continuous Galerkin formulation of also requiring continuity of the species concentration across the interface as well. This condition is of course physically unrealistic in almost all cases; as two examples, hydrophilic $H_2O_2$ has six orders of magnitude higher concentration on the liquid side of an interface and hydrophobic NO has two orders of magnitude higher concentration on the gas side of the interface. Thus in order to solve the plasma-liquid equations in a code built on top of MOOSE, the MOOSE framework itself had to be altered to allow creation of residual and Jacobian statements for interfacial conditions like continuity of flux.

MOOSE has several “systems”; before modification by the author, the systems directly responsible for computing residual contributions from pieces of user governing equations were Kernels (including a few closely related variants like Nodal Kernels and Dirac Kernels), DGKernels (for discontinuous Galerkin computations), constraints, and boundary conditions. The MOOSE constraints system could potentially have been used in a somewhat inelegant way to impose interfacial conditions; however, this would have required at a minimum purchase of an expensive and proprietary commercial meshing package. Instead the author chose to code a brand new MOOSE system called “Interface Kernels.” At the time of writing, the implementation of Interface Kernels has required modification or creation of 40 framework files, encompassing the addition or modification of 1,400 lines of code.

As described in section 3.1, at the core of MOOSE and any application like Zapdos built on top of it is the computation of residual and Jacobian statements (see eq. (3.8)). Adding a new system requires supplying all the code necessary to route the MOOSE framework core residual and Jacobian compute threads to the user provided residual and Jacobian statements. In order to add the new InterfaceKernel system, intelligent decisions about the user interface had to be made. InterfaceKernels resemble most strongly a cross between DGKernels and Integrated Boundary Conditions (IntegratedBCs). DGKernels operate on internal sides of subdomains, providing the user access to solution variable values and gradients at surface interfaces between elements; IntegratedBC's require the user to provide a boundary (or boundaries) to restrict the condition to. InterfaceKernels make use of both of these features: the InterfaceKernel inherits from the DGKernel class, providing InterfaceKernel objects with the ability to access variables on either side of element interfaces, and the BoundaryRestrictable class, allowing the user to specify the internal surfaces on the mesh where the interfacial conditions will live.

After deciding on the user interface for the InterfaceKernel class, the author had to program the MOOSE core residual and Jacobian compute threads to call the respective InterfaceKernel member functions. The partial stack trace shown in listing 3.5 shows the architecture for how InterfaceKernel residuals get called. Full stack traces trace function calls from the main program to the function under investigation; they are very useful for determining how large programs like MOOSE are structured. In listing 3.5 we show the parts of the stack trace that are important for the newly implemented interface kernel system. We describe each function in the stack trace with a comment; at the end of each comment we indicate whether the function described is newly implemented by us or whether it already existed in the framework. Shown as item # 4 in listing 3.5, the ThreadedElementLoopBase::operator method is used to loop over elements; it can call both residual and Jacobian threads. Listing 3.6 shows the calls to different geometric objects. The call to onElement computes kernel methods that exist in element volumes; onBoundary calls integrated boundary condition methods that exist on the external sides of the domain; onInternalSide calls discontinuous Galerkin kernels that exist on element sides internal to the domain; finally, the last object call, onInterface, is the call implemented by the author that calls methods for element sides that lie along an interface between subdomains. Both the ComputeResidualThread and ComputeJacobianThread classes are children of the ThreadedElementLoopBase class and must implement the onInterface method that is called by their parent.

```
    /* Call residual functions specific to application physics. In this case we
    are calling InterfaceAdvection which takes the advective flow of electrons
    out of the gas domain and creates an advective flow of electrons into the
    liquid domain. (New capability) */
#0  InterfaceAdvection::computeQpResidual (this=0xf72bf0, type=Moose::Element) at
     /home/lindsayad/projects/zapdos/src/interfacekernels/InterfaceAdvection.C:56

    /* This function tests to see whether we are on the master or slave side of
    the interface. If we are on the master side we use the test functions
    associated with the master side and visa versa for the slave side. This
    function is also responsible for adding the residual to the correct residual
    block, e.g. if we are on the master side, the residual should be added to the
     residual block for the master variable. This function calls child classes
    computeQpResidual functions like that of InterfaceAdvection above (New
    capability) */
#1  0x00007ffff69beae9 in InterfaceKernel::computeElemNeighResidual (this=0
    xf72bf0, type=Moose::Element) at /home/lindsayad/moose/framework/src/
    interfacekernels/InterfaceKernel.C:57

    /* A simple function that calls InterfaceKernel::computeElemNeighResidual
    twice. The first time it tells the InterfaceKernel class to compute the
    master side residual. The second time it tells InterfaceKernel to compute the
     slave side residual (Pre-existing capability) */
#2  0x00007ffff64463b6 in DGKernel::computeResidual (this=0xf72bf0) at /home/
    lindsayad/moose/framework/src/dgkernels/DGKernel.C:134

    /* Function that sweeps through all existing InterfaceKernel child objects
    and calls their compute residual threads; e.g. example thread is
    ComputeResidualThread::onInterface -> DGKernel::computeResidual ->
    InterfaceKernel::computeElemNeighResidual -> InterfaceAdvection::
    computeQpResidual (New capability) */
#3  0x00007ffff67dbd69 in ComputeResidualThread::onInterface (this=0x7fffffffc810
    , elem=0xd3c570, side=0, bnd_id=2) at /home/lindsayad/moose/framework/src/
    base/ComputeResidualThread.C:162

    /* Function that iterates through both residual threads like the thread
    described immediately above and Jacobian threads like ComputeJacobianThread::
    onInterface -> ComputeFullJacobianThread::computeInternalInterFaceJacobian ->
     InterfaceKernel::computeOffDiagJacobian -> InterfaceKernel::
    computeOffDiagElemNeighJacobian -> InterfaceAdvection::
    computeQpOffDiagJacobian (Some new capability) */
#4  0x00007ffff64b8903 in ThreadedElementLoopBase<libMesh::StoredRange<libMesh::
    MeshBase::const_element_iterator, libMesh::Elem const*> >::operator() (this=0
    x7fffffffc810, range=..., bypass_threading=false) at /home/lindsayad/moose/
    framework/include/base/ThreadedElementLoopBase.h:180
```

Listing 3.5 Stack trace showing the architecture for how InterfaceKernel residuals get called

```
      // Call residual and Jacobian functions of objects associated with
    volumetric elements. These are Kernel objects
    onElement(elem);

    for (unsigned int side=0; side<elem->n_sides(); side++)
    {
      // Get IDs of mesh boundaries where boundary conditions are defined
      std::vector<BoundaryID> boundary_ids = _mesh.getBoundaryIDs(elem, side);

      if (boundary_ids.size() > 0)
        // Loop over boundary IDs
        for (std::vector<BoundaryID>::iterator it = boundary_ids.begin(); it !=
  boundary_ids.end(); ++it)
          // Call residual and Jacobian functions associated with boundaries.
 These are IntegratedBC objects
          onBoundary(elem, side, *it);

      if (elem->neighbor(side) != NULL)
      {
        // Call residual and Jacobian functions associated with mesh internal
 sides. These are DGKernel objects
        onInternalSide(elem, side);
        if (boundary_ids.size() > 0)
          for (std::vector<BoundaryID>::iterator it = boundary_ids.begin(); it
 != boundary_ids.end(); ++it)
            // Call residual and Jacobian functions associated with interfaces
 between subdomains. These are the newly implemented InterfaceKernel objects
            onInterface(elem, side, *it);
      }
    } // sides
```

Listing 3.6 Snapshot of different geometric object calls in ThreadedElementLoopBase::operator

Listing 3.7 shows the implementation of the onInterface method in the ComputeResidualThread class. The method takes as arguments the current element, one of the element's sides, and a boundary ID (bnd_id). The method first checks whether any _interface_kernels exist and are active on the boundary specified by bnd_id. The initialization of _interface_kernels will be discussed later. The element's neighbor is accessed using its neighbor method. Currently, the interface kernel system does not support mesh adaptivity; this is checked by comparing neighbor->level() and elem->level() (the level method returns the level of element refinement). After checking whether the neighboring element is active (relevant for transient simulations), a reinitialization of the element face and neighboring face materials is performed. In the most important lines of the method, all of the _interface_kernels active on bnd_id are iterated over, and their individual computeResidual methods are called. The logic for the onInterface method implemented in the ComputeJacobianThread class is very similar to that of ComputeResidualThread.

```cpp
void
ComputeResidualThread::onInterface(const Elem *elem, unsigned int side,
    BoundaryID bnd_id)
{
  // Check whether any interface kernels are active on the provided boundary ID
  if (_interface_kernels.hasActiveBoundaryObjects(bnd_id, _tid))
  {

    // Pointer to the neighbor we are currently working on.
    const Elem * neighbor = elem->neighbor(side);

    if (!(neighbor->level() == elem->level()))
      mooseError("Sorry, interface kernels do not work with mesh adaptivity");

    // Check whether neighboring element is active. E.g. some transient
    simulations may not simulate all of the subdomains for all time steps
    if (neighbor->active())
    {
      _fe_problem.reinitNeighbor(elem, side, _tid);

      // Make sure material properties are up-to-date
      _fe_problem.reinitMaterialsFace(elem->subdomain_id(), _tid);
      _fe_problem.reinitMaterialsNeighbor(neighbor->subdomain_id(), _tid);

      const std::vector<MooseSharedPointer<InterfaceKernel> > & int_ks =
    _interface_kernels.getActiveBoundaryObjects(bnd_id, _tid);
      // Iterate over all interface kernels active on the provided boundary ID
    and compute the corresponding residuals
      for (std::vector<MooseSharedPointer<InterfaceKernel> >::const_iterator it =
     int_ks.begin(); it != int_ks.end(); ++it)
        (*it)->computeResidual();

      _fe_problem.swapBackMaterialsFace(_tid);
      _fe_problem.swapBackMaterialsNeighbor(_tid);

      {
        Threads::spin_mutex::scoped_lock lock(Threads::spin_mtx);
        _fe_problem.addResidualNeighbor(_tid);
      }
    }
  }
}
```

Listing 3.7 ComputeResidualThread::onInterface method. The logic is much the same for the ComputeJacobianThread::onInterface method.

The computeResidual method called from ComputeResidualThread :: onInterface is inherited from the DGKernel class. It calls in succession InterfaceKernel :: computeElemNeighResidual(Moose::Element) and InterfacKernel :: computeElemNeighResidual(Moose::Neighbor). The InterfaceKernel :: computeElemNeighResidual method is shown below in listing 3.8. Depending on whether Moose::Element or Moose::Neighbor is passed as the method argument, the space of test functions is taken from the focused element or the neighboring element respectively. The correct residual block, either for _var in the focused element or _neighbor_var in the neighboring element, is similarly determined from the method argument. At the end of the method, we loop over all the quadrature points in the element and add _JxW[_qp] * _coord[_qp] * computeQpResidual(type) to the i$^{th}$ residual. Here _JxW represents the quadrature weight, _coord is a scaling factor for converting from Cartesian to other coordinate systems (e.g. cylindrical or spherical), and computeQpResidual is the residual computed at the quadrature points by the current InterfaceKernel child class.

```
void
InterfaceKernel::computeElemNeighResidual(Moose::DGResidualType type)
{
  bool is_elem;
  // If type == Moose::Element, we are on the master side of the interface, which
     through MOOSE convention we also call the ``element'' side
  if (type == Moose::Element)
    is_elem = true;
  else
    is_elem = false;

  // Get the test functions matching the side of the interface we are on
  const VariableTestValue & test_space = is_elem ? _test : _test_neighbor;

  // Make sure residual is going towards the correct variable
  DenseVector<Number> & re = is_elem ? _assembly.residualBlock(_var.number()) :
                                          _assembly.residualBlockNeighbor(
    _neighbor_var.number());

  // Loop over quadrature points and test functions and add residual
    contributions
  for (_qp = 0; _qp < _qrule->n_points(); _qp++)
    for (_i = 0; _i < test_space.size(); _i++)
      re(_i) += _JxW[_qp] * _coord[_qp] * computeQpResidual(type);

}
```

Listing 3.8 The InterfaceKernel :: computeElemNeighResidual method responsible for calling compueQpResidual methods implemented in the various children of the InterfaceKernel class

An example of a computeQpResidual method implemented in a child class of InterfaceKernel is taken from the InterfaceAdvection class defined in Zapdos. It is shown in listing 3.9. The InterfaceAdvection class ensures that all the species being advected from one subdomain flow into the neighboring subdomain. It represents an interfacial condition acting only on the variable living on the focused element; as can be seen from the switch and case logic, no residual contribution is given for the variable living on the neighboring element. Although not relevant for describing the InterfaceKernel system, note that _r_units is a data member controlled by the user enabling mesh scaling and improved Jacobian conditioning. As noted previously, species concentration variables are in a logarithmic form such that std::exp(_neighbor_value) actually represents the physical concentration of the neighboring specie; _mu_neighbor is the mobility of the neighboring species, _sgn_neighbor is the charge sign of the neighboring speices, and _grad_potential_neighbor is the gradient of the potential on the neighboring side of the interface.

```
Real
InterfaceAdvection::computeQpResidual(Moose::DGResidualType type)
{
  Real r = 0;

  switch (type)
  {
  case Moose::Element:
    // Add the flux of electrons from the neighboring subdomain to the balance
    equation for electrons in the current subdomain
    r = _mu_neighbor[_qp] * _sgn_neighbor[_qp] * -_grad_potential_neighbor[_qp] *
     _r_neighbor_units * std::exp(_neighbor_value[_qp]) * _normals[_qp] * _test[
    _i][_qp] * _r_units;
    break;

  case Moose::Neighbor:
    // This condition is only imposed on the master side of the interface, thus
    we do not add any residual contribution to the neighboring side
    r = 0.;
    break;
  }

  return r;
}
```

Listing 3.9 InterfaceAdvection::computeQpResidual method

Interface kernels are read from their own input block in a MOOSE application's input file. An example is shown in listing 3.10. The act method in the AddInterfaceKernelAction class calls FEProblem :: addInterfacerKernel which in turn calls NonlinearSystem :: addInterfaceKernel. NonlinearSystem :: addInterfaceKernel adds the interface kernels from the input file to the protected _interface_kernels data member which are then accessible to the ComputeResidualThread and ComputeJacobianThread classes through the public accessor method, getInterfaceKernelWarehouse. The details of this initialization process can be found in the source code at [35]. An important thing to note about interface kernels is that they are uniquely assigned to elements on one side of the interface. Without unique assignment, there could be no organized residual definitions like that shown in listing 3.9. Unique assignment is achieved by using libMesh's sideset objects. The block used to create the sideset 'master1_interface' that is then used to uniquely define the interface kernel of listing 3.10 is shown in listing 3.11. Using the built-in SideSetsBetweenSubdomains class, the new sideset is constructed on the block 1 side of the interface.

```
[InterfaceKernels]
  // This condition adds the advective flux of electrons coming from the gas
  phase to the balance equation of electrons in the liquid phase
  [./em_advection]
    type = InterfaceAdvection
    // ``mean_en'' is the gas phase electron energy. There is no electron energy
    variable in the liquid phase
    mean_en_neighbor = mean_en
    // The ``potential'' variable spans both gas and liquid phases since it is
    continuous at the interface
    potential_neighbor = potential
    // ``em'' is the gas phase electron density. It is the slave variable
    corresponding to the ``emliq'' master variable
    neighbor_var = em
    // ``emliq'' is the liquid phase electron density and is the master variable
    for this InterfaceAdvection object
    variable = emliq
    // This interfacial condition is imposed on the liquid side of the interface.
     ``1'' denotes the liquid subdomain. ``0'' denotes the plasma subdomain
    boundary = master1_interface
    // Following two lines specify the mesh scaling for both liquid and plasma
    subdomains respectively
    position_units = ${dom1Scale}
    neighbor_position_units = ${dom0Scale}
  [../]
[]
```

Listing 3.10 Example of input block for an interface kernel (InterfaceAdvection in this case)

```
[MeshModifiers]
  [./interface_again]
    type = SideSetsBetweenSubdomains
    // ``1'' denotes the liquid subdomain.
    master_block = '1'
    // ``0'' denotes the plasma subdomain.
    paired_block = '0'
    // This new boundary will be a sideset tied to the liquid subdomain
    new_boundary = 'master1_interface'
  [../]
[]
```

Listing 3.11 Example of how to create a sideset in this case 'master1_interface' that can then be used in definition of an interface kernel

Addition of the interface kernel system to the MOOSE framework enables the interfacing of plasma and liquid domains required to obtain the results in section 2.2. Moreover, the system should be applicable to many other scientific and engineering applications that inherit from the MOOSE framework. In the words of MOOSE founder Derek Gaston: “Thanks for all this work! To my best knowledge I think this is the first time an external contributor has added a whole new "System" in MOOSE! Definitely a landmark occasion! This is a good one too... LOTS of people will use this for many years to come (including myself!).”

CHAPTER

4

# EXPERIMENTAL OPTIMIZATION OF PLASMA-LIQUID INTERACTIONS: VHF SOURCE

Chapter 2 outlines fundamental modeling efforts aimed at describing the physical and chemical phenomena that occur in plasma-liquid systems. Chapter 3 outlines the tool we created in order to better conduct our modeling efforts. To date modeling has been used to gain a better qualitative understanding of transport processes in convective plasma-liquid systems (section 2.1) and to explore the effects of key interfacial parameters on plasma properties (section 2.2). Model geometries have been based on relatively simple experimental set-ups (point-to-plane corona discharge for section 2.1) or simplified to one dimension as in section 2.2. However, the groundwork has been laid to model more complex plasma-liquid geometries and exotic electromagnetic fields. Such models will be used to describe the physiochemical properties observed in the complex experimental configurations described in this chapter. This chapter outlines plasma-liquid geometries that exhibit increasing degrees of plasma-liquid contact. In section 4.2 we describe our base experimental configuration: a very high frequency (VHF) atmospheric plasma source that is pointed down into a reservoir of water such that the end of the plasma column is in direct contact with the water surface. In section 4.3 we discuss spraying water droplets directly through the plasma. After discussing the typical electrodes used on the VHF source in section 4.4, we introduce in section 4.5 a completely novel design where the VHF source is pointed upward and water is pumped through the center of the inner conductor to form a water layer on top of the powered electrode. Finally, in section 4.6 we show experimental measurements of aqueous chemistry generated by our plasma-liquid systems. We hope to reproduce our experimental measurements in section 4.6 using a future combination and extension of the models and code described in chapters 2 and 3.

## 4.1 Description of NCSU VHF Source

A detailed description of the NCSU VHF source is given in [41]; a summary of the design is given below. The source is powered by a 3.5 kW 162 MHz generator (Advanced Energy Ovation 35162). The generator has a termination impedance of 50 Ω and is connected to the plasma source using a 50 Ω high-power coaxial cable. A directional coupler located at the output of the generator is used to track forward and reflected power. Source impedance matching is achieved using tuned stub matching. At the connection of the RF power cable and the plasma source, the RF signal is split towards load and ground terminations (see fig. 4.1), the input and terminations are joined by a coaxial transmission line formed by aluminum inner and outer conductors. The inner diameter of the source coax is 2.25 cm and the outer diameter is 5.25 cm. With air as the feed gas, the coaxial structure's characteristic impedance is 51.7 Ω. The length of the load and ground terminations are both variable, giving two degrees of freedom for matching. The last 3.8 cm of the inner conductor at the load termination is flared to a 3.5 cm diameter to assist in plasma ignition; the flared conductor piece is often hereafter referred to as the powered electrode. After ignition, a plasma column is observed in front of the powered electrode. It is speculated that the high driving frequency of the discharge creates a ballasting effect that prevents thermal arcing of the discharge. Ballasting occurs because increasing electron density actually increases the bulk plasma resistance, creating a negative feedback loop that tends to stabilize the glow discharge. [41]

## 4.2 Base Set-up for Water Treatment

The experimental set-up shown in fig. 4.1 is known as the "batch" set-up. It was the first plasma-water configuration explored by the group. Originally it was intended for degradation of perfluorinated compounds like perfluorooctanesulfonic acid (PFOS) and perfluorootanoic acid (PFOA). It turned out that the batch configuration was unable to degrade these persistent chemicals; however, in the process it was discovered that the configuration generated large amounts of $NO_x$, mostly $NO_3^-$, in the aqueous phase. Generation of nitrogen and oxygen species (RONS) in solution by plasmas is now a well-known phenomenon in the plasma-liquid community; however, at the time it was a novel discovery for our group. Recognizing that aqueous nitrogen, specifically $NO_3^-$, is a key component in fertilizer, we were motivated to begin a study in collaboration with the horticulture department of plant fertilization using plasma activated water (PAW). This study is outlined in section 5.1. Later, the batch configuration was used in exploration of dioxane degradation; this is discussed in section 5.2.2.

Depending on the application, delivered power to the plasma for the batch configuration ranges between 350 and 1000 W. Many different gases are used, including air, argon, helium, nitrogen, and carbon dioxide. Gas flow rates range from 2-5 standard cubic feet per minute. The gap distance between the powered electrode and the water surface range from a few milimeters to a couple centimeters, with larger gap distances typically used in combination with larger gas flow rates in order to avoid splashing of the electrode and extinguishing of the plasma. Water treatment volumes typically span 100-500 mL for persistent chemical studies up to several liters for PAW generation in the fertigation experiments.

## 4.3 Spray-through Design

One way thought to increase plasma-water interaction is to directly introduce water droplets into the active plasma region. We hypothesize that there are two good reasons for doing this. Firstly, it is reasoned that water droplets passing through the core of the plasma as opposed to the edge or afterglow are exposed to greater densities of electrons, ions, and reactive radical species. Secondly, by breaking the water volume into droplets, the surface-to-volume ratio is increased, increasing the rate of mass transfer of plasma species into the aqueous phase. Two different configurations are used to explore these concepts; they are outlined in the following subsections.

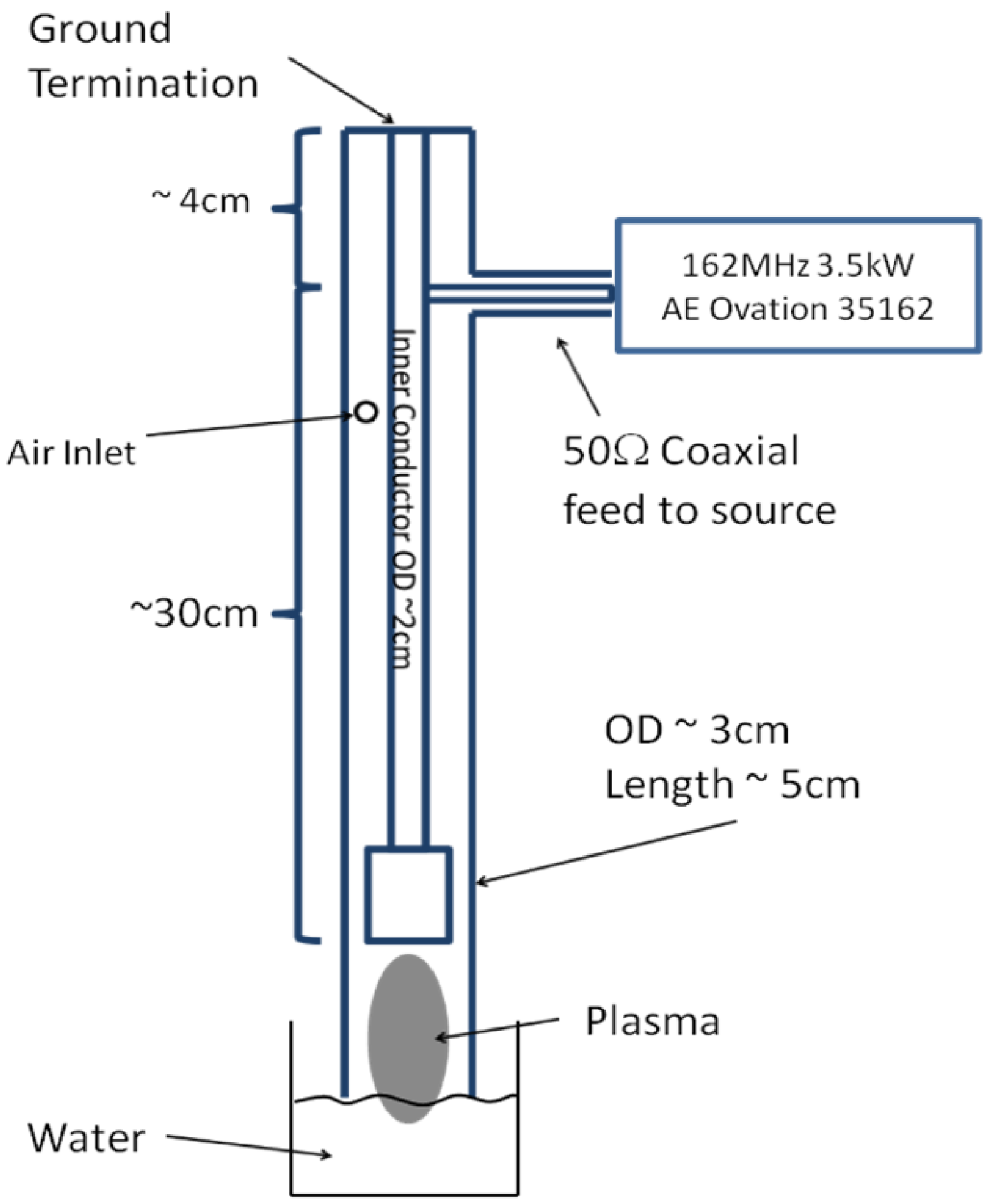


Figure 4.1 Schematic of the atmospheric plasma source and batch water treatment set-up

### 4.3.1 Spray Bottle

The easiest way to achieve a droplet configuration is to take the batch set-up (see fig. 4.1) and remove the beaker of water under the coaxial plasma source. Then after turning the plasma on, a greenhouse sprayer is used to pass droplets radially through the plasma; a beaker is used to catch the droplets after passage through the plasma. A summary of the configuration is shown in fig. 4.2.

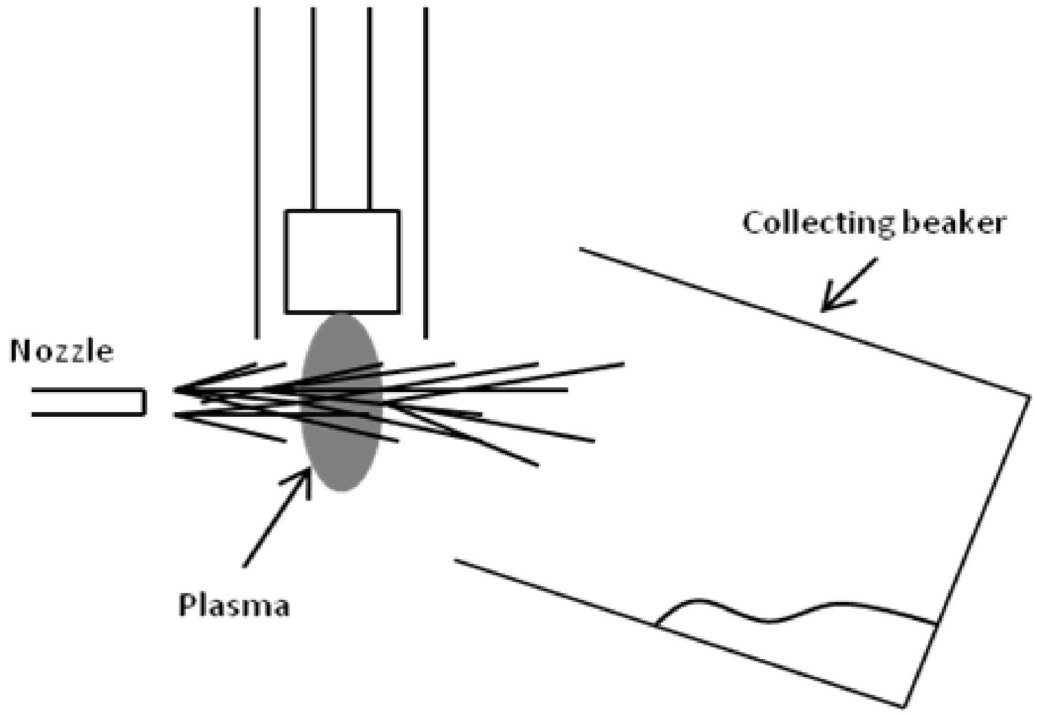


Figure 4.2 Set-up for introducing water directly into the active plasma region. A greenhouse sprayer injects water from the side of the plasma source; water is collected in a beaker on the other side

A comparison of batch and greenhouse sprayer configurations for generation of nitrate in solution per unit energy is shown in fig. 4.3. It is found that in general the greenhouse sprayer configuration performs more favorably than the batch treatment design. This is especially clear at higher powers. Moreover, the performance of the greenhouse sprayer configuration appears to improve with increasing power delivered to the plasma. However, increasing plasma power also has some negative effects. One is an increased rate of erosion of the powered electrode. A second negative consequence is increased reflected power back to the generator during plasma instabilitiesarising from the water droplets. Both of these effects decrease the lifetime of the design; decreasing the lifetime of the generator is particularly undesirable because of its cost.

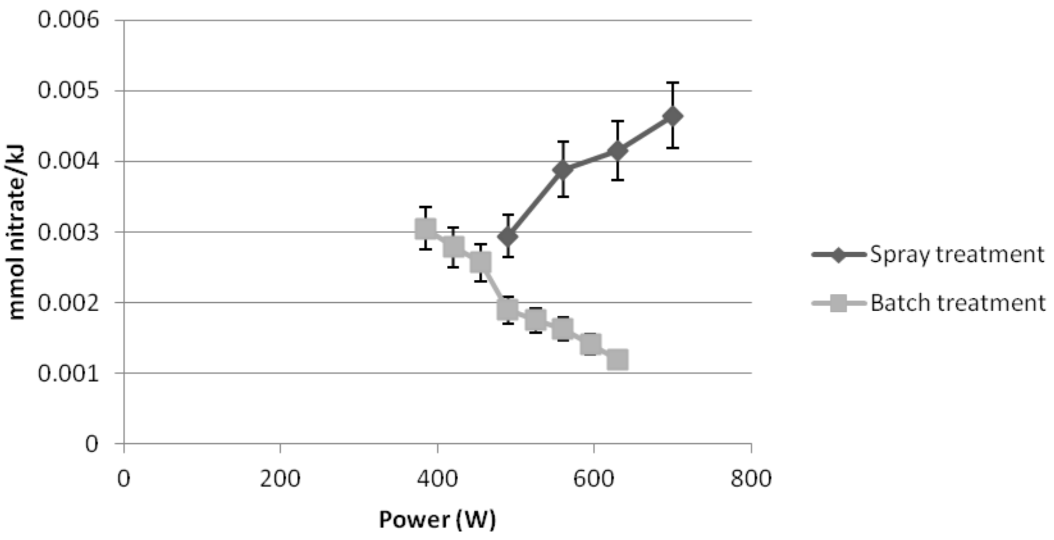


Figure 4.3 Comparison between batch and spray treatment methods using mmol of nitrate generated per kJ of electrical energy as the figure of merit. For lower powers batch treatment is more energetically efficient for nitrate generation. For higher powers, spray treatment is more efficient.

What ultimately curtails further investigation of the spray-through design is the instability of the plasma. The sprayer must be placed such that droplets do not touch the surface of the powered electrode or else the plasma is immediately extinguished. Moreoever, even if the sprayer is properly placed and the electrode is not wetted, the plasma actively trys to avoid the droplet stream. This may occur for several reasons. Firstly, a much higher electric field must be applied to a dense liquid as opposed to a gas to create or sustain a discharge. Secondly, highly oxidative species like OH and $OH_2$ originating from the liquid phase can scavenge electrons. Typically even in the most optimized sprayer set-up, the plasma extinguishes after a few tens of seconds. Compare this with the batch set-up in which water can be treated continuously for multiple hours.

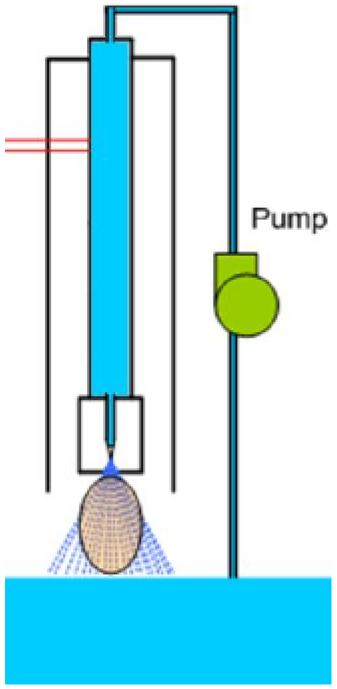


Figure 4.4 Schematic of the nozzle electrode spray-through configuration

### 4.3.2 Built-in Nozzle

Figure 4.4 shows a schematic of the nozzle electrode experimental design. In terms of plasma-liquid contact, the concept is very similar to fig. 4.2 except the droplets are introduced vertically through the VHF source's inner conductor. Unfortunately, the nozzle electrode design suffers from the same pitfall as the greenhouse sprayer design. During operation, the plasma actively avoids the water droplets, moving with the cyclonic flow of the gas feed around the outside of the droplet stream. It is speculated that the droplet stream may form a partial Faraday cage inhibiting the discharge. Additionally, electronegative species like OH and $OH_2$ originating from the liquid phase may scavenge electrons in a manner analogous to the spray-through configuration of section 4.3.1. On top of plasma instability originating from liquid interactions, the surface non-uniformity introduced by the nozzle on the electrode leads to faster surface erosion.

## 4.4 Base electrode designs

As mentioned in section 4.3.1, plasma erosion of the source's powered electrode can occur, especially at higher powers. Evidence of this erosion can be seen both with the naked eye and in the optical emission spectrum of the discharge. Figure 4.5 shows the presence of an atomic aluminum line at 395 nm and several AlO bands between 425 and 575 nm. Visually, this emission manifests itself as an intense bright blue. Figure 4.6 shows plasma color during normal operation, plasma color during aluminum damage, and the resulting appearance of the electrode after significant erosion.

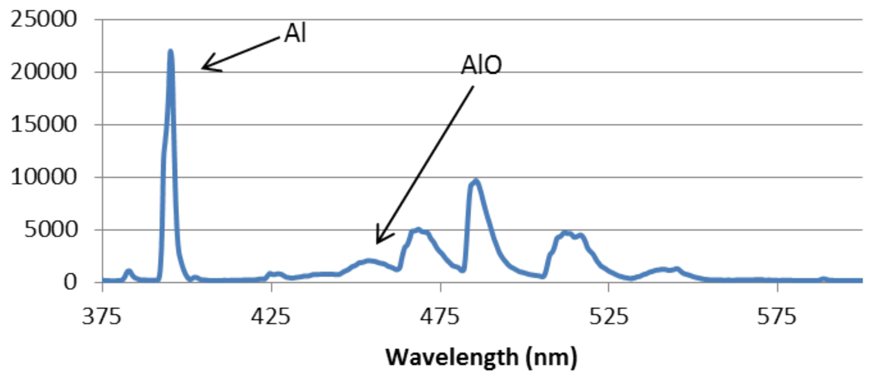
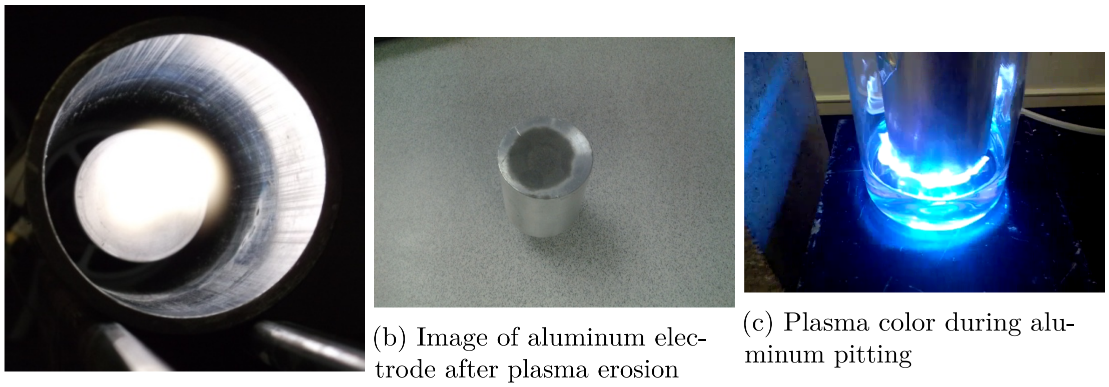


Figure 4.5 OES spectrum of plasma damaged aluminum electrode

The plasma damage to the electrode can be investigated more closely using Secondary Electron Microscopy (SEM) and Energy Dispersive X-ray Spectroscopy (EDS). Even with a 1mm zoom (fig. 4.7), the growth of a damage layer is evident. Taking an EDS measurement of the clean aluminum yields the spectrum shown in fig. 4.8. Unsurprisingly, the spectrum shows almost pure aluminum with a trace of magnesium. An EDS scan of the damaged aluminum portion, however, reveals the growth of substantial carbon and oxygen peaks (fig. 4.9). The oxidation is unsurprising considering the flow gas is often compressed air and the ambient environment is also air (also consistent with the OES spectrum (fig. 4.5)). The carbon could be coming from oils/hydrocarbons present in the compressed air feed.

In an attempt to prolong the lifetime of the powered electrode, metals other than aluminum are



(a) Normal plasma color

(b) Image of aluminum electrode after plasma erosion

(c) Plasma color during aluminum pitting

Figure 4.6 Normal vs. abnormal plasma glows

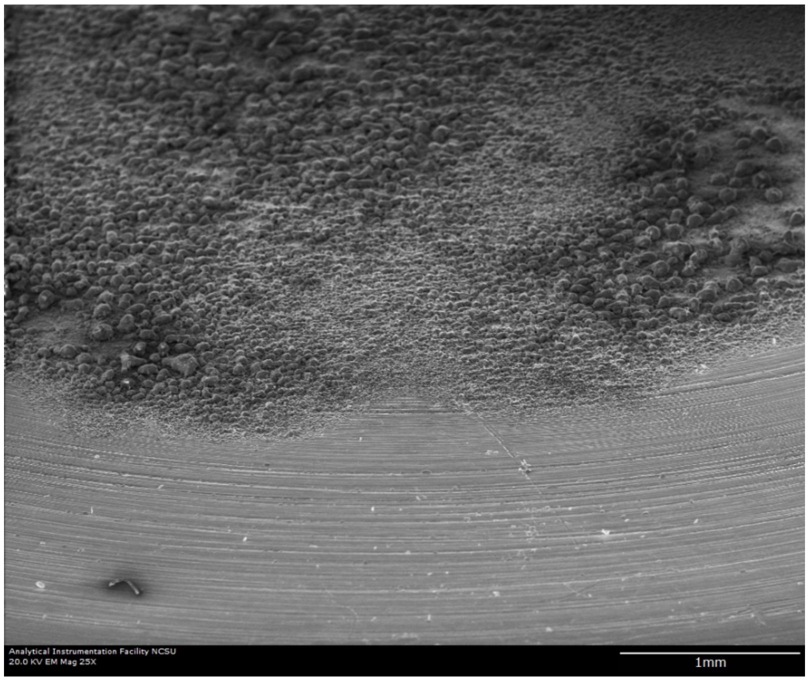
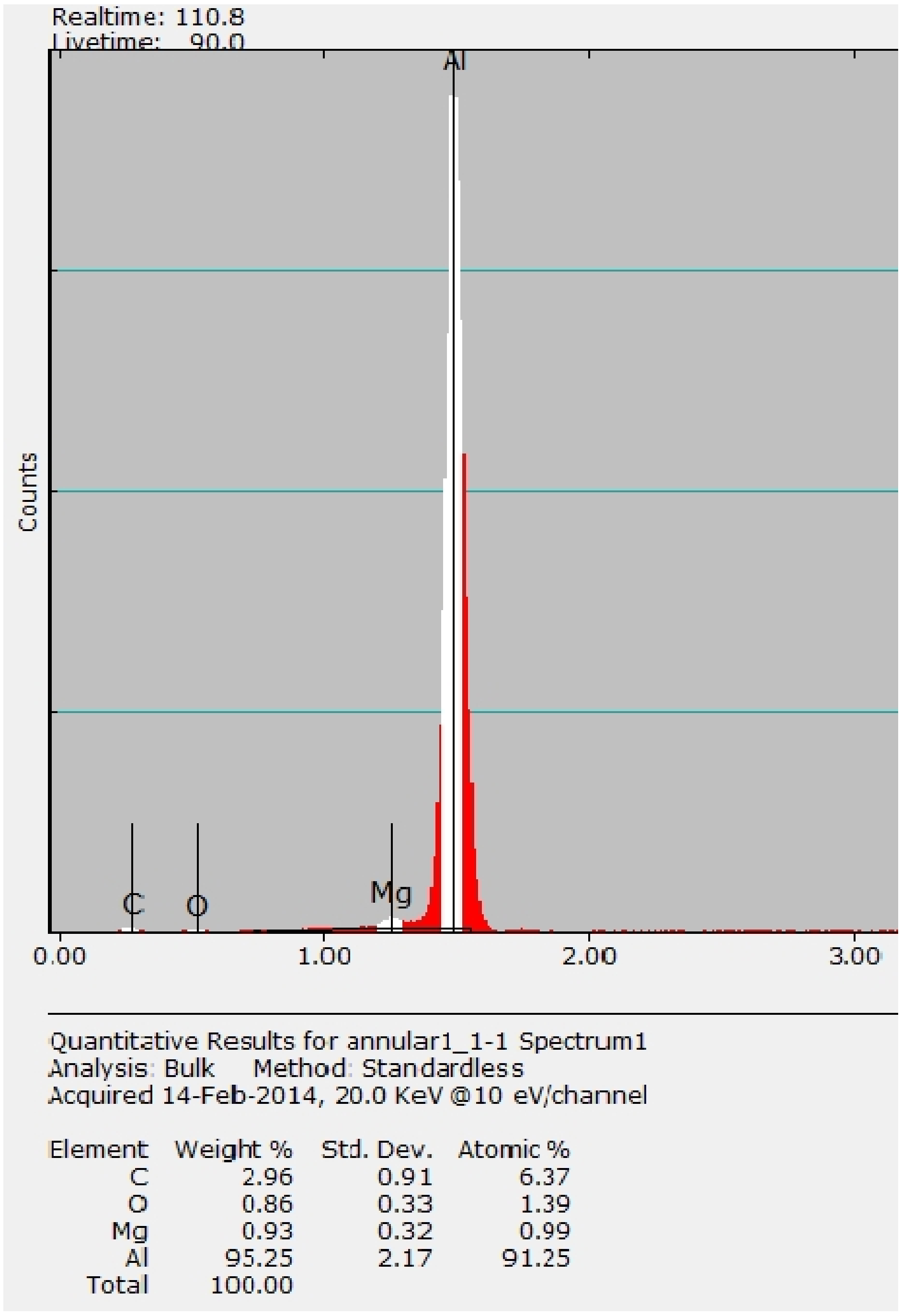


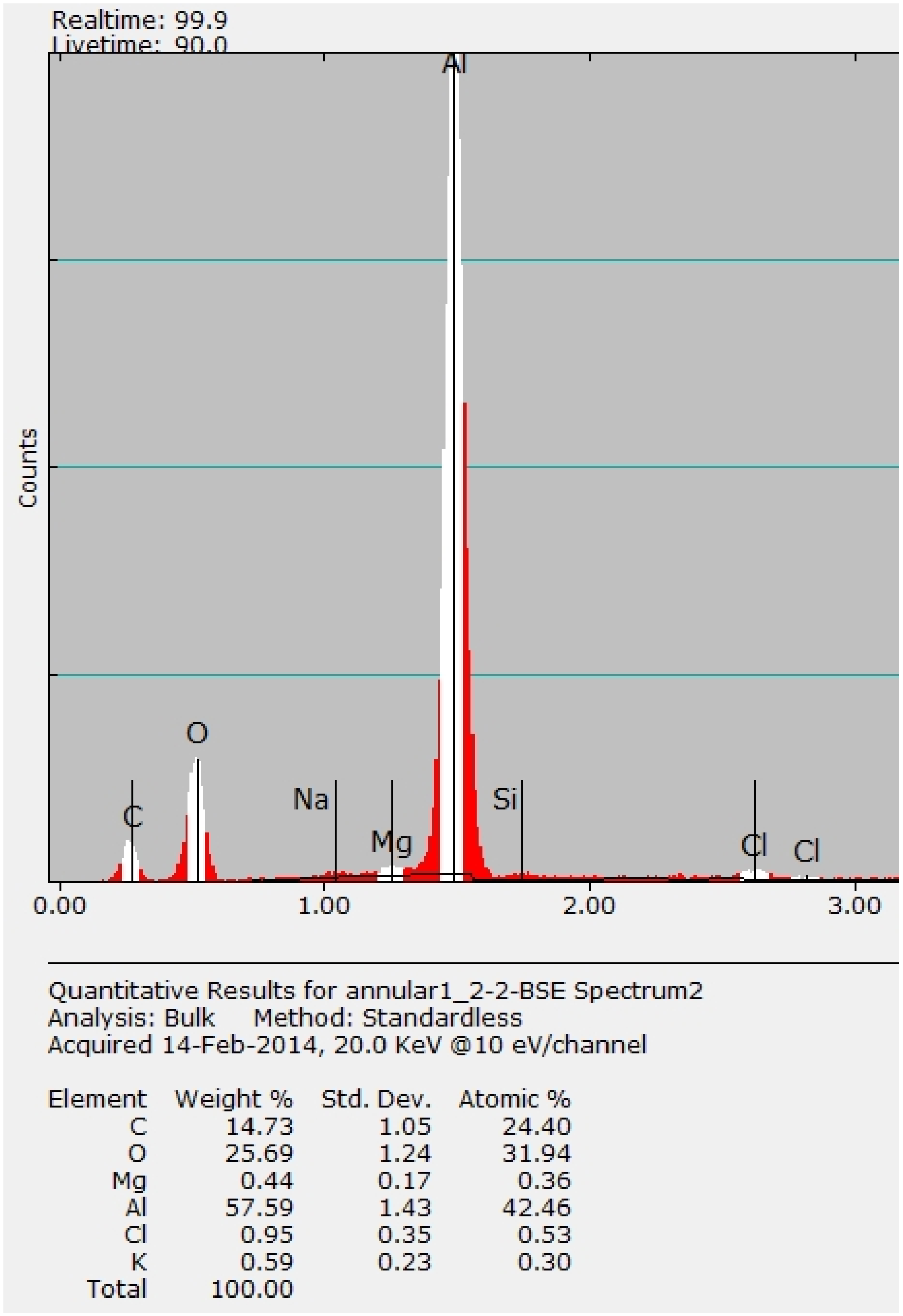
Figure 4.7 SEM image of aluminum electrode after plasma erosion. 1mm zoom. 45 degree tilt.



Quantitative Results for annular1_1-1 Spectrum1
Analysis: Bulk Method: Standardless
Acquired 14-Feb-2014, 20.0 KeV @10 eV/channel

| Element | Weight % | Std. Dev. | Atomic % |
|---|---|---|---|
| C | 2.96 | 0.91 | 6.37 |
| O | 0.86 | 0.33 | 1.39 |
| Mg | 0.93 | 0.32 | 0.99 |
| Al | 95.25 | 2.17 | 91.25 |
| Total | 100.00 | | |

Figure 4.8 Energy dispersive X-ray spec (EDS) for clean aluminum electrode



Quantitative Results for annular1_2-2-BSE Spectrum2
Analysis: Bulk Method: Standardless
Acquired 14-Feb-2014, 20.0 KeV @10 eV/channel

| Element | Weight % | Std. Dev. | Atomic % |
|---|---|---|---|
| C | 14.73 | 1.05 | 24.40 |
| O | 25.69 | 1.24 | 31.94 |
| Mg | 0.44 | 0.17 | 0.36 |
| Al | 57.59 | 1.43 | 42.46 |
| Cl | 0.95 | 0.35 | 0.53 |
| K | 0.59 | 0.23 | 0.30 |
| Total | 100.00 | | |

Figure 4.9 Energy dispersive X-ray spec (EDS) for plasma eroded aluminum electrode

considered. A relatively inexpensive choice is brass. Overall, brass performs much better than aluminum. Between 300-700 W, there is no plasma-metal interactions observed with OES or presence of pitting when the plasma is turned off. Typically aluminum begins to erode around 560 W. When the brass electrode is run between 700-1000 W, plasma-metal interactions are evinced by a plasma color change as well as an increase in the intensity of the emitted light. A comparison of the plasma OES with and without metal interactions is shown in fig. 4.11. The 560 W spectrum shows a more or less normal air plasma spectrum: NO bands between 230 and 290 nm (along with their 2x peaks around 500nm) and an OH band around 310 nm. However, the 945 W spectrum is dominated by sharp copper and zinc atomic lines. Despite the presence of copper and zinc in the discharge emission, no visual damage appears on the electrode surface when operated between 700 and 1000 W. However, if the power is raised too much over 1000 W, surface pitting and scarring analagous to the damage on the aluminum electrode are observed (see fig. 4.10).

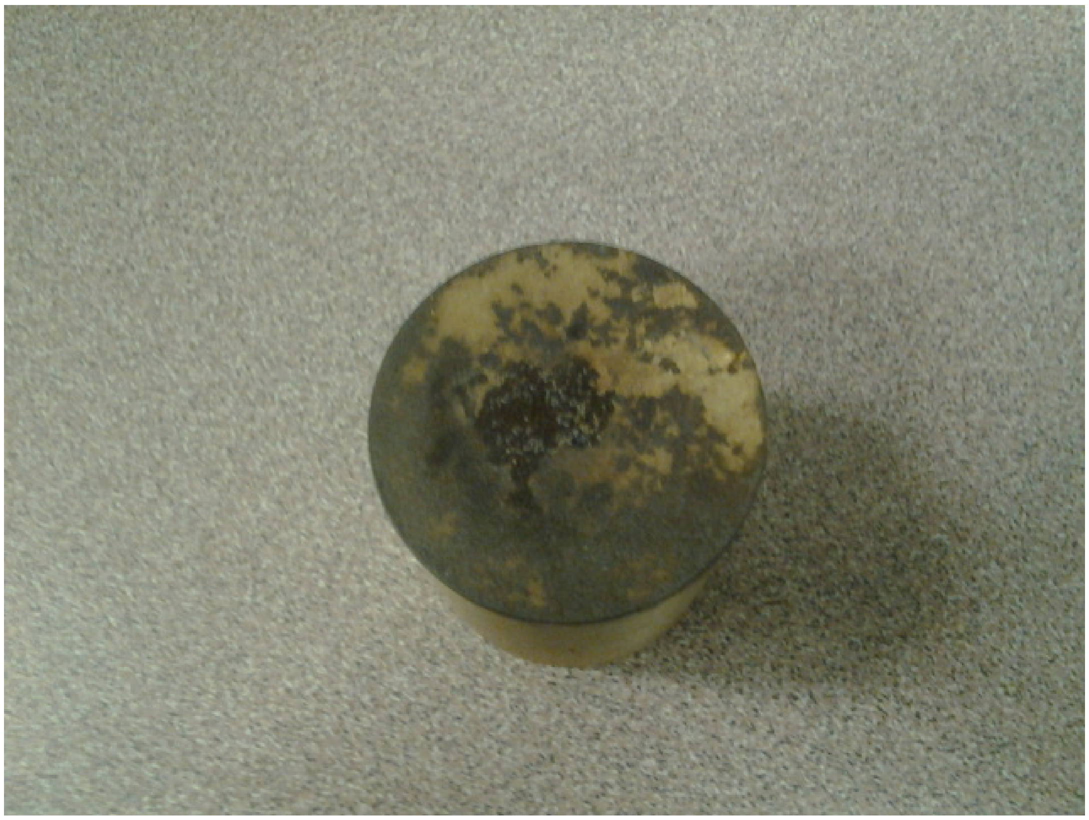


Figure 4.10 Image of brass electrode after plasma erosion

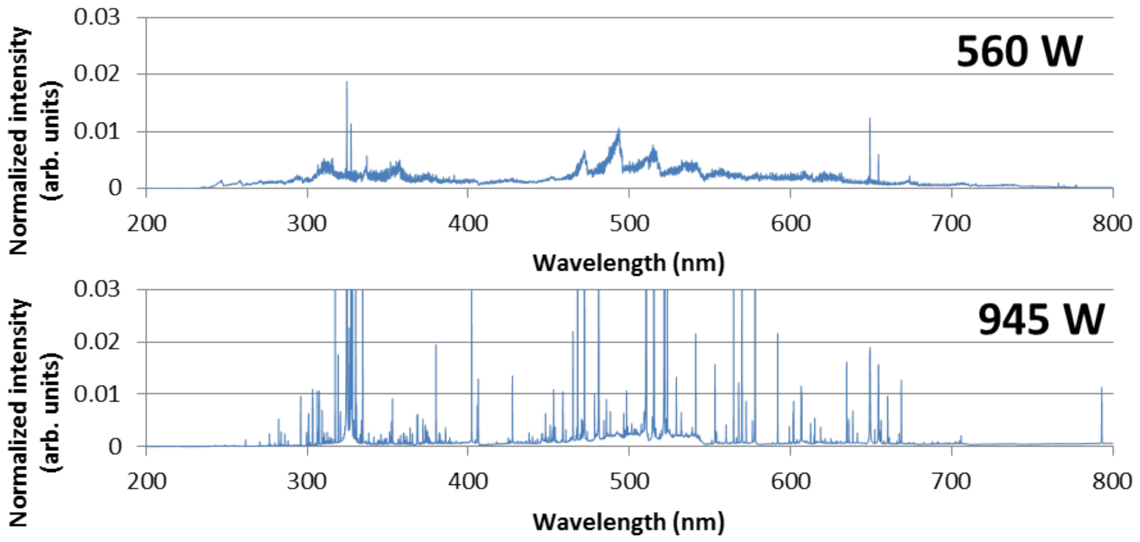


Figure 4.11 Top OES spectrum shows plasma emissions during normal operation with the brass electrode. The bottom spectrum shows emissions that occur during brass damage

## 4.5 Water Electrodes

An ideal plasma-liquid geometry has to provide both maximum interfacial contact between reactive plasma species and the liquid phase as well as system components that are resistent to plasma corrosion. Unfortunately, none of our previous configurations realize this goal. However, by utilizing the unique nature of the VHF source and recognizing that the entire coaxial structure is DC grounded, we can do something rather novel. We can apply a liquid layer to the surface of the powered electrode without worry of causing a short circuit. With this configuration, shown in fig. 4.12, the treated water is exposed to the most reactive part of the plasma. Both ion and electron fluxes to the water surface are anticipated to be much higher than in the batch configuration. Additionally, powered plasma-facing solid surfaces are completely eliminated from the geometry. The liquid surface is forever renewable and does not erode. This reduces system cost as well as experimental down-time.

The actual design of the water electrode can be seen in fig. 4.13. The electrode of most utility, the “pure” water electrode, is shown on the right. The pure water electrode has no powered

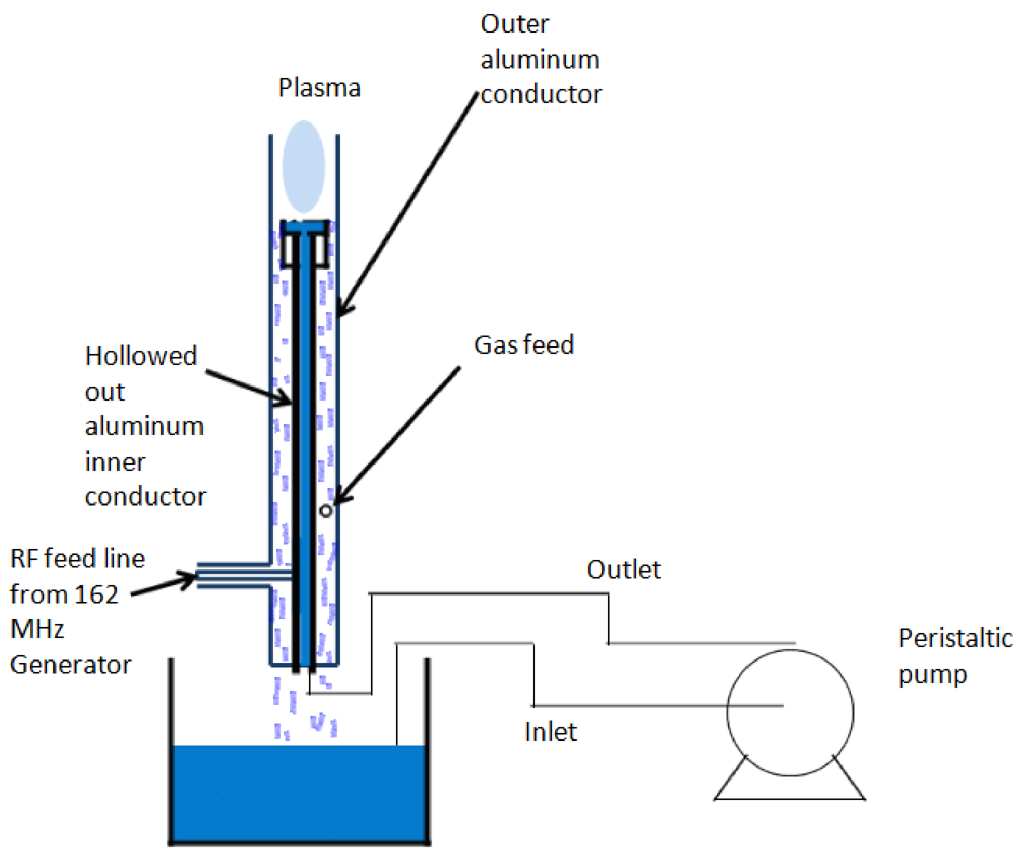

Plasma
Outer aluminum conductor
Hollowed out aluminum inner conductor
Gas feed
RF feed line from 162 MHz Generator
Outlet
Peristaltic pump
Inlet

Figure 4.12 Representative experimental set-up for using a "water" electrode

metal surfaces facing the plasma; the powered plasma-facing surface is 100% water. If some plasma-metal contact is desired, for instance if the contact favorably modifies some plasma or liquid application variable, then the "annular" electrode shown on the left can be used.

### 4.5.1 Circuit Analysis

In order to better understand the coupling of the RF power to the plasma-liquid system, it is useful to construct a circuit model. The first question the circuit model should answer is whether conduction current coming from the inner conductor propagates along the water electrode surface or the underlying aluminum. This is done by comparing the relative resistances presented by the water and aluminum surfaces, treating both as conductors. The resistance is calculated using the relationship:

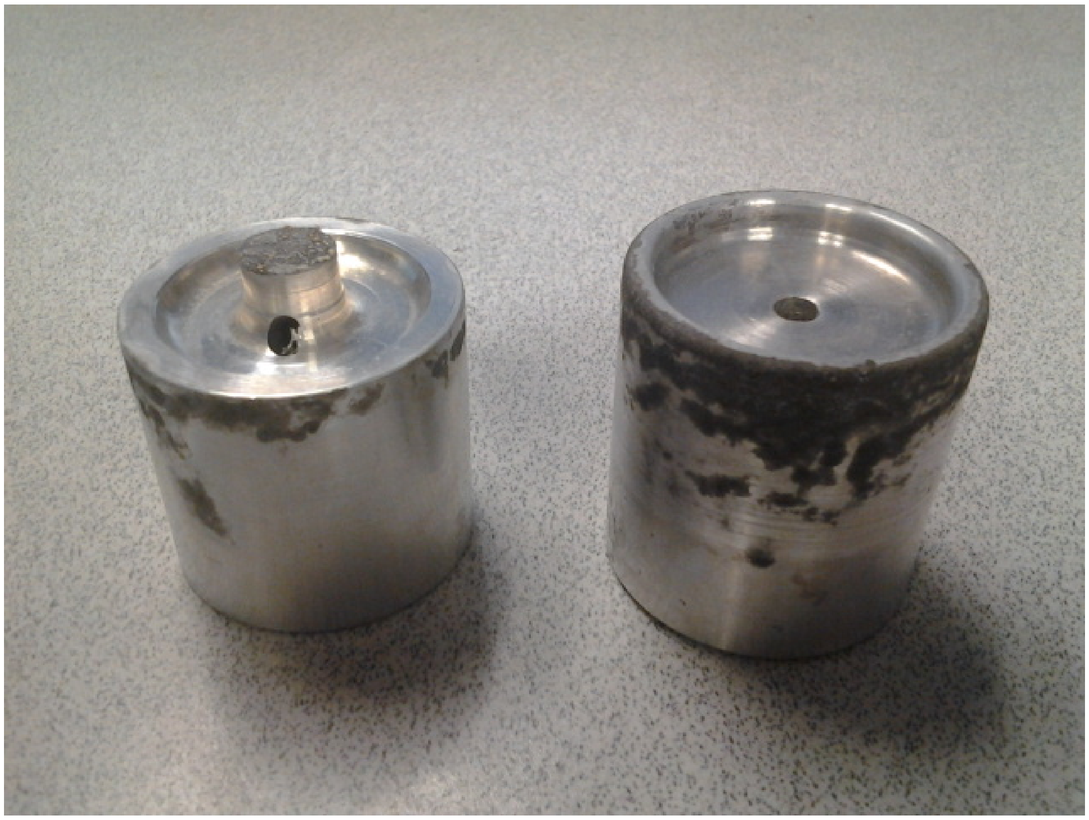


Figure 4.13 Image of the two versions of "water" electrodes. The "annular" version still allows a small metallic area of plasma contact. In the "pure" version, the plasma has no metallic content with the powered electrode. The powered surface is entirely composed of water.

$$R = \frac{L\rho}{A} \tag{4.1}$$

where $R$ is the resistance, $L$ is the length of the conducting surface, $\rho$ is the resistivity of the medium, and $A$ is the cross-sectional area through which the conduction current can flow. For current propagation across the top face of the cylindrical electrode, we approximate $L$ by the radius $r$ of the electrode, and $A$ by the product of the skin depth $\delta$ and the radius $r$. The skin depth $\delta$ is calculated using:

$$\delta = \sqrt{\frac{2\rho}{\omega\mu_B}} \tag{4.2}$$

where $\omega$ is the driving frequency in radians/s and $\mu_B$ is the material permeability, set equal to $\mu_0 = 4\pi \times 10^{-7}$. For aluminum, $\rho$ is set equal to its literature value at $20^{circ}$ Celsius, $2.82 \times 10^{-8}\Omega m$. A typical tap water conductivity of 50 mS/m is used to calculate the resistivity of water, $\rho = \frac{1}{\sigma}$. The corresponding skin depth for water at 162 MHz is .18 m, which is significantly larger than the milimeter depth of the water layer. As a consequence, $A$ for eq. (4.1) is calculated with $\delta = 1$ mm. With these definitions,tThe resistance of the water surface to conduction current is calculated to be 20 thousand $\Omega$ at 162 MHz. The skin depth of aluminum at 162 MHz is 6 $\mu$m. The corresponding resistance to conduction current is 4 m$\Omega$ at 162 MHz. One can ask whether plasma modification of the water surface might substantially decrease the water resistance; however, because the mobility of electrons in water is so much lower than their gaseous mobility, the effect of the plasma is nowhere near enough to overcome the seven order of magnitude difference in resistance between water and aluminum. This analysis suggests that all of the conduction current coming from the feed line propagates along the underlying aluminum electrode as opposed to the water surface. A frequency analysis shown in fig. 4.14 demonstrates that conduction current will likely flow through the underlying aluminum regardless of the device operating frequency.

The demonstration that conduction current likely does not propagate along the water *surface* means that current, most likely in the form of displacement current, must pass through the water *volume*. The question then becomes: what is the relative split in dissipated power between the water and plasma? Where is the potential drop occuring? These are answered by treating the water volume and plasma as lossy dielectrics. We define the medium dielectric constant by:

$$\epsilon = \epsilon_r \epsilon_0 \left(1 - \frac{\omega_c^2}{\omega\left(\omega - \nu j\right)}\right) \tag{4.3}$$

where $\epsilon_r$ is the relative dielectric constant, $\epsilon_0$ is the permittivity of free space, $\omega_c$ is the characteristic frequency coming from oscillatios of free charges in the medium, $\omega$ is again the driving frequency, $\nu$ is the rate of collisions of charges with background molecules, and j is the imaginary number $\sqrt{-1}$. For the plasma, $\epsilon_r = 1$; for the water, $\epsilon_r = 80$. Because the free electrons are much lighter than their corresponding ions, $\omega_c$ in the plasma is essentially equal to the plasma electron frequency $\omega_{pe}$. In the water, $\omega_c$ is calculated assuming sodium and chloride charge carriers. $\omega_c$ in both plasma and water is calculated using the equation:

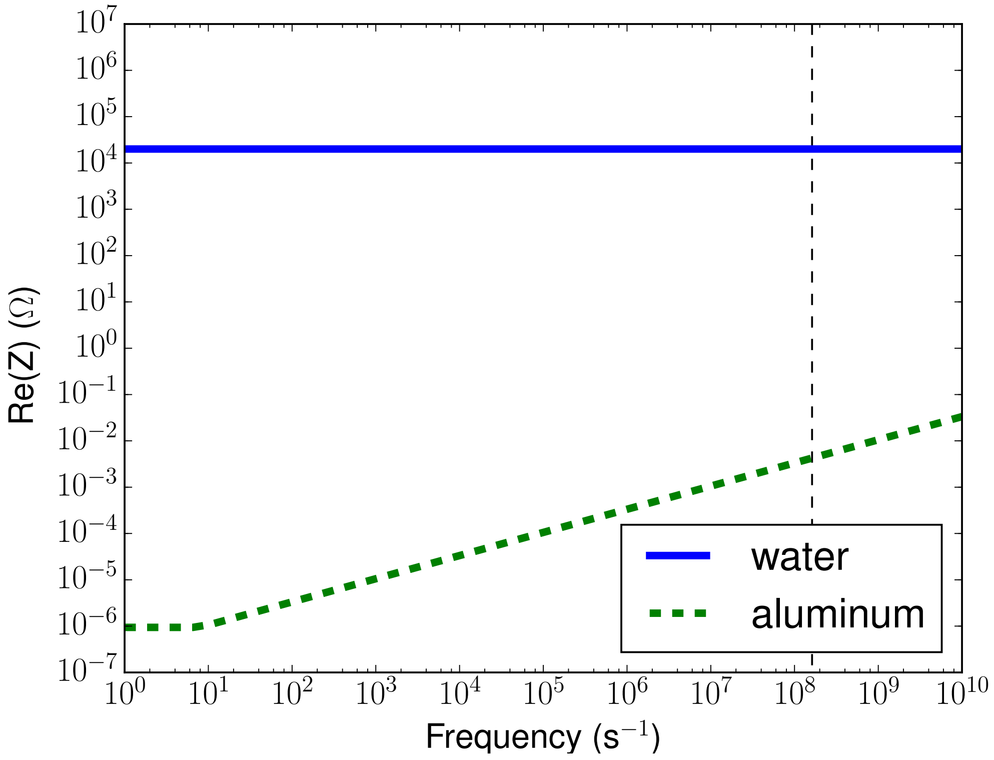


Figure 4.14 Resistance to flow of conduction current for aluminum and water for a range of frequencies. Vertical, black, dotted line indicates the 162 MHz operating frequency of the NCSU source. Aluminum is orders of magnitude less resistive for all frequencies considered; consequently the current propagating along the feed line is likely to prefer the underlying aluminum electrode over the water surface.

$$\omega_c = \sqrt{\frac{e^2 n_0}{\epsilon_0 m}} \tag{4.4}$$

where $e$ is the Coulombic charge, $n_0$ is the number density, and $m$ is the particle mass. For the plasma, $\nu$ is calculated using:

$$\nu = n_g \sqrt{\frac{\pi \alpha e^2}{m_e \epsilon_0}} \tag{4.5}$$

where $n_g$ is the background gas density and $\alpha$ is the polarizability equal to $2.1 \times 10^{-29} m^3$ for air. For the water, $\nu$ is determined via

$$\nu = \frac{e}{\mu m} \tag{4.6}$$

where $\mu$ is the mobility calculated with Einstein's relation

$$\mu = \frac{De}{k_b T} \tag{4.7}$$

where D is the diffusivity of sodium and chloride ions, equal to $2 \times 10^{-9} m^2 s^{-1}$ [130], $k_b$ is Boltzmann's constant, and T is the temperature of the water (assumed equal to 300 K). Once the medium dielectric constant $\epsilon$ is calculated, the medium admittance is computed using the approximation of a parallel plate capacitor:

$$Y = \frac{\omega \epsilon A j}{d} \tag{4.8}$$

Finally, the impedance $Z$ is computed using the simple relation, $Z = \frac{1}{Y}$. For a lossy dielectric, the impedance $Z$ is complex, e.g. $Z = R + Xj$ with R the resistance and X the reactance. Figure 4.15 compares the plasma and water resistance over a wide range of frequencies. Over the entire range of frequencies presented the water resistance is $< 1\%$ of the plasma resistance. At the operating frequency of the VHF source, the water resistance is close to six orders of magnitude less than the plasma resistance, suggesting that virtually all of the RF power is dissipated in the plasma.

Figure 4.16 compares the magnitude of the plasma and water reactance as a function of frequency. For both mediums, the reactance is negative for all frequencies, consistent with capacitive behavior. (Lower gas pressures lead to more inductive behavior.) For frequencies $< 1$ MHz, the plasma and water reactance are roughly equivalent. Beyond 1 MHz, the magnitude of the water reactance decreases while the magnitude of the plasma reactance continues to increase until roughly 80 MHz beyond which it begins to decline. Figure 4.17 compares the magnitude of the impedance ($|Z| = \sqrt{R^2 + X^2}$) for plasma and water for a range of frequencies. The impedance magnitude for water is determined primarily by its resistive component below 1 MHz and by its reactive component above 1 MHz. The behavior for the plasma is similar except the transition occurs around 30 MHz. The result is that the plasma impedance magnitude is

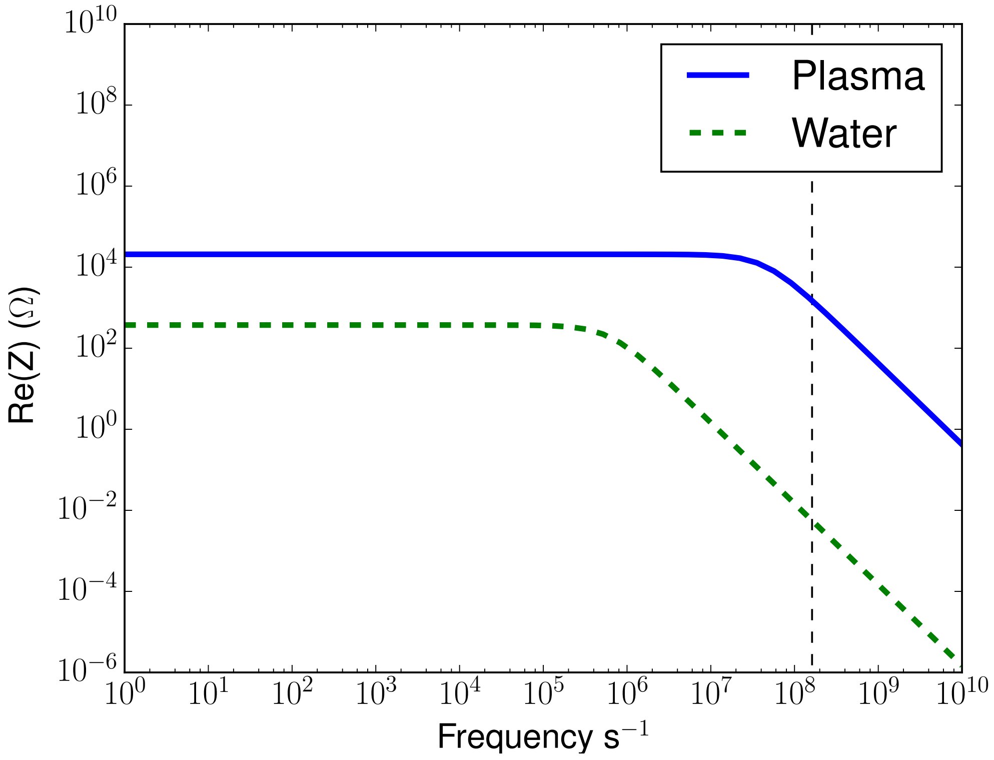


Figure 4.15 Comparison of plasma and water resistances over a range of operating frequencies. Over the whole domain, the water resistance is $< 1\%$ the plasma resistance. At 162 MHz they differ by six orders of magnitude, suggesting that all of the RF power is dissipated in the plasma as opposed to the water.

always a couple of magnitudes larger than the water impedance magnitude for all frequencies, ensuring that the majority of the potential drop always occurs across the plasma as opposed to the water.

### 4.5.2 Optical Emission

Figure 4.18 compares typical OES spectra obtained for annular and pure water electrodes running at 700 W. Evidence of plasma-metal contact with the annular electrode is evident in the presence of aluminum atomic lines and AlO molecular bands. Additionally, there is a sodium line from sputtering of the tap water. The pure water electrode spectrum is much less intense and consists only of OH bands.

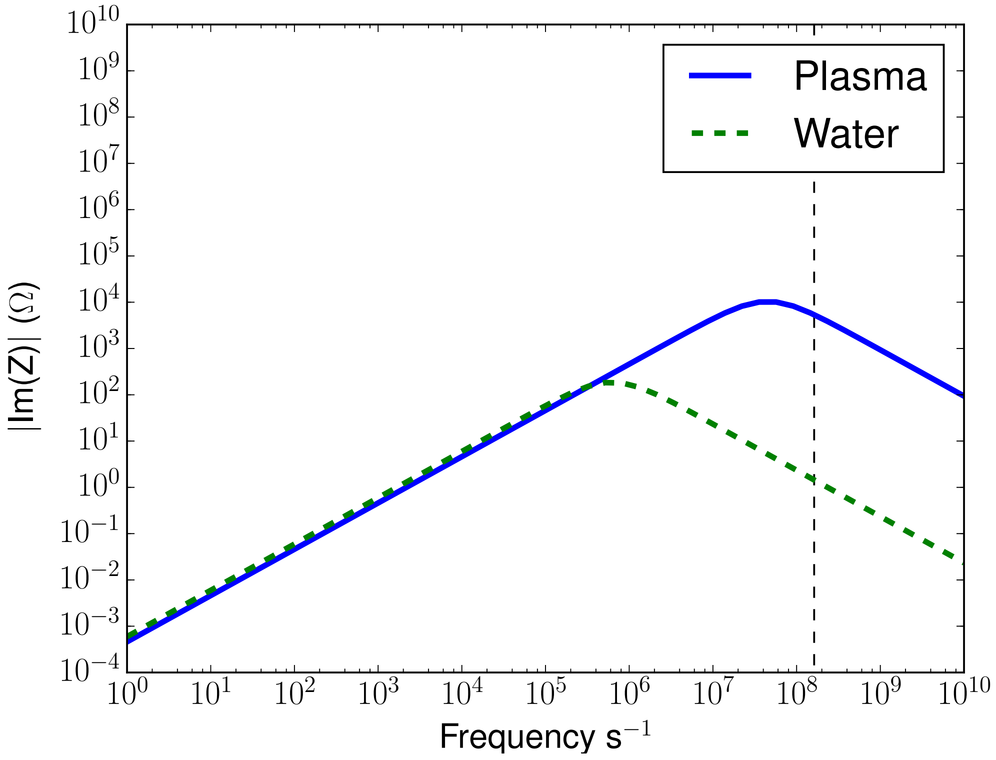


Figure 4.16 Comparison of plasma and water reactance magnitude for a range of frequencies. Magnitudes are roughly equivalent up to 1 MHz, where water reactance magnitude begins to decline. Plasma knee occurs around 80 MHz.

Figure 4.19 shows the effect of increasing power on the optical emission spectrum with the pure water electrode. Because the plasma is in immediate contact with the water surface and not any solid surfaces, the device can be operated at much higher powers. Whereas with a metal electrode the source cannot be run at powers much greater than 700 W without significant damage to the electrode, the pure water electrode can be run up to 1155 W. The only reason that the device cannot be operated at even higher powers is because of the increased difficulty in matching impedances using the main and stub lines; reflected power becomes high enough to potentially damage the generator.

Up above 1000 W, aluminum atomic lines and AlO bands become apparent (fig. 4.19). The presence of the aluminum associated emissions is interesting because the metal is removed from the gaseous discharge region by the few milimeter thick water layer; this probably explains why

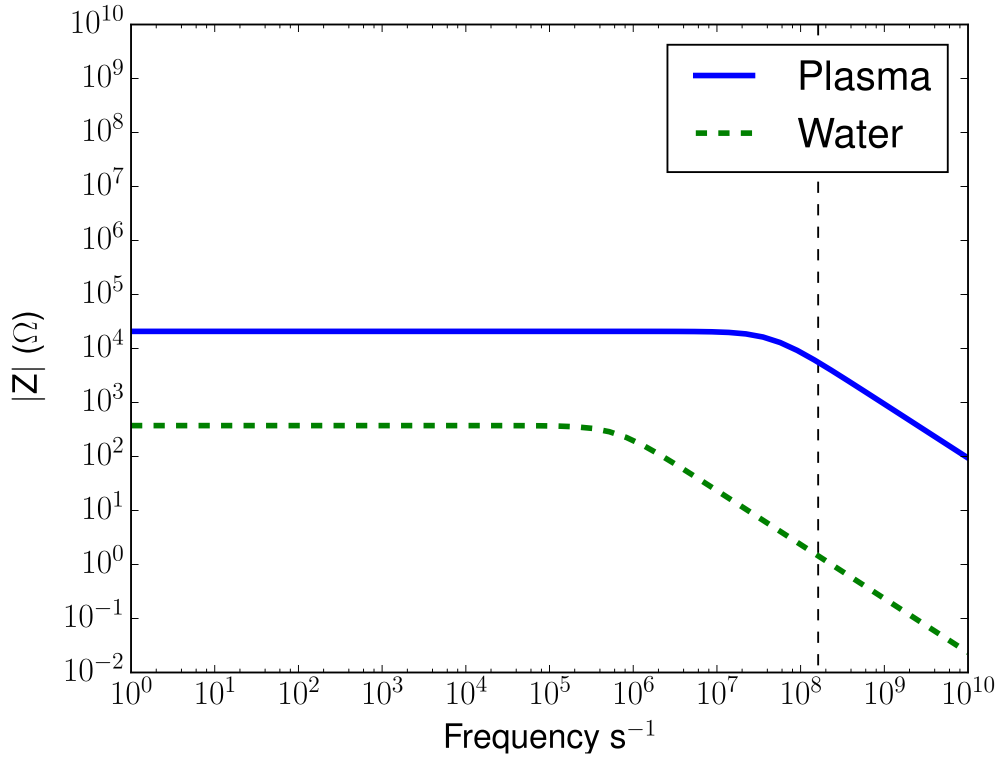


Figure 4.17 Impedance magnitudes for plasma and water as a function of frequency. Plasma impedance magnitude is significantly larger over the whole frequency domain.

the intensities are significantly below that of discharges with direct plasma-metal contact (see fig. 4.18). However, the existence of any aluminum lines at all implies that the discharge is penetrating the aqueous phase to reach the metal. This is perhaps indirect evidence of high charged particle fluxes to the water electrode surface, suggesting that the pure water electrode design is a good one for maximizing interactions between the plasma and aqueous phases.

### 4.5.3 Absorption Work

Some idea of the magnitude of OH produced by the water electrode geometry can be gained by performing absorption spectroscopy. A picture of the experimental set-up is shown in fig. 4.20. The diameter of the plasma column is approximately 2 cm. We pass a broadbeam light source through slits cut in the outer conductor of the VHF shource. Mirrors on each slit side can be

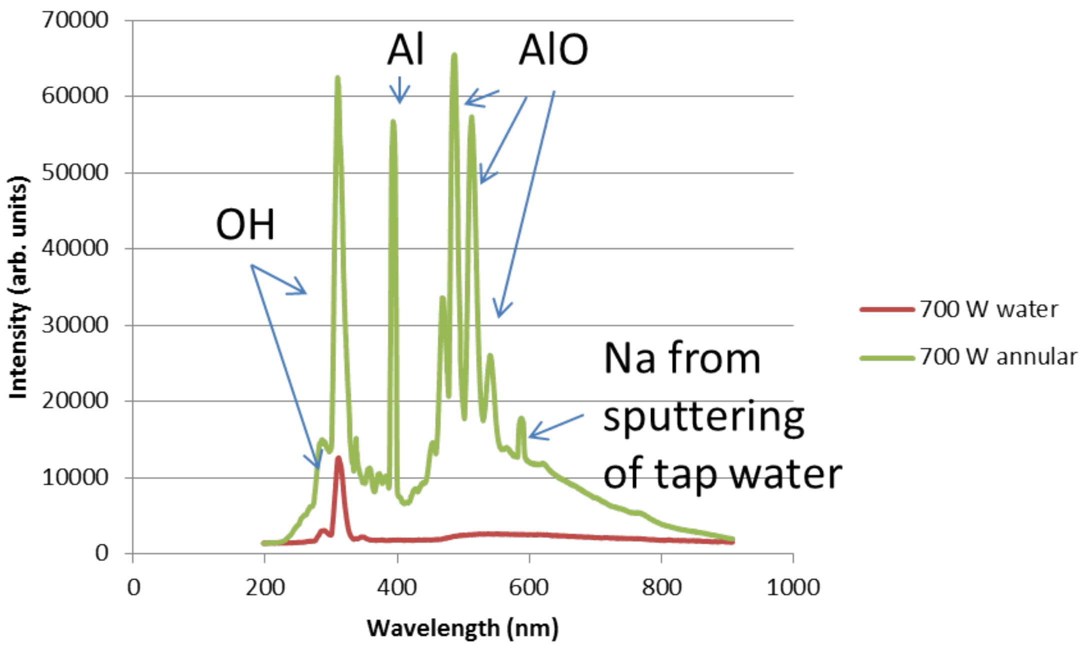


Figure 4.18 Comparison of OES spectra for annular and pure water electrodes

used to route the light beam through the plasma column for a total of up to 4 passes and a path-length up to 8 cm. After passing through the plasma, the beam enters an optical fiber connected to a high resolution spectrometer. The spectrum of the broadband light source in the absence of plasma is presented as series “no-plasma” in fig. 4.21. When the plasma is turned on, there is significant attenuation of the broadband signal in the region of the OH X-A electronic transition with just a single pass through the plasma as shown in fig. 4.21. The y-axis data for fig. 4.22 is calculated using:

$$1 - \frac{I - I_p}{I_0} \tag{4.9}$$

where $I$ is the spectrum taken with both the broadband light source and plasma, $I_p$ is the spectrum of plasma only, and $I_0$ is the spectrum of just the broadband light source. The fingerprint of the OH X-A transition is evident.

We can approximately calculate the density of ground-state OH in the plasma using Beer-Lambert’s law:

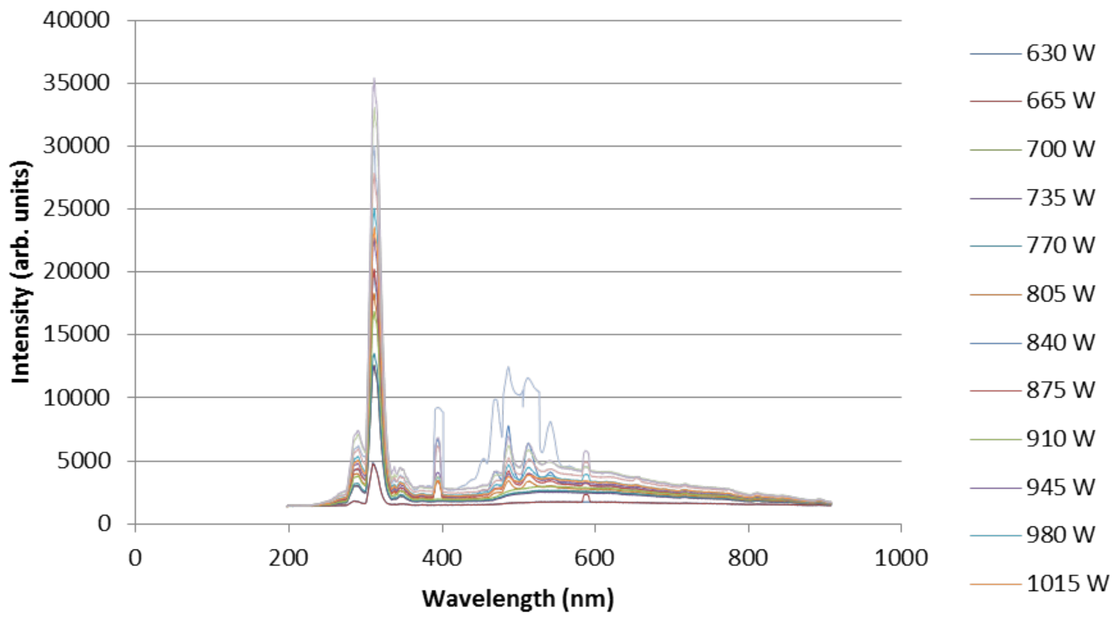


Figure 4.19 OES spectra showing power sweep with pure water electrode. Relatively small aluminum peaks grow in at very high powers. Cause of aluminum peak deformations unknown, but speculation is that it could be signal attenuation by the water layer

$$\frac{I - I_p}{I_0} = exp\left(-\sigma(\lambda)LN\right) \tag{4.10}$$

where $\sigma$ here is the cross section in units of area for absorption of light of wavelength $\lambda$, $L$ is the path length for absorption, and $N$ is the density of the absorbing species, OH in this case. Because the OH(X-A) transition is spread over a variety of vibrational and rotational states, one can choose a wavelength range to integrate over. As indicated by the highlight in figs. 4.21 and 4.22, the wavelengths spanning the P branch, roughly 309-309.5 nm, are chosen for integration. When performing the integration,

$$\int_{309}^{309.5} \left(1 - \frac{I - I_p}{I_0}\right) d\lambda = \int_{309}^{309.5} \left(1 - exp\left(-\sigma(\lambda)LN\right)\right) d\lambda \tag{4.11}$$

the integrand on the RHS can be simplified because the argument of the exponential is small, allowing the approximation $exp(-x) \approx 1-x$. After performing this substitution, the concentration

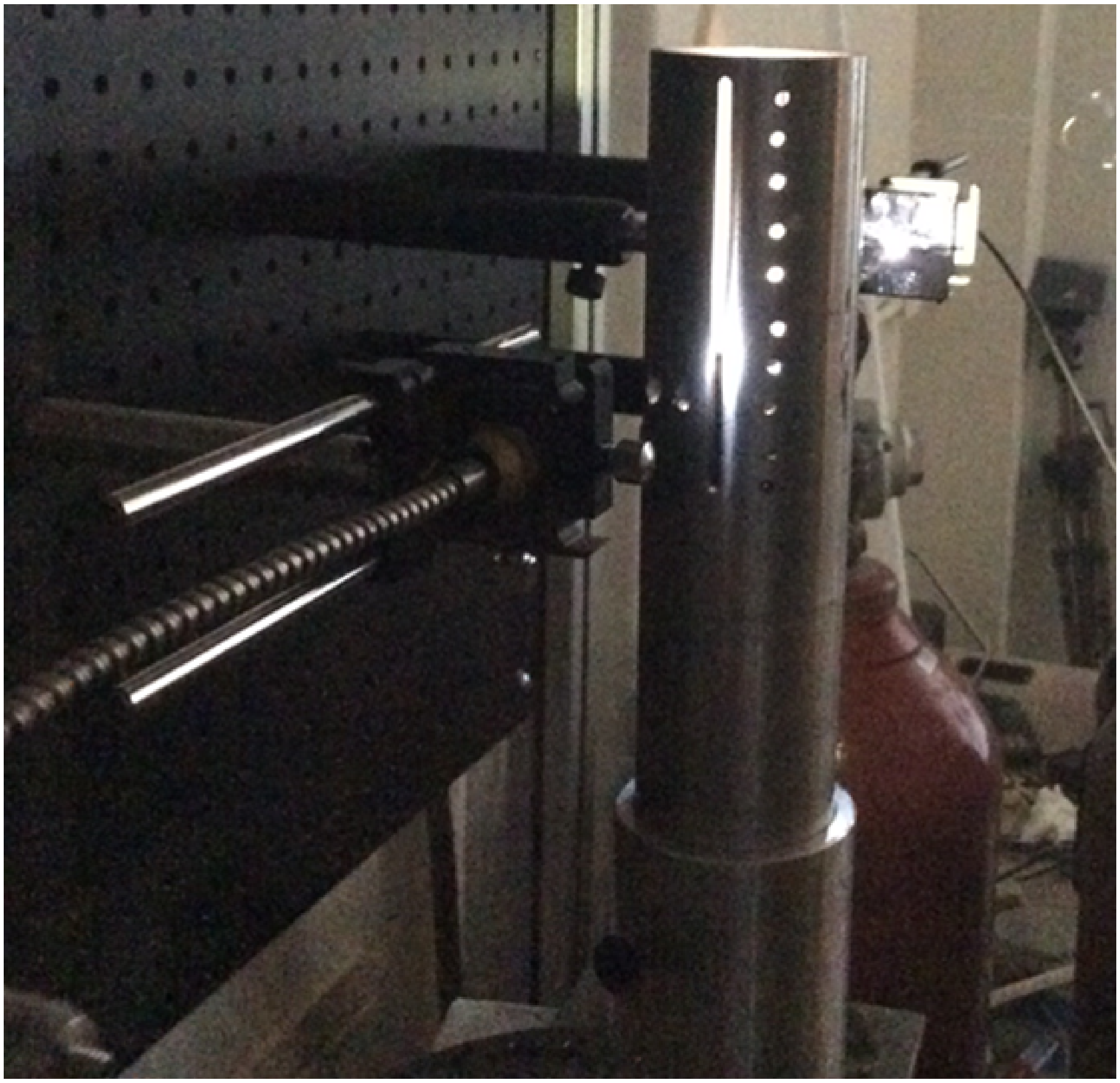


Figure 4.20 Expermental set-up for absorption spectroscopy with the water electrode.

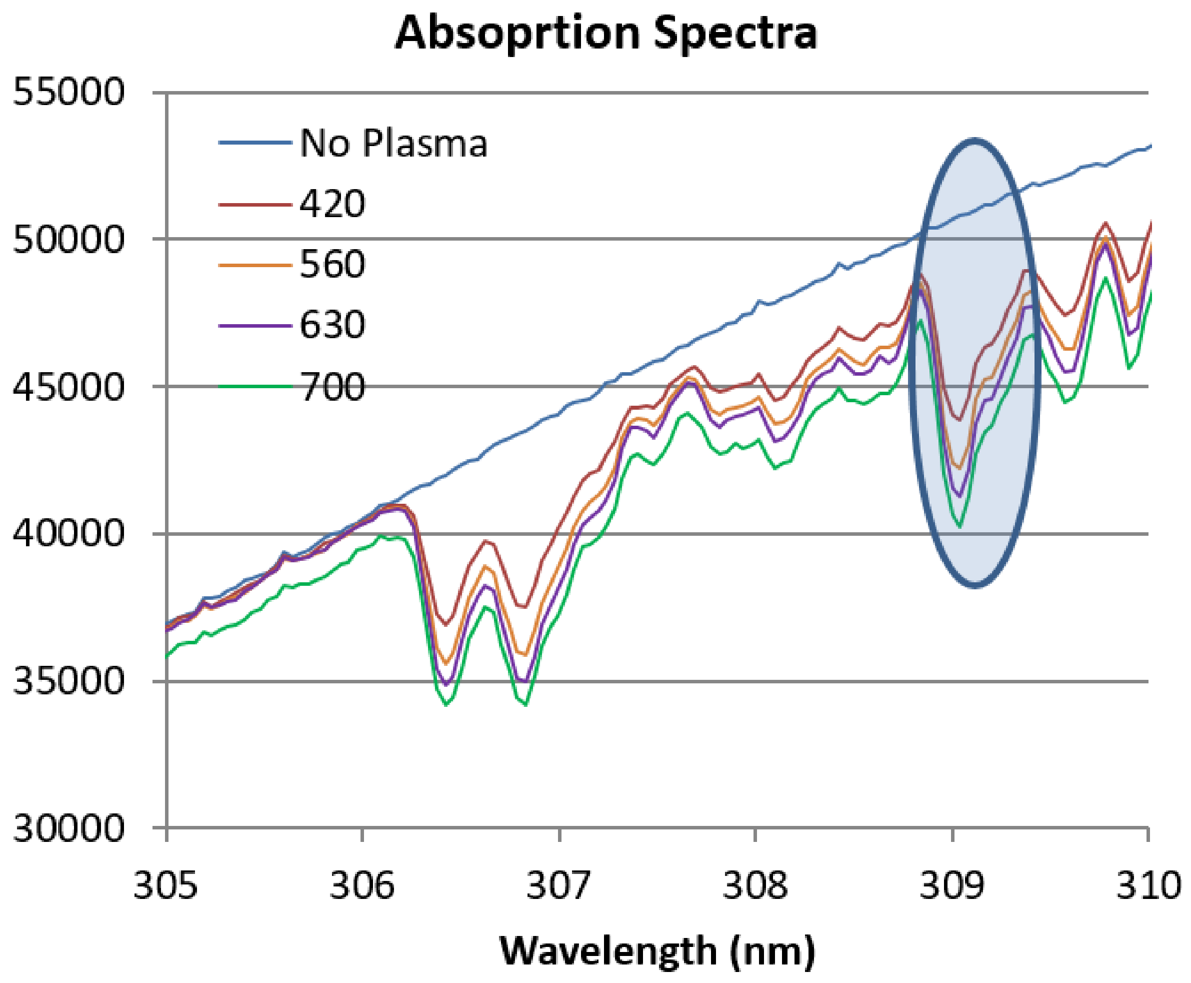


Figure 4.21 Raw optical spectra in the OH wavelength region for different plasma powers and a path length of 2 cm (single pass). The series “No plasma” shows the light intensity of just the broadband light source. The other series clearly show the absorption of light by gaseous OH radicals. Highlighted region shows the area of integration for later calculation of OH densities

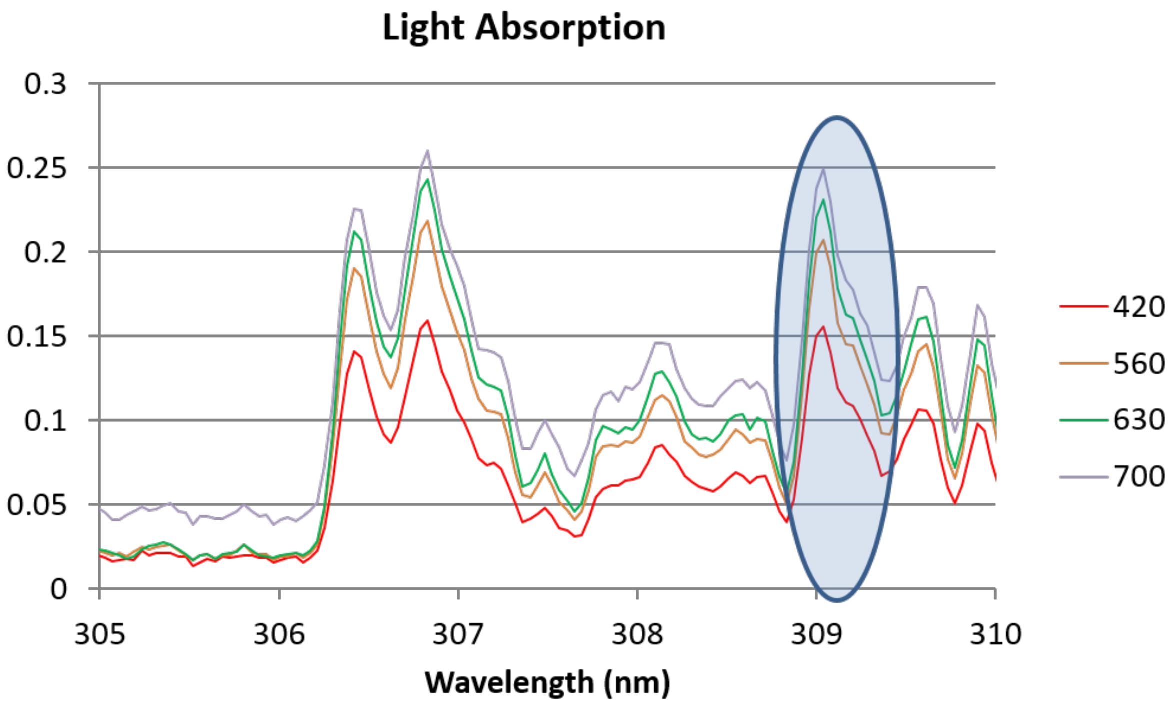


Figure 4.22 Net results obtained by subtracting plasma absorption spectra from broadband light source spectrum and normalizing. Obvious OH X-A transition fingerprint. Highlighted region shows the area of integration for later calculation of OH densities

of OH can be calculated using:

$$N = \frac{\int_{309}^{309.5} \left(1 - \frac{I - I_p}{I_0}\right) d\lambda}{L \int_{309}^{309.5} \sigma(\lambda) d\lambda} \tag{4.12}$$

Figure 4.23 shows the density of OH as a function of distance from the powered electrode for powers of 455, 560, and 665 W. OH densities are on the order of $10^{15}$ cm$^{-3}$. The density decreases monotonically with increasing distance from the powered electrode.

Figure 4.24 shows the OH density as a function of power at a distance of 8 cm from the powered electrode. From 455 to 700 W the OH density increases linearly from a minimum value of 2x$10^{14}$ cm$^{-3}$ to a maximum value of 1.2x$10^{15}$ cm$^{-3}$. These high OH densities could be valuable in applications requiring a high degree of oxidative stress, such as in various fields of plasma

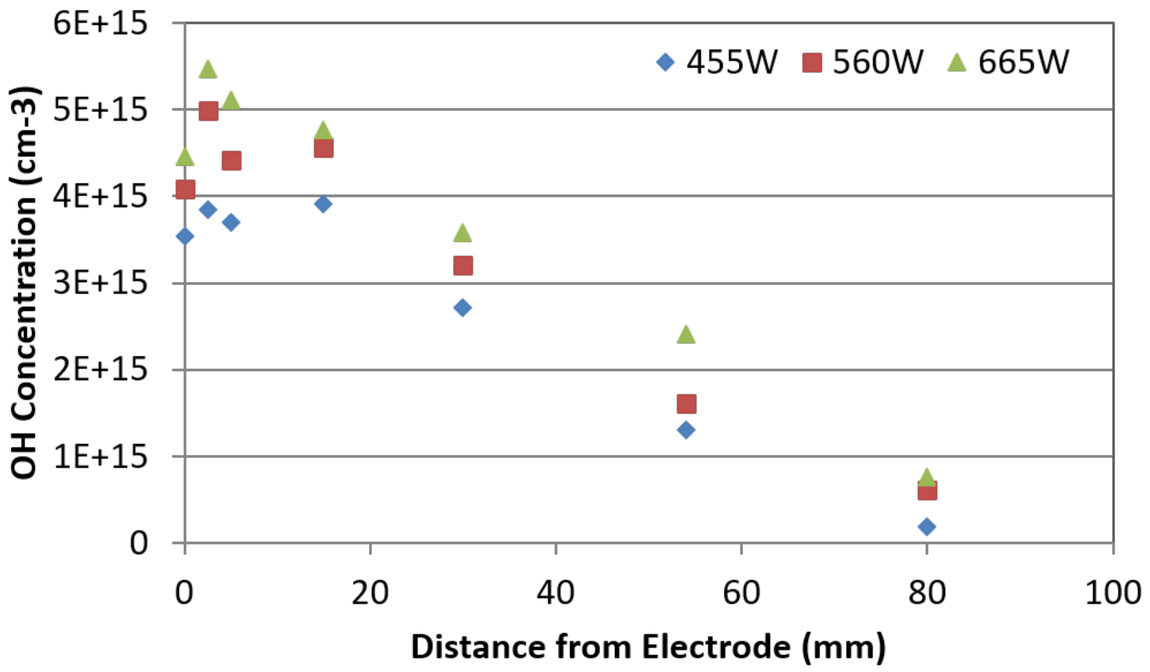


Figure 4.23 OH density versus position from the powered electrode for a variety of powers

medicine and pollutant degradation. Concentrated OH could be a key player in the degradation of persistent chemicals like dioxane and PFOS, as described in section 5.2.

## 4.6 Exploring Aqueous Chemistry Generated by Plasma-Liquid Interactions

In addition to plasma characterization with OES and absorption spectroscopy, additional research has focused on optimizing and understanding generation of nitrates and nitrites in aqueous solution. Several variables have been explored, including power supplied by the 162 MHz generator, flow rate of the feed gas, type of interface between the plasma and water phases, and the effect of aqueous impurities, particularly basic species. The majority of experiments were performed using the experimental set-up shown in fig. 4.1. However, the greenhouse sprayer scheme shown in fig. 4.2 was also employed. The number of impurities and basic species in

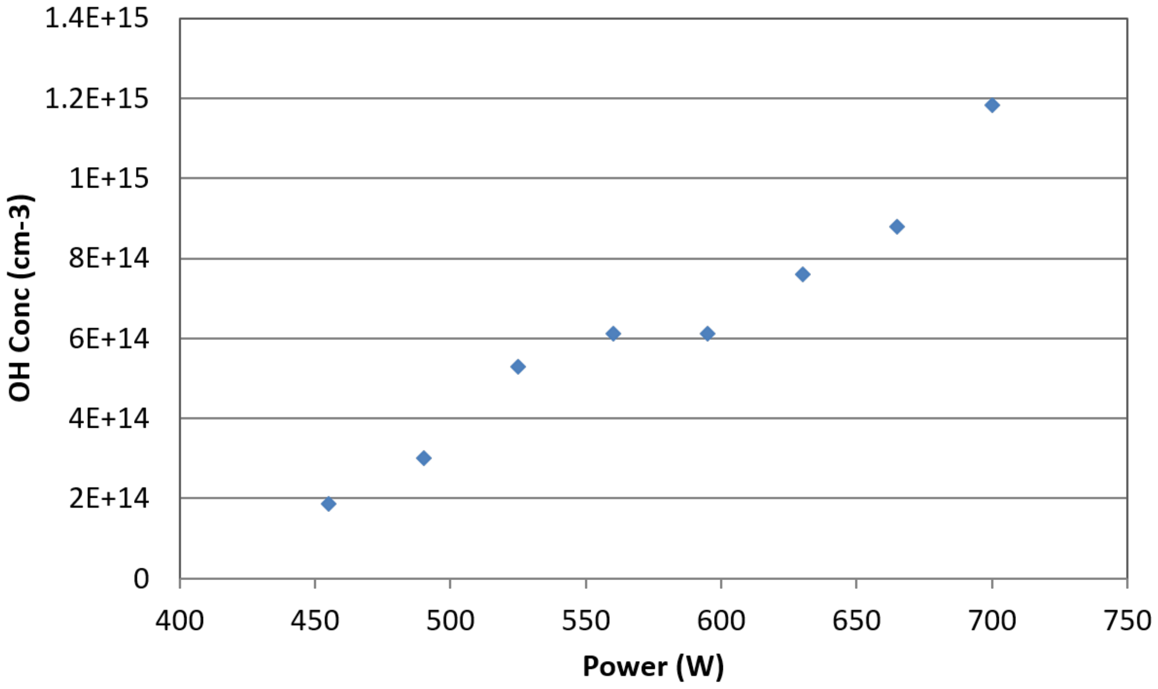


Figure 4.24 OH density versus power 8 cm from the powered electrode. Clear increasing trend of OH density with power

water were controlled in two manners. The first was the choice between distilled and tap water, with the former containing negligible impurities and the latter containing impurities found in Raleigh's municipal water supply; these impurities are summarized in table 4.1. The most relevant item in table 4.1 is the alkalinity, which comes primarily from the carbonate system. At a pH of 8.4, it is reasonable to assume that the tap water alkalinity is completely due to bicarbonate. [131] Using this assumption, the concentration of bicarbonate in tap water is .50 mmol/L. The concentration of bicarbonate can also be directly controlled by adding measured amounts of $NaHCO_3$. $NaHCO_3$ can be added pre- or post-exposure depending on the experiment. The motivation for adding basic species like $NaHCO_3$ to solution is that they are known to react with dissolved NO and $NO_2$ to form nitrite. [132] Thus basic species concentrations can be a control knob for adjusting the nitrogen chemistry in PAW.

At a gas flow rate of .14 $m^3$/min, an exposure time of 3 minutes, and a treatment volume of 150 mL distilled water, nitrate concentrations were determined for powers ranging from 385 to 630 W and are shown in fig. 4.25. For a better comparison with spray treatment results

Table 4.1 Impurities in Raleigh tap water

| | |
|---|---|
| pH | 8.4 |
| Free $CO_2$ | .23 |
| Total alkalinity (mg/L as $CaCO_3$) | 24.8 |
| Total hardness (mg/L as $CaCO_3$) | 24.4 |
| Total dissolved solids (mg/L) | 150 |
| Specific conductivity ($\mu$S/cm) | 225 |
| Iron (mg/L) | .01 |
| Manganese (mg/L) | .02 |
| Fluoride (mg/L) | .78 |
| Chloride (mg/L) | 13.3 |
| Silica (mg/L) | 8.12 |
| Silt density index (SDI) | 5.00 |

shown in fig. 4.29, the horizontal axis is defined in terms of the energy deposited in the plasma per mass of water exposed to the plasma. The results indicate a general downward trend in nitrate concentration with respect to power and total energy deposition. To decouple the effects of power and total energy deposition, a second experiment was conducted in which treatment times were varied with power in order to keep the total energy delivered to the plasma constant. Consequently, whereas the exposure time was 3 minutes for a 420 W plasma, exposure time was only 2 minutes for a 630 W plasma for a constant plasma energy deposition of 75.6 kJ; for comparison with fig. 4.25, the energy deposited in the plasma per mass of exposed water was 504 kJ/kg. Gas flow was again .14 $m^3$/min and water volumes were 150 mL. Results from the second experiment are shown in fig. 4.26. Though the plasma energy deposition is constant, nitrate concentrations in both tap and distilled water samples decrease with increasing power, consistent with the trend in fig. 4.25. Though no nitrite appears in distilled water samples, nitrite concentrations increase in tap water with increasing power. The total nitrogen anion levels in tap and distilled water samples are within experimental error for powers between 420 and 560 W.

Another variable explored was gas flow rate. For an exposure of 3 minutes, a treatment volume of 150 mL distilled water, and a plasma power input of 420 W, nitrate concentrations were measured for flow rates of .08, .11, and .14 $m^3$/min and are recorded in fig. 4.27. A factor of 4.9 improvement in nitrate concentration is observed between .08 and .14 $m^3$/min flow settings. Again no detectable amount of nitrite was observed.

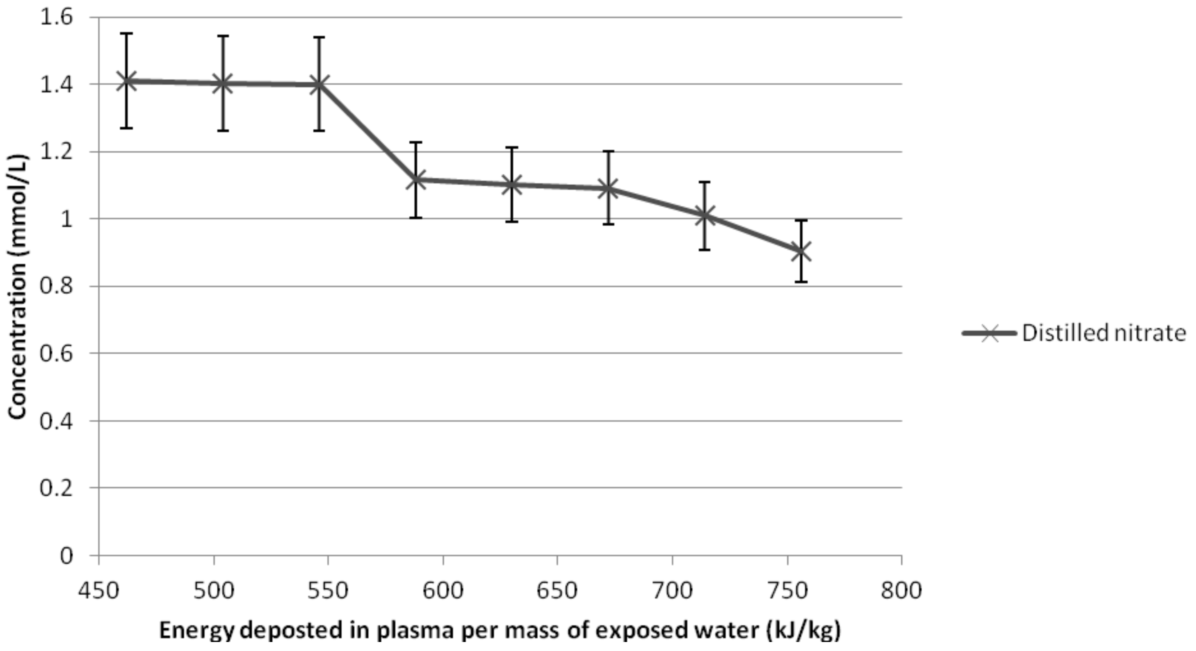


Figure 4.25 Nitrate concentration in distilled water versus energy deposited in the plasma per mass of exposed water. No detectable amount of nitrite generated

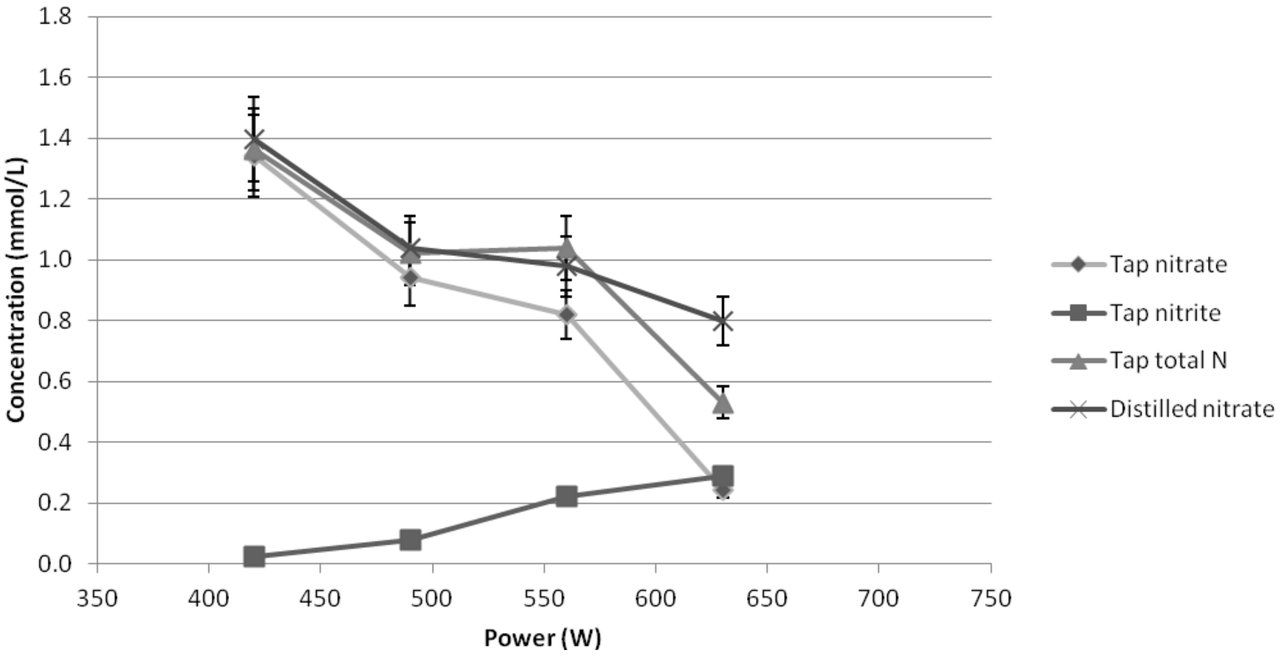


Figure 4.26 Nitrate and nitrite concentrations in tap water versus power. Treatment times scaled such that for each power setting, total energy deposited in system is constant at 504 kJ/kg H2O

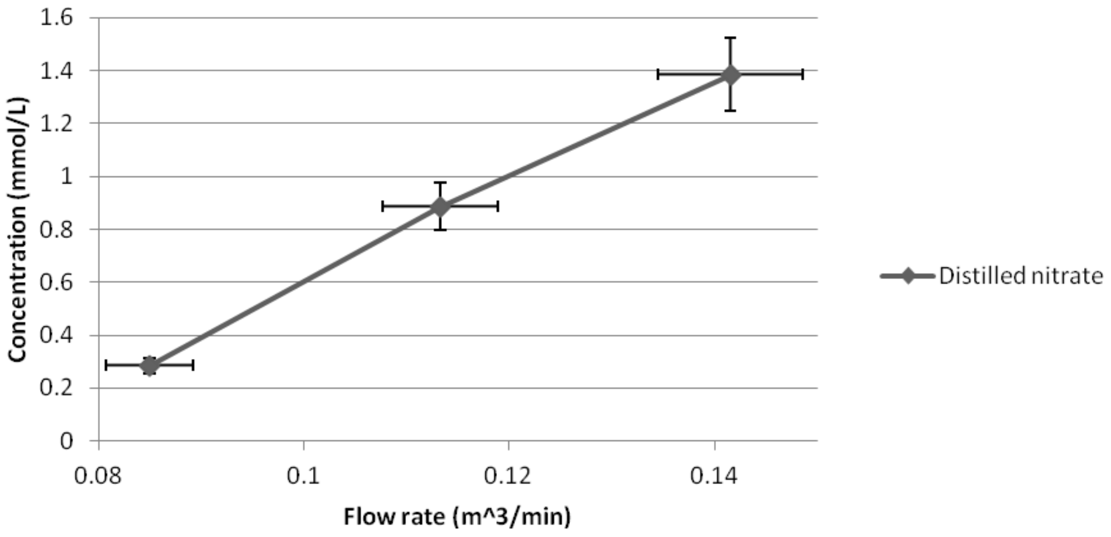


Figure 4.27 Nitrate concentration in distilled water versus air flow rate. No detectable amount of nitrite generated

One variable with remarkable effects on nitrogen species concentration is the presence of basic species before plasma exposure and also addition of basic species after plasma exposure. As mentioned in the experimental section and as will be touched on further in the discussion section, basic species are known to react with dissolved NO and $NO_2$ (which are formed in the plasma) to form nitrite via reaction 5 in table 4.3. table 4.2 summarizes a series of experiments in which the effect of adding approximately 6 mmol/L of sodium bicarbonate before or after plasma exposure was observed on tap and distilled water substrates (200 mL volume). In both distilled and tap water samples, adding sodium bicarbonate before plasma exposure produced significantly more nitrite than when it was added post-exposure, which in turn produced significantly more nitrite than when no bicarbonate was added at all. For all three treatment schemes, tap water ended with more nitrite than distilled. Nitrate trends were not as clear.

Another variable that was manipulated was the time between plasma exposure and post-exposure addition of $NaHCO_3$. Figure 4.28 shows that while the total molar concentration of ionic nitrogen species is a constant, increasing the time between plasma exposure and $NaHCO_3$ addition increases nitrate concentration and decreases nitrite concentration.

A fundamental change in the set-up of the system can be realized by removing the stagnant water volume from underneath the electrode and instead spraying the water substrate directly through the active plasma region as described in the experimental section and as shown in

Table 4.2 Dependence of nitrogen ionic species on water type and amount of $NaHCO_3$ in solution. The sample #'s are used as references in the discussion section. From reaction 5 in table 4.3, the trends observed here for $NaHCO_3$ would be expected to be observed for any conjugate base of a weak acid.

| **Nitrite (mmol/L)** | **Nitrate (mmol/L)** | **pH** | **Description** | **Sample reference #** |
|---|---|---|---|---|
| .041 | 1.09 | 3.18 | Tap, no $NaHCO_3$ addition | 1 |
| 2.43 | .795 | 7.99 | Tap, 5.71 mmol/L $NaHCO_3$ added pre-exposure | 2 |
| .617 | .981 | 7.68 | Tap, 6.19 mmol/L $NaHCO_3$ add post-exposure | 3 |
| .004 | .795 | 2.88 | Distilled, no $NaHCO_3$ addition | 4 |
| .854 | .273 | 8.02 | Distilled, 5.71 mmol/L $NaHCO_3$ added pre-exposure | 5 |
| .235 | 1.37 | 7.55 | Distilled, 6.67 mmol/L $NaHCO_3$ added post-exposure | 6 |

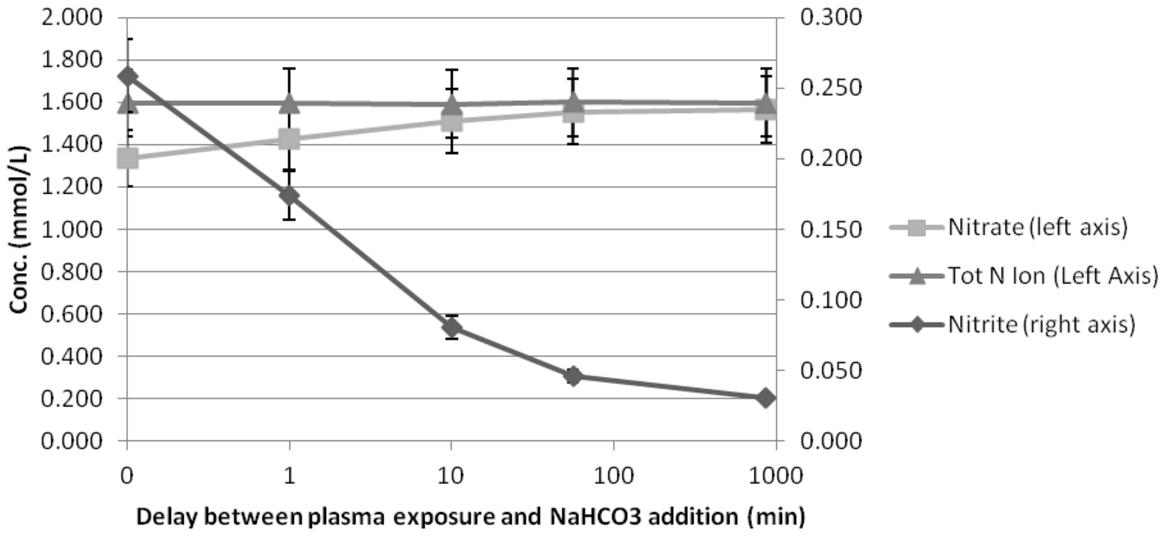


Figure 4.28 Effect of time delay between plasma exposure and bicarbonate addition on nitrite and nitrate species concentrations

fig. 4.2. Some difficulty is experienced in maintaining the plasma during water spray operation. The plasma actively attempts to avoid the region through which the water passes; if the water spray blankets the entire area which the plasma normally occupies, the discharge may extinguish. However, if the plasma is maintained, the increased biphasic interaction is demonstrated by frequent orange light emission from excited sodium in tap water. For this alternative geometry the effects of power and gas flow rate on nitrate uptake are in opposition to the trends witnessed for the stationary water phase geometry. For one pass of distilled water through the active plasma region fig. 4.29 shows increasing nitrate uptake with increasing power for a gas flow rate of .11 $m^3/min$ (no nitrite formed). Instead of power on the x-axis, energy per kg of exposed water is used in order to enable a comparison to the results shown in fig. 4.25. The concentration of nitrate generated in the water is an order of magnitude less in fig. 4.29 than it is in fig. 4.25, but the energy usage per kg of exposed water is also an order of magnitude less. A more obvious comparison between spray and batch treatments can be done by combining figs. 4.25 and 4.29 and plotting the amount of nitrate generated per energy usage as a function of power, as is done in fig. 4.30. The most efficient nitrate generation occurs at 700 W using spray treatment, yielding 4.6 $\mu$mols of nitrate per kJ. However, based on the observed trends, even more efficient nitrate generation may be realized by continuing to increase power with spray treatment or by decreasing power with batch treatment. Figure 4.31 shows that spray treatment efficiency may also be improved by decreasing gas flow rate.

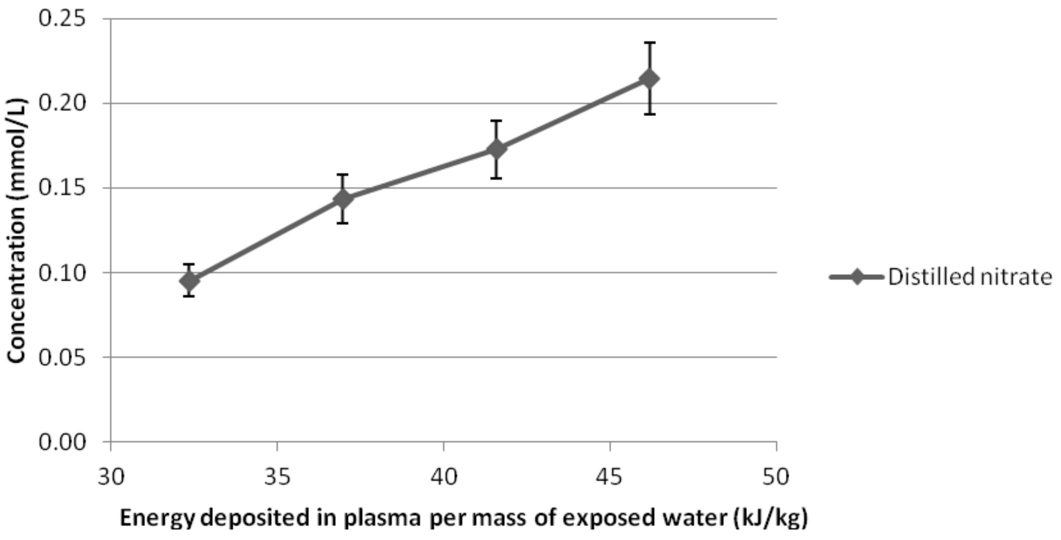


Figure 4.29 Dependence of nitrate uptake on plasma energy deposition for water sprayed through the active plasma region. Compare with results in Figure 17 for batch treatment

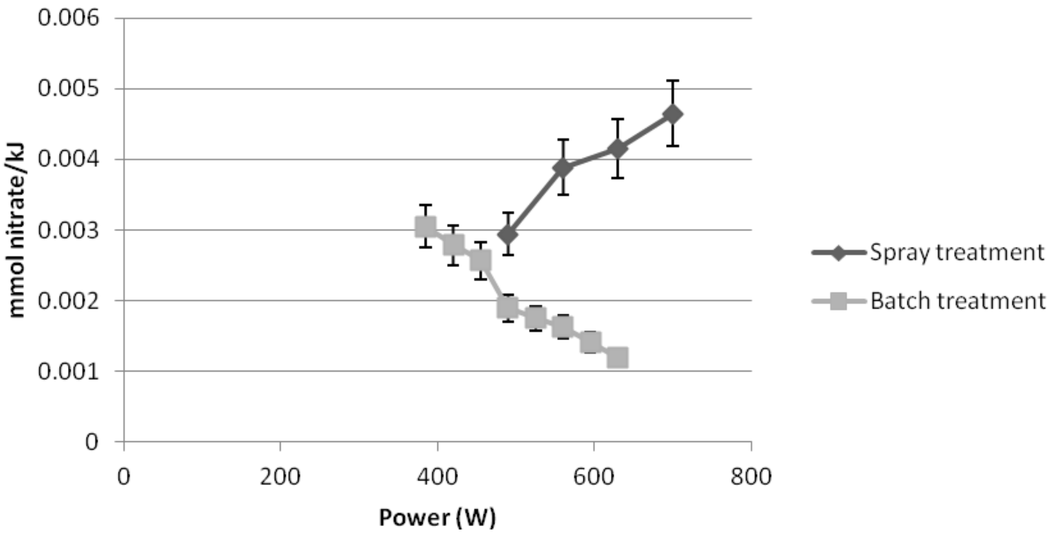


Figure 4.30 Comparison between batch and spray treatment methods using mmol of nitrate generated per kJ of electrical energy as the figure of merit. For lower powers batch treatment is more energetically efficient for nitrate generation. For higher powers spray treatment is more efficient. Further investigation of batch process at lower powers and spray process at higher powers required to determine optimal process for nitrate generation

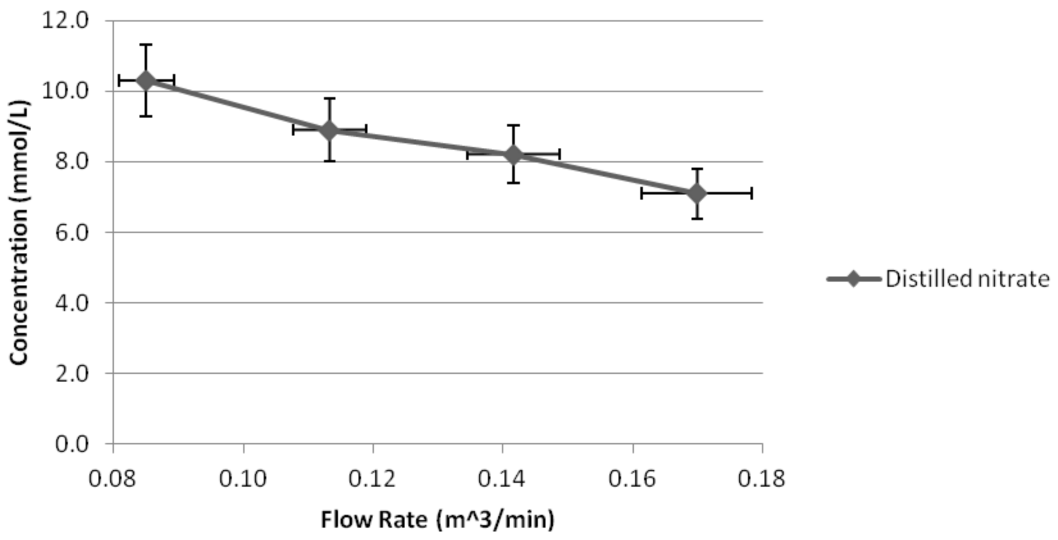


Figure 4.31 Dependence of nitrate uptake on gas flow rate for water sprayed through the active plasma region. Power = 560 W.

The species concentration trends observed in figs. 4.25, 4.26 and 4.29 are believed to result from the dependence of electron density and gas temperature on delivered power and from the dependence of interfacial mass transfer on system configuration. Consider the trend shown in figs. 4.25 and 4.26 for the concentration of nitrate as a function of power and energy deposition for the case where the water surface is held stationary directly under the plasma. As power increases, the amount of nitrate produced decreases. It is possible that the increase in electron density that occurs simultaneously with increasing power creates a more reductive environment which enables more formation of nitrite as evidenced in fig. 4.26 (oxidation state = +3) or other more reduced NOx forms such as NO (+2), $NO_2$ (+4), and $N_2O$ (+1) relative to nitrate (+5). This analysis, however, is confounded by the trend observed in fig. 4.29 in which nitrate uptake increases with increasing power when water is sprayed through the active plasma region. A tentative explanation is that when the water surface is held stationary below the plasma the outgoing convective flow of the feed gas restricts diffusion of water vapor into the plasma region, a restriction that is not present when water is directly injected into the discharge. If water vapor is present in the active plasma region, an increase in power should correspond to an increase in hydroxyl radical formation because of an increase in the rate of electron-impact dissociation. This should increase the oxidizing nature of the plasma and subsequently increase nitrate production; this is observed in fig. 4.29 for direct water injection. While this logic may also extend to the case where the plasma hovers over the stationary water surface, it can be expected to occur to a much more limited degree compared to the direct injection case because the convective wind of the feed gas whisks water vapor away from the active plasma region. Consequently there is not a sufficiently large increase in the oxidizing character of the plasma to offset the increase in reductive character due to electron density; the nitrate concentration then decreases with power as observed in figs. 4.25 and 4.26.

The argument presented in the previous paragraph also supports the trend shown in fig. 4.31, where nitrate concentration decreases with increasing gas flow rate for the case of direct water injection. An increase in gas flow rate decreases the residence time of gas molecules in the glow region, decreasing the gas temperature. Decreasing gas temperature decreases the vaporization rate of liquid droplets, leading to a decreased concentration of hydroxyl radicals in the plasma and a decreased ability to oxidize gaseous nitrogen species to nitrate. A corresponding growth in nitrite as nitrate concentration decreases, which would be predicted by the theory, is not observed in fig. 4.31 like it is in fig. 4.26. The hydroxyl theory contradicts the trend seen in fig. 4.27 for stationary water where nitrate increases with flow rate. One explanation is that the decreased transit time between plasma and water phases results in a decreased radical species recombination rate capable of offsetting the proposed decrease in hydroxyl concentration in the plasma region.

In addition to arguing that increasing hydroxyl concentration in the plasma region should increase the oxidizing nature of the discharge and subsequently increase nitrate concentrations, a stoichiometric outlook suggests that introducing another source of elemental oxygen increases the ratio of oxygen to nitrogen in the discharge, allowing greater formation of high O:N ratio species like $NO_3^-$. This theory could be explored more in future experiments with varying feed ratios of $N_2$ and $O_2$.

In order to address the last variable considered in the study, the effect of basic aqueous species on nitrogen ion concentrations, it is worthwhile to summarize some of the potentially important reaction mechanisms involving reactive nitrogen and oxygen species in solution. Aside from nitrite and nitrate ions, hydrogen peroxide is known to be a prevailing species in solution following plasma treatment [45]; this is confirmed by colorimetric analysis in fig. 4.32. Moreover, volatile $NO_x$ species like NO and $NO_2$ may also be present and may be responsible for the observation in [45] of a spectroscopic peak at 262 nm when samples are sealed; when samples are left unsealed, the 262 nm peak is not observed. Relevant redox reactions involving these species are taken from [133] and [132] and presented in Table 7.

References [132] and [16] illustrate that peroxynitrous acid (ONOOH) is formed as an unstable

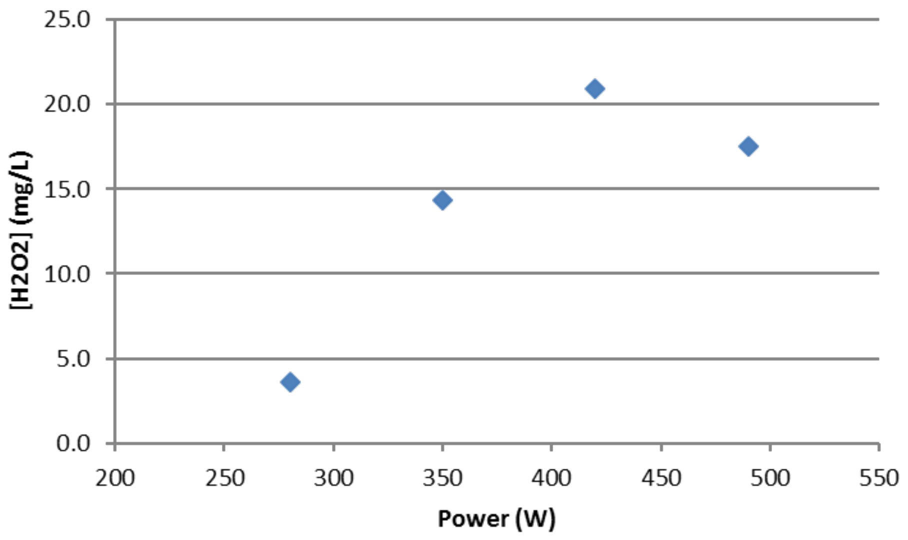


Figure 4.32 Hydrogen peroxide concentration in solution as a function of plasma power

Table 4.3 Important reactions between nitrogen and oxygen species which may occur in the aqueous phase

| **Reaction Description** | **Reaction reference #** |
|---|---|
| $4NO + O_2 + 2H_2O \rightarrow 4H^+ + 4NO_2^-$ | 1 |
| $H_2O_2 + NO_2^- \rightarrow ONOO^- + H_2O$ | 2 |
| $ONOOH \rightarrow H^+ + NO_3^-$ | 3 |
| $3HNO_2 \rightarrow 2NO + NO_3^- + H^+ + H_2O$ | 4 |
| $NO + NO_2 + 2A^- + H_2O \rightarrow 2NO_2^- + 2HA$ | 5 |
| $2NO + O_2 \rightarrow NO_2$ | 6 |
| $3NO_2 + H_2O \rightarrow 2H^+ + 2NO_3^- + NO$ | 7 |
| $4NO_2\,(or\,2N_2O_4) + O_2 + 2H_2O \rightarrow 4HNO_3$ | 8 |

intermediate during the oxidation of acidified aqueous solutions of nitrites to nitrates using $H_2O_2$, and that such solutions are more highly oxidizing than either $H_2O_2$ or $HNO_2$ alone. Because the conditions of the former statement are satisfied in PAW, it is reasonable to assume that peroxynitrous acid is the intermediate species between nitrite and nitrate as hypothesized in [45]. Moreover, the much greater efficacy of PAW compared to a control mixture of nitric acid and hydrogen peroxide for degrading bacteria [134] further suggests the presence of a reactive oxidizing species like peroxynitrous acid.

Applying the equations in table 4.3 to the investigation of basic species effects on nitrogen ion formation provides some insight into observed trends in fig. 4.28, where the nitrite and nitrate molar concentrations in PAW as a function of time delay between plasma exposure and addition of sodium bicarbonate are shown. The +3 oxidation state of nitrogen in water, e.g. nitrite/nitrous acid, is unstable at acidic pH. Following plasma exposure, PAW is acidic and reaction (4) in table 4.3 will occur as long as the solution is acidic. Subsequently, for long time delays between exposure and base addition, the solution has time to convert nearly all aqueous nitrogen species into nitrate. If base is added immediately following exposure more nitrite will be preserved in solution. Moreover, if base is added while +2 and +4 oxidation state nitrogen is present in solution, e.g. species such as NO and $NO_2$, reaction (5) may occur. It is conceivable that reaction (5) is responsible for the nitrite trends witnessed in table 4.2. Tap water contains more basic species than distilled water which theoretically contains none other than a $10^{-7}$ molar concentration of hydroxide. Subsequently, tap water contains more $A^-$ species that are capable of reacting with NO and $NO_2$ to form nitrite. Moreover, if a large quantity of additional base is added to solution before plasma exposure, the amount of A- available for reaction (5) increases significantly, leading perhaps to the comparatively large concentration of nitrite observed in sample 2 in Table 6. This result is not observed to the same degree in the distilled water sample, sample 5, but some effect is still present. Relative to solutions that received no additions, the increased presence of nitrite following post-treatment basic additions could be a combination of both reaction (4) and (5) effects, with (5) occurring when base is added quickly enough that NO and $NO_2$ are still dissolved in solution.

Much of the theory suggested above needs to be validated by further experiment and by computational models. Models should include the relevant chemical reactions shown in [135], [133], and [132] and must be coupled to reactions and mass transfer from the plasma phase. With the construction of the models in chapter 2 and the flexbility of the code in chapter 3, exploration of solution chemistry and the theories presented here are well within reach and are on the agenda for future research.

## 4.7 Summary

Chapter 4 describes our experimental designs for investigating plasma-liquid interactions. Section 4.2 describes the base configuration where the 162 MHz plasma source is pointed downward into a reservoir of water. Section 4.3 examines trying to increase the surface area of plasma-liquid interaction by directly introducing water droplets into the plasma dischare. In section 4.4 we discuss the electrodes placed on the end of the VHF source's inner conductor and their tendency for plasma erosion. To increase fluxes of charged plasma species to the water surface and to alleviate electrode damage, we introduce in section 4.5 an experimental configuration in which the source is pointed upwards and water is pumped through the inner conductor to form a liquid layer on top of the powered electrode. Finally, in section 4.6 we measure different aqueous specie concentrations as a function of different system variables and speculate on the observed trends. The need to extend the models presented in chapter 2 to confirm some of the hypotheses in section 4.6 is noted. Having built and characterized these experimental configurations, it is worthwhile to explore some of the applications of plasma-liquid systems. In chapter 5 we explore a couple of these applications, including fertilization of plants using plasma activated water and degradation of persistent aqueous contaminants.

CHAPTER

# 5 APPLICATIONS OF PLASMA-LIQUID SYSTEMS

Chapter 4 describes the experimental designs used to create and optimize plasma-liquid interactions. The several designs include the base case where the VHF source is pointed straight into a water reservoir, cases where water is sprayed through the plasma using either a green house sprayer or a specially designed nozzle electrode, and the final case where water is pumped through the middle of the VHF source's inner conductor to create a water layer on top of the powered electrode. Along with measuring and performing diagnostics to understand the physical and chemical nature of these systems, we can use these systems in various applications. This chapter explores two such applications. Section 5.1 investigates use of plasma to generate fertilizer and enhance plant growth. The latter half of the chapter studies degradation of aqueous contaminants like 1,4-dioxane and perfluorooctanesulfonic acid (PFOS) with the VHF plasma source.

## 5.1 Fertigation

For a published version of much of the fertigation work described below, the author encourages the interested reader to navigate to [15].

### 5.1.1 Experiment

The glow discharge used to create PAW for plant treatment is generated using the single-stub matched coaxial structure and 162 MHz power source depicted in chapter 4. For detailed design and electrical characteristics, see [41]. Delivered power to the plasma was held constant at 420 W; the air feed gas was flowed at .11 m3/min. To generate a single "batch" of PAW, 1.9 L of distilled water was exposed to the air discharge for 72-80 minutes. The height of the treatment container was controlled such that the discharge was held roughly .5 cm above the water surface for the duration of exposure. Treatment time was chosen such that the final water pH was 2.7. PAW batches were stored at acidic pH for two days and then NaHCO3 was added until a plant friendly pH of 6 was achieved. Final nitrate and nitrite concentrations were determined using ion chromatography (IC), and were between 113-120 ppm and 4-6 ppm respectively. A new batch of PAW was created once every two days in order to keep up with plant watering demand. A representative experimental set-up for exposure of water to the glow discharge can be seen in fig. 4.1.

In a four week fertilizer experiment, Janie marigolds, Better Boy tomatoes, and Early Scarlet radishes were subjected to three different treatment types. A control-control (CC) group was given control water (tap water) for the four-week duration. A control-plasma (CP) group received control water for two weeks and then PAW for weeks 3 and 4; a plasma-plasma (PP) group received PAW throughout. During the germination phase, weeks 1 and 2 of treatment, the plants were arranged as shown in figs. 5.1 to 5.3. Plant potting soil was composed of 60% Canadian sphagnum peat, 20% horticultural grade vermiculite, and 20% horticultural grade perlite; all ingredients were blended together and brought to a moisture content of 50% before potting. A standard greenhouse environment was used with temperatures between 24 and 29 degrees Celsius during the day and between 16 and 21 degrees Celsius at night. Additional experiments not discussed here indicate too much sunlight may negatively affect plant growth irrespective of water treatment type; consequently, shade curtains were used in the presented study to mitigate that effect.

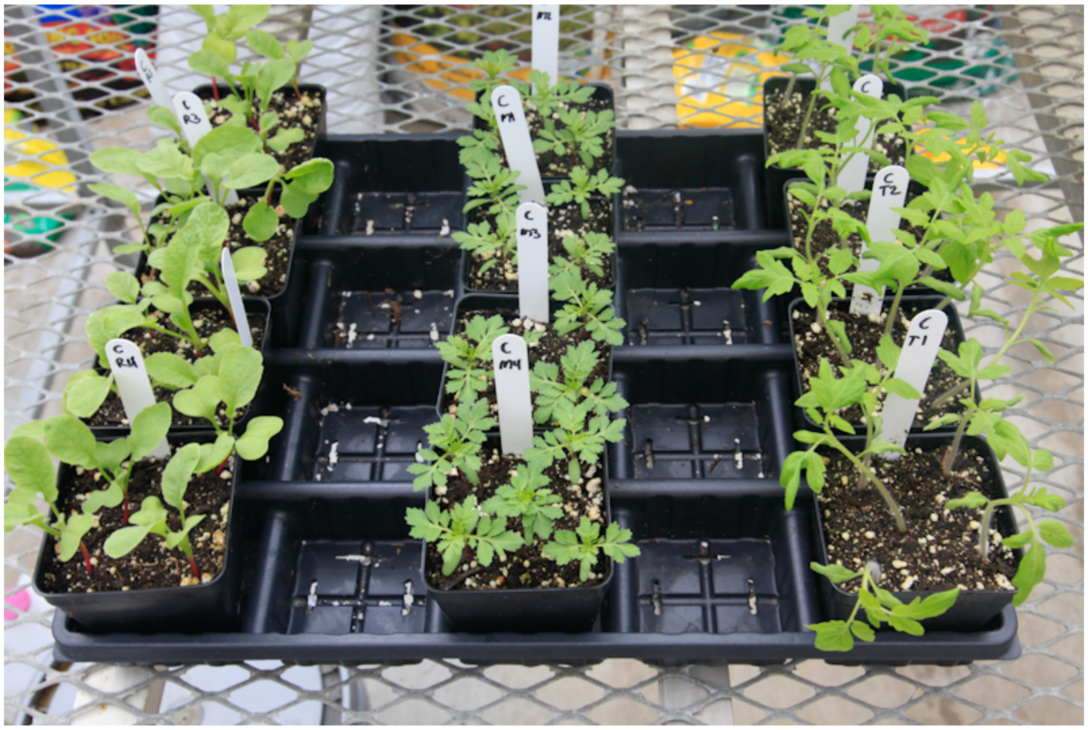


Figure 5.1 CC group potting arrangement during weeks 1 and 2 (germination phase). Photo taken at end of week 2. For scale, each pot is 8.9 cm x 8.9 cm x 6.1 cm (length x width x height)

At the end of the germination phase, a representative plant from each pot was chosen for treatment during the growth phase, weeks 3 and 4. All other plants were removed from the pot. This is exemplified by fig. 5.4.

During the germination phase, plants were misted 4-5 times per day; during the growth phase, plants received a traditional garden-style watering, e.g. steady water stream, 1-2 times per day.

### 5.1.2 Results

As explained in the experimental section, at the end of two weeks, a representative plant from each pot was chosen to continue into the growth phase. At that time the height of the representative plants was recorded; this resulted in a sample size of eight plants for each control strain (radish, marigold, and tomato) and a sample size of four plants for each plasma strain (radish, marigold, and tomato). The control sample size was twice as large as the plasma sample size because both CC and CP groups received tap water through the first two weeks. The

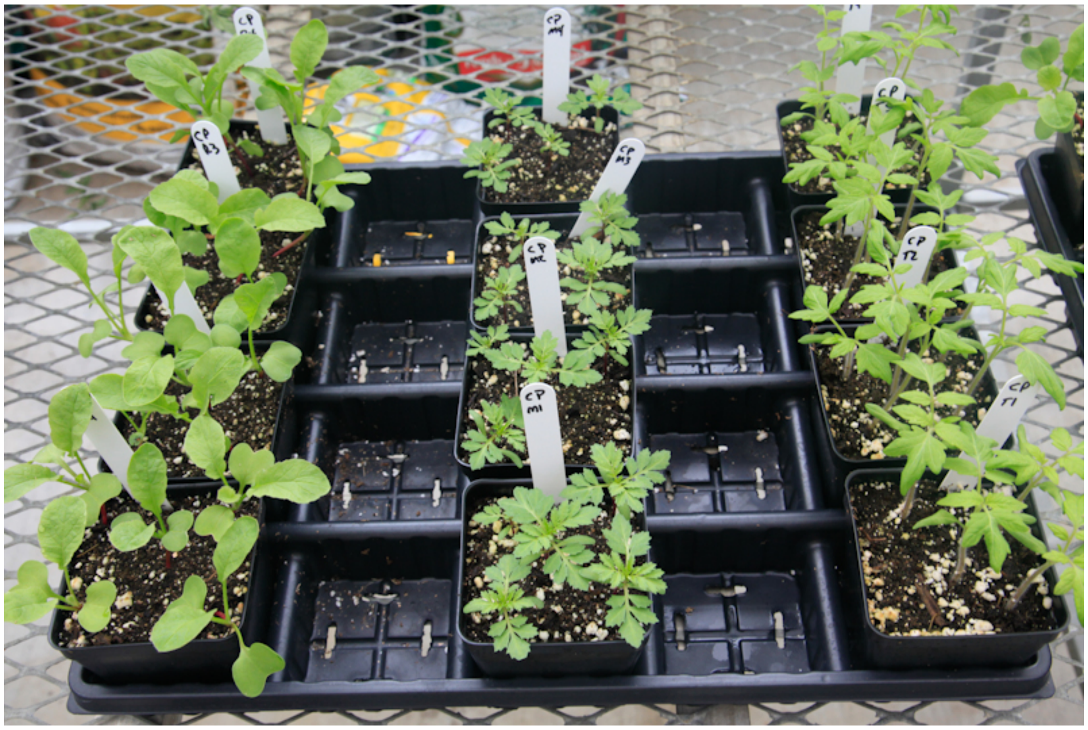


Figure 5.2 CP group potting arrangement during weeks 1 and 2 (germination phase). Photo taken at end of week 2. For scale, each pot is 8.9 cm x 8.9 cm x 6.1 cm (length x width x height)

average height of these plants is shown in fig. 5.5. PAW treated plants showed a larger average height than their control treated counterparts; however, a two-tail Welch's t-test showed that none of the differences were statistically significant for a significance level of .05. The t-test results are summarized in table 5.1. The number of sprouted seedlings per pot was also counted and is presented in fig. 5.6. Though the number of sprouted seedlings per pot was higher for control radishes and tomatoes compared to plasma groups, the differences were not statistically significant as indicated again by a two-tail Welch's t-test with a significance level of .05. The t-test results for the number of sprouted plants per pot are summarized in table 5.2.

Table 5.1 Two-tail Welch's t-test results comparing control and plasma treated plants at end of germination phase. Values shown are p-values

| Radish | Marigold | Tomato |
|---|---|---|
| .054 | .243 | .219 |

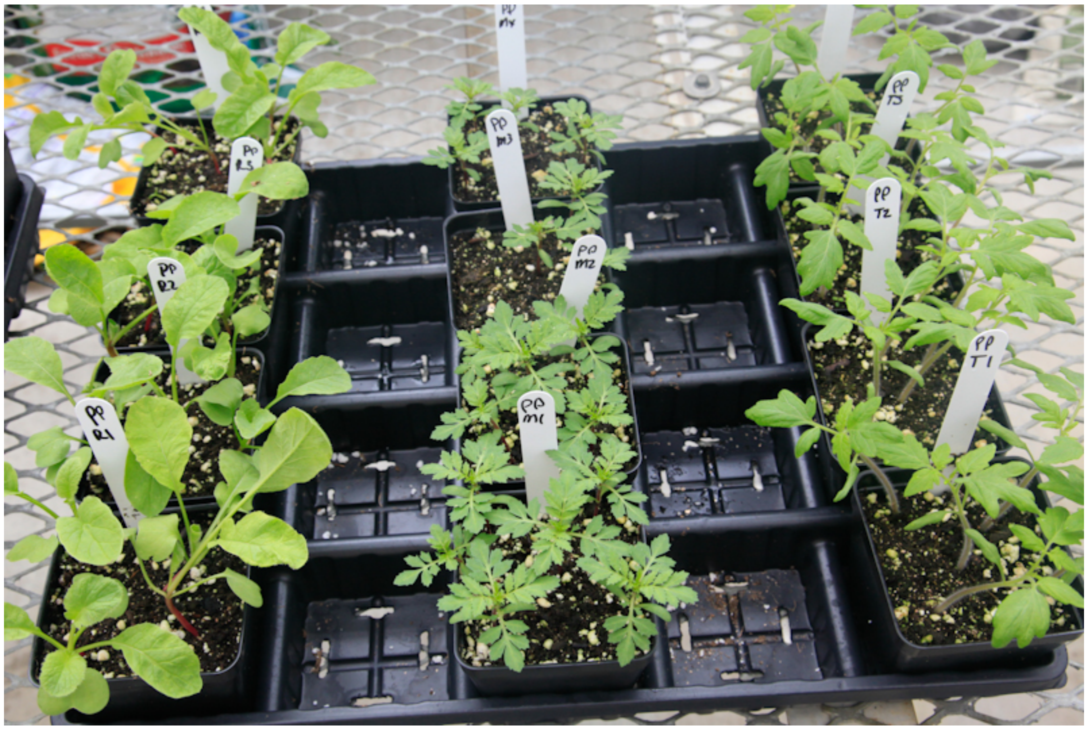


Figure 5.3 PP group potting arrangement during weeks 1 and 2 (germination phase). Photo taken at end of week 2. For scale, each pot is 8.9 cm x 8.9 cm x 6.1 cm (length x width x height)

Table 5.2 Two-tail Welch's t-test results comparing the number of sprouted plants per pot for control and plasma treated plants at end of germination phase. Values shown are p-values

| Radish | Marigold | Tomato |
|---|---|---|
| .163 | 1 | .728 |

Beginning at the start of the growth phase, plant dimensions were measured almost daily. Because of practical difficulties with measuring the height, the distance spanned by the plants' true leaves was recorded. Measurements are plotted in figs. 5.7 to 5.9. Plants receiving PAW during this phase of the experiment, e.g. CP and PP groups, showed a marked improvement in growth relative to the CC group.

In addition to the leaf span measurements recorded throughout the growth phase, photographs of representative plants were taken at the end of experiment in order to visually compare the relative sizes of the CC, CP, and PP groups. These photos are shown in figs. 5.10 to 5.12. CP

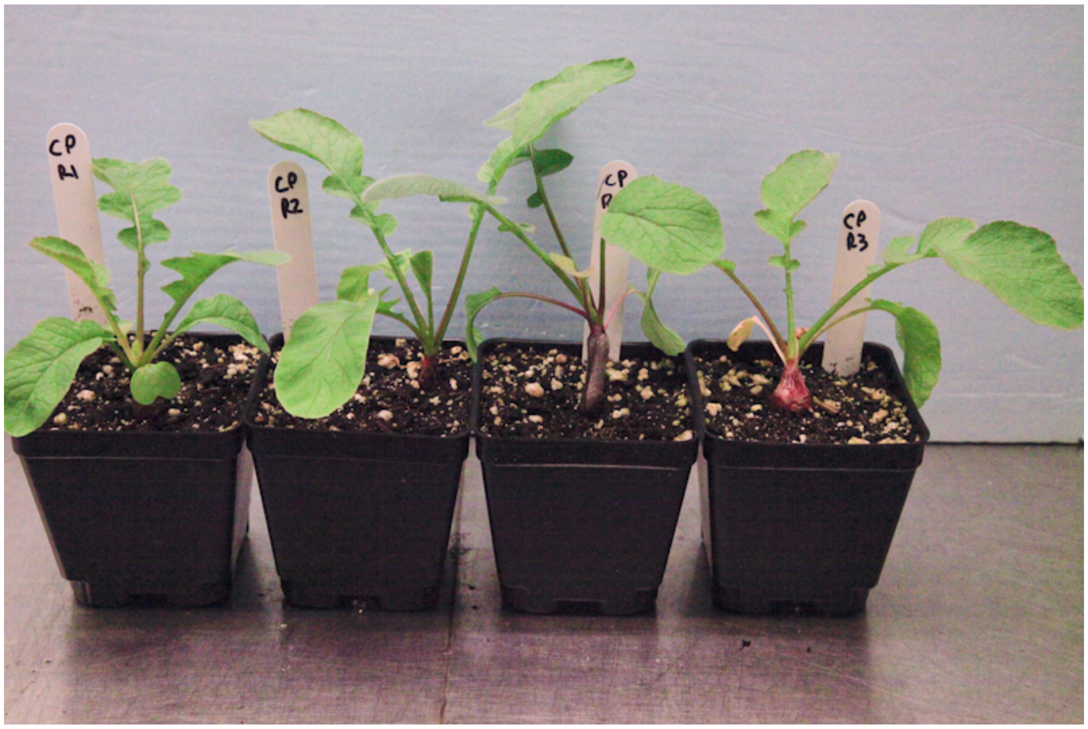


Figure 5.4 Potting arrangement during weeks 3 and 4 (growth phase) for CP group radishes. A single representative plant from each pot was chosen at the end of the germination phase to continue on during the growth phase. For scale, each pot is 8.9 cm x 8.9 cm x 6.1 cm (length x width x height). Note that the 8.9 cm x 8.9 cm dimensions refer to the potâĂŹs top as opposed to its base

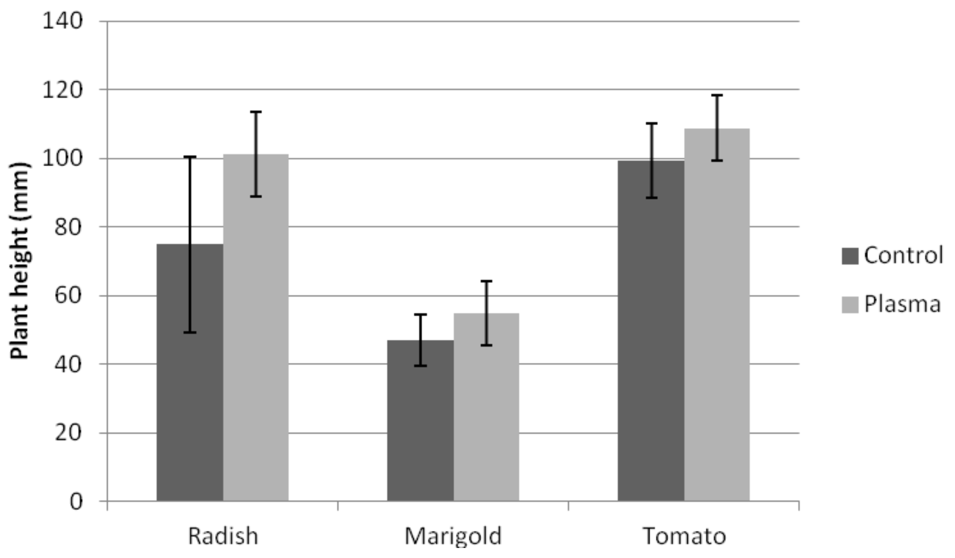


Figure 5.5 Comparison of control and plasma treated plant heights at end of germination phase (end of week 2) with accompanying error bars

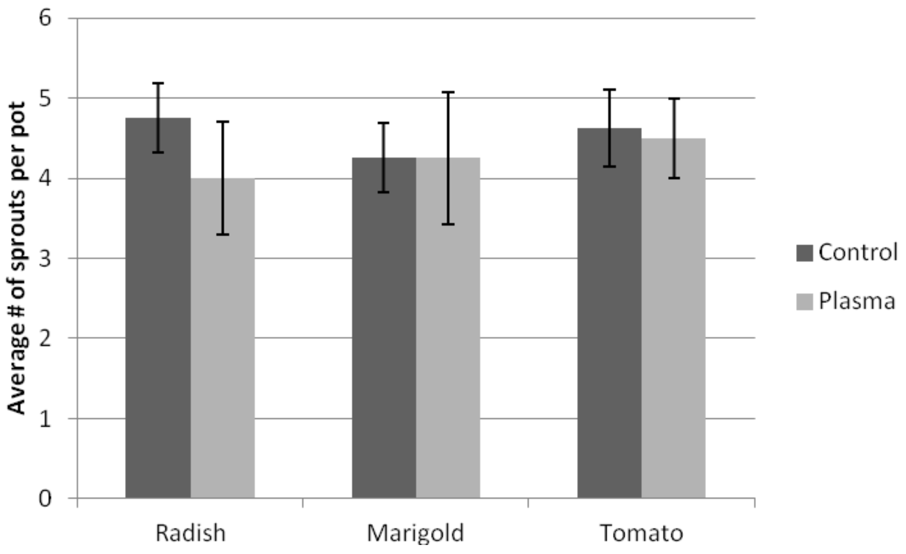


Figure 5.6 Comparison of control and plasma treated sprout data at end of germination phase (end of week 2) with accompanying error bars

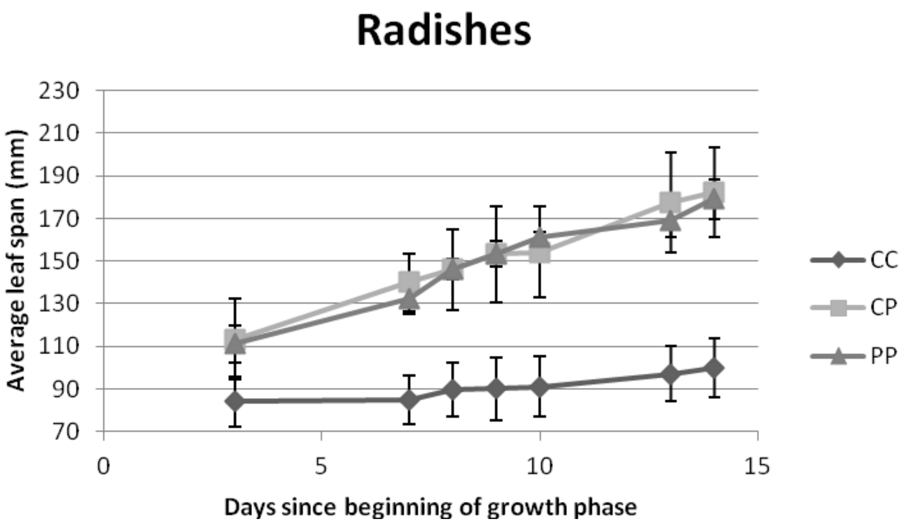


Figure 5.7 Average radish leaf span vs. time (growth phase, weeks 3 & 4)

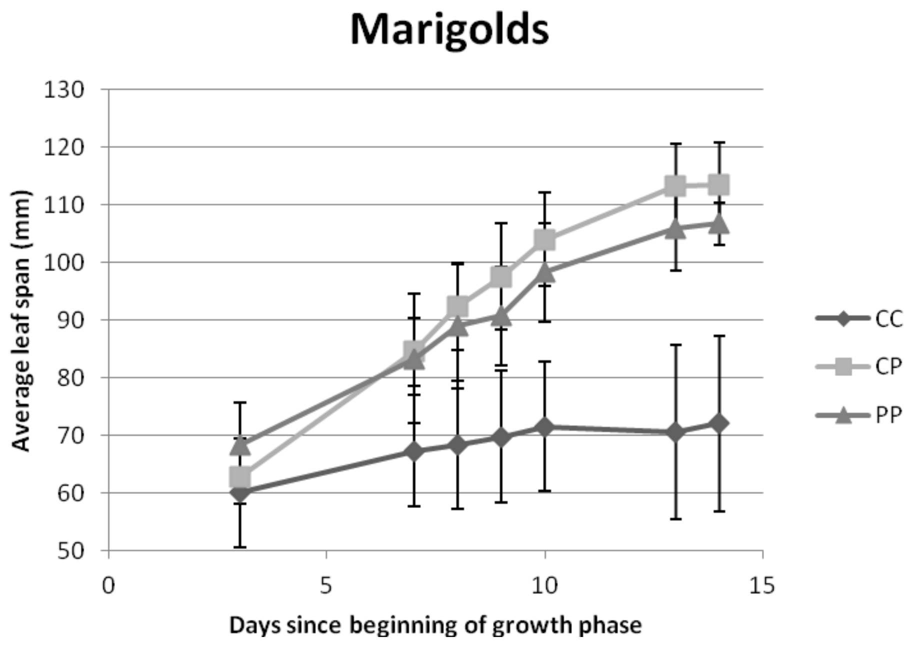


Figure 5.8 Average marigold leaf span vs. time (growth phase, weeks 3 & 4)

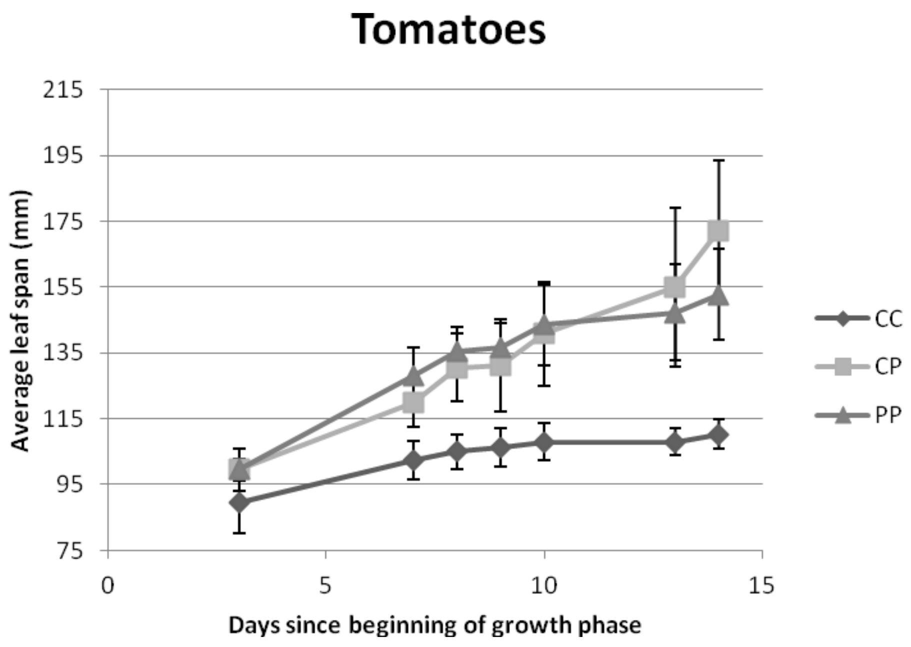


Figure 5.9 Average tomato leaf span vs. time (growth phase, weeks 3 & 4)

and PP plants were larger in size than their CC counterparts.

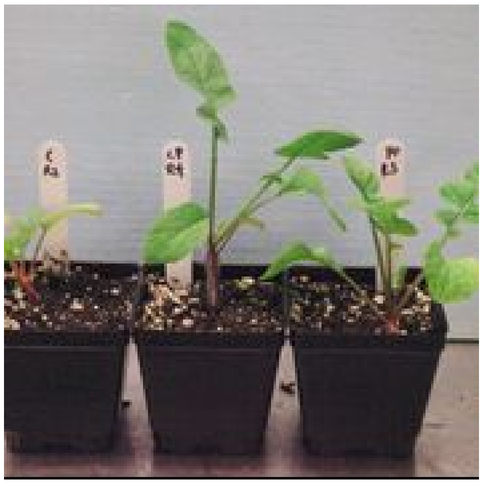


Figure 5.10 Representative radish plants at end of experiment. Left pot is CC; center is CP; right is PP

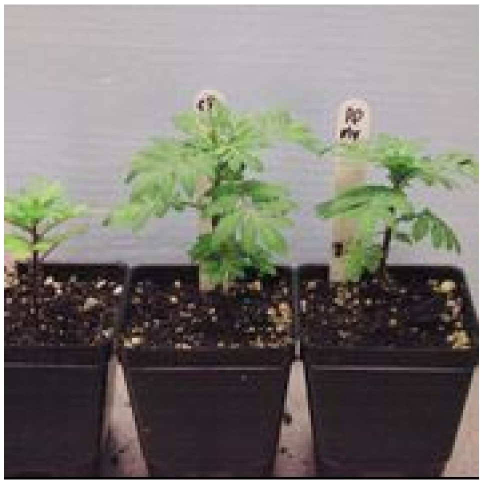


Figure 5.11 Representative marigold plants at end of experiment. Left pot is CC; center is CP; right is PP

After the above photos were taken, plants were removed from their pots, washed, and dried. Roots were separated from the above-ground plant called the shoot and both sections were

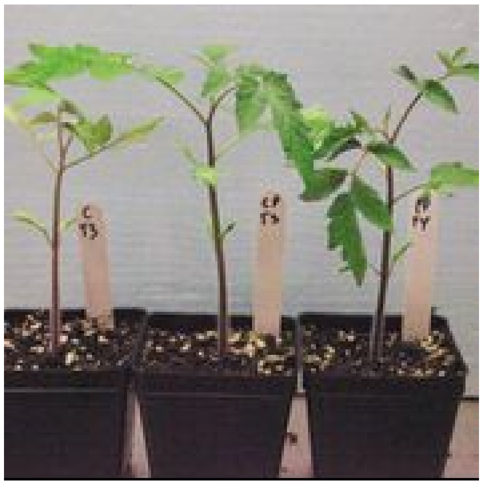


Figure 5.12 Representative tomato plants at end of experiment. Left pot is CC; center is CP; right is PP

weighed. Average shoot and root dry weights are summarized in figs. 5.13 and 5.14 respectively. In agreement with figs. 5.10 to 5.12, the average shoot masses of CP and PP plants were larger than CC plants. A t-test, summarized in table 5.3, showed that all of these differences were statistically significant except for the difference between PP and CC marigolds (however, its test statistic of .06 was very close to our significance cut-off of .05). In marigolds and tomatoes CP shoot masses were greater than PP shoots, however, the differences were within the error of the measurement. Root mass results did not track with the shoot sizes and masses. The root masses of CC radishes were on average larger than CP and PP radishes. CP marigold and tomato root masses were greater than their PP counterparts which were in turn larger than CC root masses. However, all of the root mass differences were within the error of the measurement, and the t-test summarized in table 5.4 indicates that the differences are not statistically significant.

Table 5.3 p-values for comparisons between the shoot masses of different plans and treatment groups. Values below .05 indicate a statistically significant difference between the species being compared

| Shoot Mass | PP vs. CP | PP vs. CC | CP vs. CC |
|---|---|---|---|
| Radish | 1.000 | 0.001 | 0.005 |
| Marigold | 0.224 | 0.060 | 0.017 |
| Tomato | 0.414 | 0.035 | 0.044 |

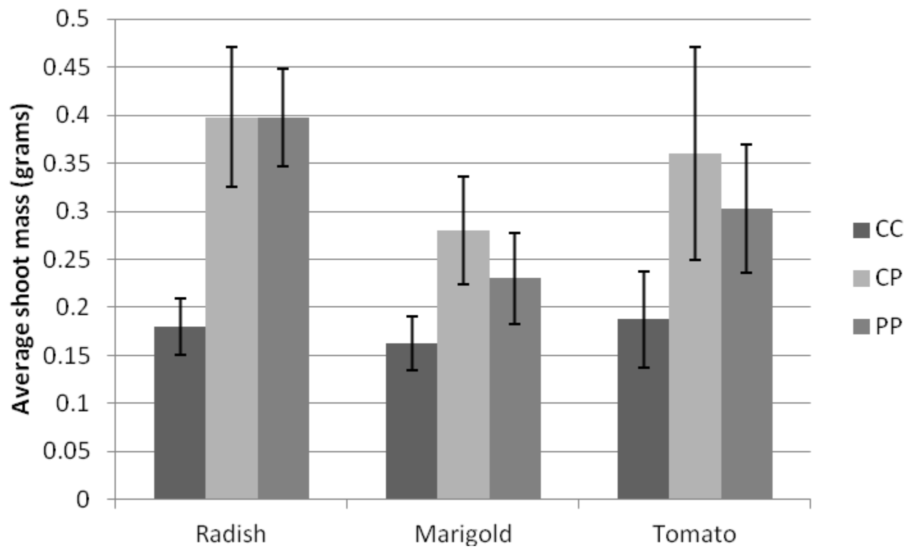


Figure 5.13 Average shoot dry mass by plant and treatment types at end of experiment

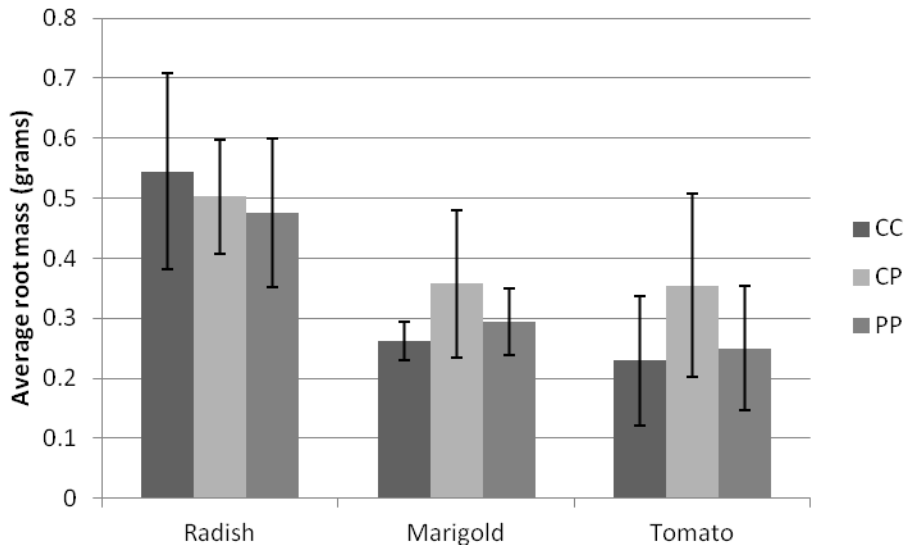


Figure 5.14 Average root dry mass by plant and treatment types at end of experiment

Table 5.4 p-values for comparisons between the root masses of different plans and treatment groups. Values below .05 indicate a statistically significant difference between the species being compared

| Root Mass | PP vs. CP | PP vs. CC | CP vs. CC |
|---|---|---|---|
| Radish | 0.738 | 0.523 | 0.674 |
| Marigold | 0.402 | 0.360 | 0.218 |
| Tomato | 0.304 | 0.798 | 0.235 |

### 5.1.3 Discussion

Over the course of a four week experiment plants which received PAW in weeks 3 and 4 (CP group) and plants which received PAW for all four weeks (PP group) grew significantly larger than tap water controls (CC group). Differences between PAW and control groups did not emerge immediately. As shown in fig. 5.5 and by the statistical analysis in table 5.1, control and plasma treated plants were not significantly different in size after two weeks. It should also be noted from fig. 5.6 and table 5.2 that PAW and control groups did not show significant differences in germination rates. However, during the growth phase, weeks 3 and 4, differences between PAW treated plants and tap water controls became evident. In figs. 5.7 to 5.9 the increased growth rate of CP and PP plants relative to CC is evident in the sizable slope differences. The side-to-side photographs in figs. 5.10 to 5.12 show the greater height and foliage of CP and PP plants compared to their CC counterpart. Moreover, although it is difficult to note in the photographs, CP and PP plants had a healthy, green color at the end of the 4-week experiment; CC plants had begun to yellow and wither. Figures 5.13 and 5.14 compare the root and shoot masses for different treatment groups. Tables 5.3 and 5.4 show p-values indicating the level of difference in shoot and root masses between groups. Smaller p-values indicate larger statistical differences; a value of .05 has been chosen as the threshold level to indicate significant statistical differences. Using that significance level, it is found that CP plants all had significantly larger shoot masses than tap water controls. PP radishes and tomatoes were significantly larger than controls; PP marigolds were not significantly different from control marigolds (although the p-value of .06 is close to the threshold for significance). There were not any significant differences between CP and PP shoot masses. The differences found in shoot masses were not reflected in the root mass data. Table 5.4 shows that no groups demonstrated significant differences in root mass. The general increase in shoot mass of PAW treated plants is believed to occur primarily because of the nitrate present in PAW after plasma exposure. Nitrogen is well known to be an essential plant nutrient, necessary for proteins, enzymes, and metabolic processes; ion chromatograph and Total N analyses reveal that nitrite and nitrate are the long-lived nitrogen species in PAW. Moreover, nitrate concentrations are a factor of 20 larger than nitrite concentrations in these plant experiments and so should be the dominant nitrogen specie that the plants are exposed to.

## 5.2 Remediation of Aqueous Pollutants

### 5.2.1 Overview of Current Techniques

As discussed in chapter 1, plasmas in contact with liquids produce a cornucopia of reactive species, including both highly oxidative species like OH and highly reductive species like $e^-$. There is considerable interest in the low-temperature plasma community in using these highly reactive species to degrade persistent chemicals in waste streams, both gaseous and liquid. Below, we explore using the VHF atmospheric source to degrade dioxane, a known carcinogen, and perfluorooctanesulfonic acid (PFOS), a perfluorinated compound (PFC) that has been associated with increased risk of chronic kidney disease. [136] As discussed in section 1.2.3.1, current research into dioxane degradation focuses on Advanced Oxidative Techniques (AOT) that ultimately produce hydroxyl radicals which attack and cleave contaminant bonds. A common AOT is combination of $H_2O_2$ and UV light. Another that receives significant attention is $TiO_2$ and UV. A problem with both of these techniques is the requirement of external chemicals. As shown in the following section, plasmas are capable of generating OH in aqueous solution without the need for chemical additives.

Techniques that have shown promise for aqueous PFC degradation include photolysis combined with persulfate [67], reduction with iron under high pressure conditions [68], and pyrolysis brought about through acoustic cavitation. [69, 64]. Despite their promise, there are some drawbacks to these techniques. While photochemical techniques have shown efficacy against perflourinated carboxylic acids like perfluorooctanoic acid (PFOA), they have been unsuccessful in degrading PFOS. [68] Zerovalent iron under high pressure conditions has proven effective; however, the reaction takes place in a stainless steel container under 226 atmospheres of pressure. [68] Sonolysis is perhaps the most promising with demonstrated efficacy against both perfluorinated carboxylic acids (PFOA) and sulfonates (PFOS) and without the need for high pressure equipment. Sonication requires around 250 Watts per liter of treated solution [64], which is comparable to the atmospheric plasma techniques discussed below.

### 5.2.2 Plasma Treatment of Dioxane

In an experiment to test the efficacy of plasma for treating dioxane, a 500 mL solution of 400 $\mu$g/L dioxane was treated for 26 minutes with a 420 W air discharge using the geometric configuration shown in fig. 4.1. The time profile of dioxane concentration is shown in fig. 5.17; it follows a simple exponential decay shape. After 26 minutes, 98% of the dioxane has been removed. This compares very favorably to an AOT study in the literature ([55]). In the literature study, the treated solution was a factor of two larger; however, the concentration of dioxane was two to three orders of magnitude higher. The plasma decadal treatment time was slightly shorter than the literature study's.

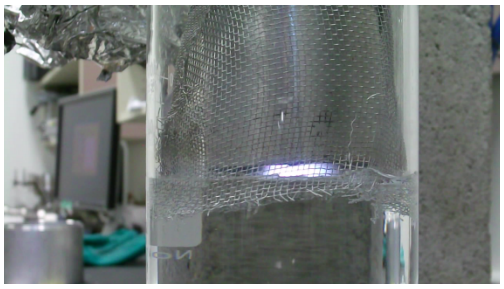


Figure 5.15 Photograph of 420 W "calm" air discharge treating aqueous dioxane solution.

A 350 W argon discharge was also used in the dioxane treatment study. Even at a lower power relative to the air discharge, no dioxane was detected after 5 minutes of treatment. This result can be seen in fig. 5.17. The near order of magnitude better performance of argon over air discharge is interesting. Some feeling of the fundamental difference in performance can be gleaned by looking at the appearance of the discharges. The calm air discharge can be seen in fig. 5.15; the much brighter and much larger surface area argon discharge can be seen in fig. 5.16. The bright blue color in the argon discharge is likely due to plasma interactions with the aluminum electrode. It is conceivable that the plasma-metal interactions create a larger density of electrons in the discharge that in turn lead to greater creation of oxidative species like OH from water vapor and/or the ambient air. Or it is possible that the greater density in gas phase electrons translates into a greater density of hydrated electrons and that degradation of dioxane may proceed through a reductive pathway as opposed to an oxidative one. Such uncertainty in reaction mechanisms is one of the fundamental reasons that the modelling work described in chapter 2 was begun; until experimental diagnostics are developed that are capable of detailed and comprehensive probing of both gas and near-inteface liquid chemistry, models are a good way to qualitatively explore the plasma-liquid dynamics.

Despite its rapid success in treating dioxane, there are several drawbacks to using the argon

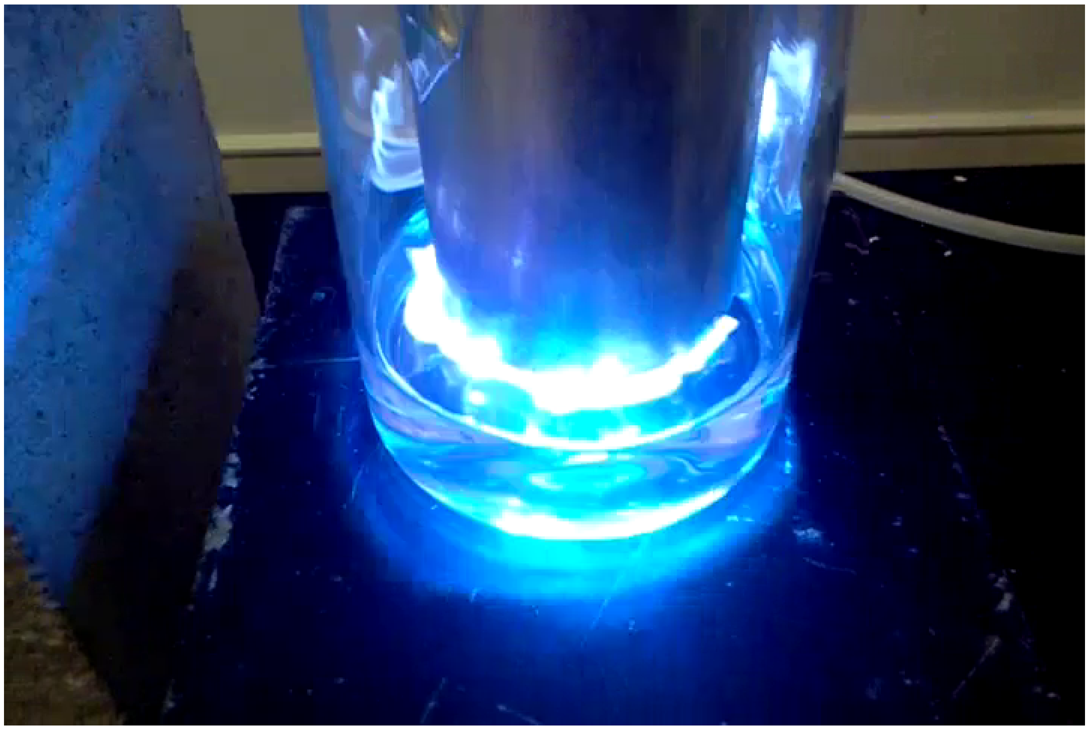


Figure 5.16 Photograph of 350 W “bright” argon discharge treating aqueous dioxane solution

discharge. One is that argon is much more expensive than air; for treatment plant wastewater scales, the cost is likely to be prohibitive. Another problem is the erosion of the metal electrode; this could be alleviated by using the pure water electrode configuration (see figs. 4.12 and 4.13). However, removal of the argon-metal interaction could very well decrease the efficacy of the argon-dioxane treatment. A final problem with the argon discharge is its transient nature. When operating the argon discharge, the load impedance can oscillate wildly. This makes impedance matching very difficult, leading to lots of reflected power to the generator. Despite these issues, the argon treatment result is intriguing because of its considerably greater efficacy in terms of log reduction time when compared to leading AOTs like $H_2O_2/O_3$. Moreover, the air-dioxane treatment, which lacks the issues associated with the argon treatment, also compares reasonably well to the $H_2O_2/O_3$ method in terms of log reduction time. Finally, as illustrated by the PFOS results presented in section 5.2.3, the fig. 4.1 configuration is unlikely to be the best scheme for maximizing plasma-liquid interactions and destroying persistent chemicals. The configuration presented in fig. 4.12 is likely a better choice; indeed a preliminary experiment showed a one-pass reduction in the concentration of dioxane from 365 $\mu$g/L to 172 $\mu$g/L using the pure water electrode design. It should be noted that from an industrial application standpoint, the figure of merit is likely the amount of energy required for a log reduction in pollutant concentration as opposed to the time required for log reduction. Using the energetic figure of merit, the current atmospheric source design will compare less favorably with purely chemical treatments like $H_2O_2/O_3$. However, using a pulsed source design, the energetic cost of the plasma source may be drastically reduced without significantly reducing the flux of reactive species to the aqueous phase.

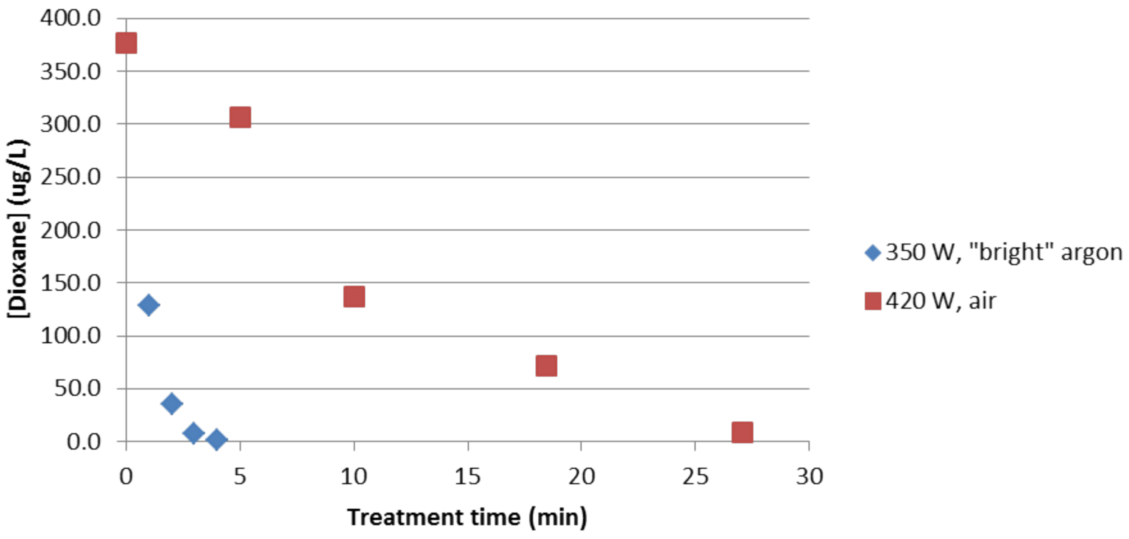


Figure 5.17 Comparison of argon and air discharges for removing dioxane from solution

### 5.2.3 Plasma Treatment of PFOS

When PFOS solution is treated using the geometric configuration shown in fig. 4.1, no degradation is observed. However, when PFOS solution is treated using the pure water electrode geometry of fig. 4.12, degradation is observed. Experimental conditions are 700 W delivered to the plasma, 3 standard cubic meet per minute of air flow, and a roughly 50 $\mu$g/L starting concentration of PFOS. PFOS concentrations are measured using a high performance liquid chromatography (HPLC) instrument provided by the Environmental Protection Agency. The time vs. PFOS concentration profile is shown in fig. 5.18. The curve shows an exponential decay shape that appears to level around a concentration of 5 $\mu$g/L. Ninety percent reduction in PFOS concentration is considered a significant success by colleagues at the EPA. Additional work has focused on trying to elucidate the PFOS degradation mechanism by examining degradation products; however, measurements with HPLC/TOF-MS, ion chromatography, and other methods have been inconclusive. Thus, it is unknown whether PFOS breaks down via oxidative or reductive routes. That the batch treatment scheme is totally unsuccessful in achieving degradation suggests a reductive route since the OH radical concentrations are not expected to vary significantly between fig. 4.1 and fig. 4.12 while the charged particle fluxes, including electrons, are expected to be much larger in the latter case. However, new experimental techniques or detailed models will be required to confirm that hypothesis. The small gap between the powered electrode and the water substrate in the batch configuration makes absorption spectroscopy like that performed for the water electrode geometry (described in section 4.5.3) difficult; otherwise, a direct comparison of OH densities could be made. This is again motivation for extending the modelling work begun in chapters 2 and 3.

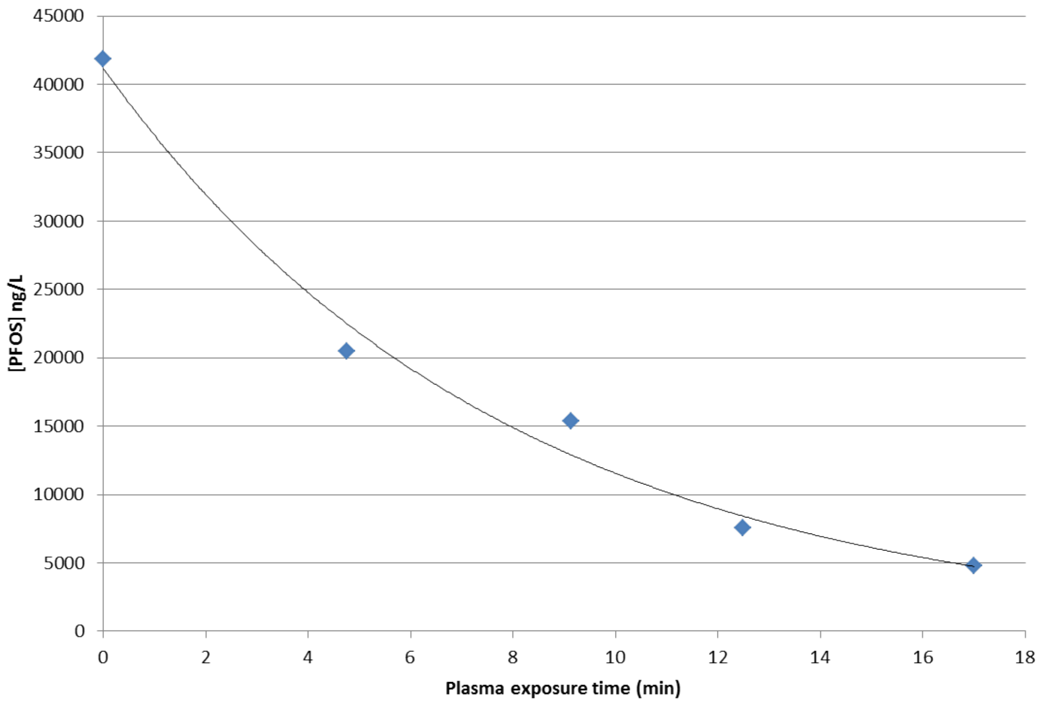


Figure 5.18 PFOS concentration vs. treatment time using the water electrode

## 5.3 Summary

This chapter investigates a couple applications of plasma-liquids. Section 5.1 explores fertilization of radishes, tomatoes, and marigolds indirectly through plasmas. PAW treated plants are shown to grow 1.7-2.2 times larger than their tap-watered peers because of the aqueous $NO_x$ species dissolved into solution by the plasma. Section 5.2 describes remediation of aqueous contaminants using the VHF source. The batch configuration using air as the feed gas is shown to be effective at removing 1,4-dioxane. However, replacing air with argon leads to an order of magnitude improvement in dioxane removal. While the batch configuration is unable to degrade PFOS, the water electrode geometry is capable of removing the contaminant. Improvement of pollutant removal rates will rely on increased understanding of the reaction mechanism and dissolution of reactive plasma species. This is a logical extension of the modelling work in chapters 2 and 3 and experimental characterization in chapter 4. Concluding thoughts on intertwining modeling and experimental efforts in the broader scope of plasma-liquid science are contained in the final chapter of the dissertation, chapter 6.

CHAPTER

# 6
# CONCLUSION & FUTURE WORK

## 6.1 Work to Date

This dissertation includes both modelling and experimental studies of plasma-liquid systems. The keystones of the plasma-liquid research are the modelling chapters, chapters 2 and 3. Chapter 2 addresses a couple of fundamental questions of the plasma-liquid research community. Section 2.1 considers the role of convective fluid flow on transport processes in plasma-liquid systems. To our knowledge, this is the first comprehensive model of transport processes in convective atmospheric plasma systems; previous studies have focused on diffusive systems. [1] Through evaporative cooling, convection creates a significant difference in bulk temperatures (roughly 10 K) between gas and liquid phases. Convection also significantly increases mass transfer rates from gas to liquid for hydrophobic species like NO. Additionally, penetration of reactive species like OH and surface ONOOH formed from reaction of OH and $NO_2$ into solution is limited to a few tens of microns. This suggests that reactivity at a substrate covered by an aqueous layer is likely due to longer lived species like $NO_2^-$ and $H_2O_2$ capable of reacting to create OH and $NO_2$ radicals through a peroxynitrous acid intermediate.

Section 2.2 describes a fully coupled discharge-liquid model in which the effect of varying interfacial parameters like the electron surface loss coefficient is considered. Previous works [1, 28] have a ssumed surface loss coefficients of unity; however, there is no current experimental support for this assumption. We believe our work is the first to explore this uncertainty in electron transport at the interface. Moreover, this is the first published application of the burgeoning multiphysics code, MOOSE, to problems in plasma physics. It is found that over a range of interfacial parameters, the near-interface gas electron density can vary by four orders of magnitude. The interfacial electron energy is similarly a sensitive function of loss coefficients. This motivates finer scale computational efforts like Molecular Dynamics simulations or novel experimental techniques to probe the near-interface plasma dynamics that are capable of accurately determining the interfacial cofficients required for fluid modelling.

In chapter 3, we present the code Zapdos we developed for simulating plasma discharges in contact with liquids. Included in the chapter are descriptions of the kernels used to re-create plasma-liquid governing equations, auxiliary kernels for computing important system variables, materials used to describe features of the gas and liquid phases, boundary conditions describing the interactions of our domains with the environment, and interfacial kernels for connecting the physics in the plasma and liquid domains. We also describe the changes made to the MOOSE framework allowing coupling of the gas and liquid phases. Where other atmospheric plasma codes have relied on segregated methods, the combination of Zapdos and MOOSE allows full coupling for the first time of the equations describing both plasma and liquid dynamics. Studies in other computational science fields have demonstrated that fully coupled methods illustrate more robust convergence than segregated methods. [26]

Experimental optimization and characterization of experimental plasma-liquid systems are the subjects of chapter 4. Several configurations based around our 162 MHz VHF source are described. The most successful of these points the VHF source upward; water is then pumped through the center of the inner conductor and an approximately milimeter thick layer of water is formed on top of the powered electrode. As far as we know, this is the first example of a glow discharge in contact with a powered water surface. Because the glow does not contact any powered metal, the system can be operated at much higher powers than normal, up to 1155 W. Additionally, OH is an abundant plasma specie with densities up to $5 \times 10^{21} m^{-3}$ as indicated by broadband absorption spectroscopy. The final section of chapter 4 explores the aqueous chemistry generated by contact with the plasma. The chemistry depends sensitively on the pre-existing solution chemistry; it shows a weaker dependence on parameters like power and gas flow rate. $NO_3^-$ and $H_2O_2$ are produced ubiquitously.

Chapter 5 explores a couple of applications of plasma-liquids. In a unique, collaborative experiment with the horticulture department, plasma treated water was used to water tomatoes, marigolds, and radishes, and its performance was compared to tap-water controls. The plasma water treated plants grew significantly larger than controls because of the >100 ppm nitrate concentration generated in solution by the plasma. The VHF source was also shown to be effective at remediating 1,4-dioxane and perfluorooctanesulfonate, to our understanding the first time degradation of these contaminants has been demonstrated with plasma.

## 6.2 Future Work

There is potential for numerous fascinating projects stemming from the work presented in this dissertation. First on the agenda is extending the model in section 2.2 to multiple dimensions. Using a 2D axisymmetric model, we hope to reproduce the experimentally observed behavior shown in fig. 6.1: increased spreading of a DC discharge over a water surface as solution conductivity is decreased. Figure 6.2 shows a preliminary simulated plot of the electron density before the plasma is able to propagate from the needle to the water. Further work will complete the development of the discharge and evolution to a steady-state. Once the discharge model is vetted in multiple dimensions, it can be combined with the model in section 2.1 to provide a comprehensive description of the physiochemical phenomena present for point-to-plane DC discharges. Addition to Zapdos of the momentum and heat transport kernels needed for model coupling should completable within a day. The more challenging aspect will be defining consistent and numerically feasible boundary conditions for the potential in multiple dimensions.

We also plan to add electromagnetic field modelling capabilities to Zapdos. With EM capability we can begin to model the experimental systems described in chapter 4. The VHF plasma-liquid system presents a unique blend of physical and chemical phenomena, including EM fields, charged and neutral particle transport, fluid flow, heat transport, and gas and liquid chemistry. Zapdos is ideally suited for combining such rich dynamics. If such a comprehensive model of the VHF plasma-liquid system can be built, then it can be compared to some of the measurements made in section 4.6. If the model compares favorably to the measurements, then the model can be used to elucidate the mechanisms that lead to long-lived aqueous species like $H_2O_2$, $NO_2^-$, and $NO_3^-$. This could be beneficial for determining optimal plasma parameters for production of fertigation water. Additionally, reaction kinetics for dioxane and PFOS could be added to the model in an attempt to describe the behavior seen in section 5.2 and optimize the system for

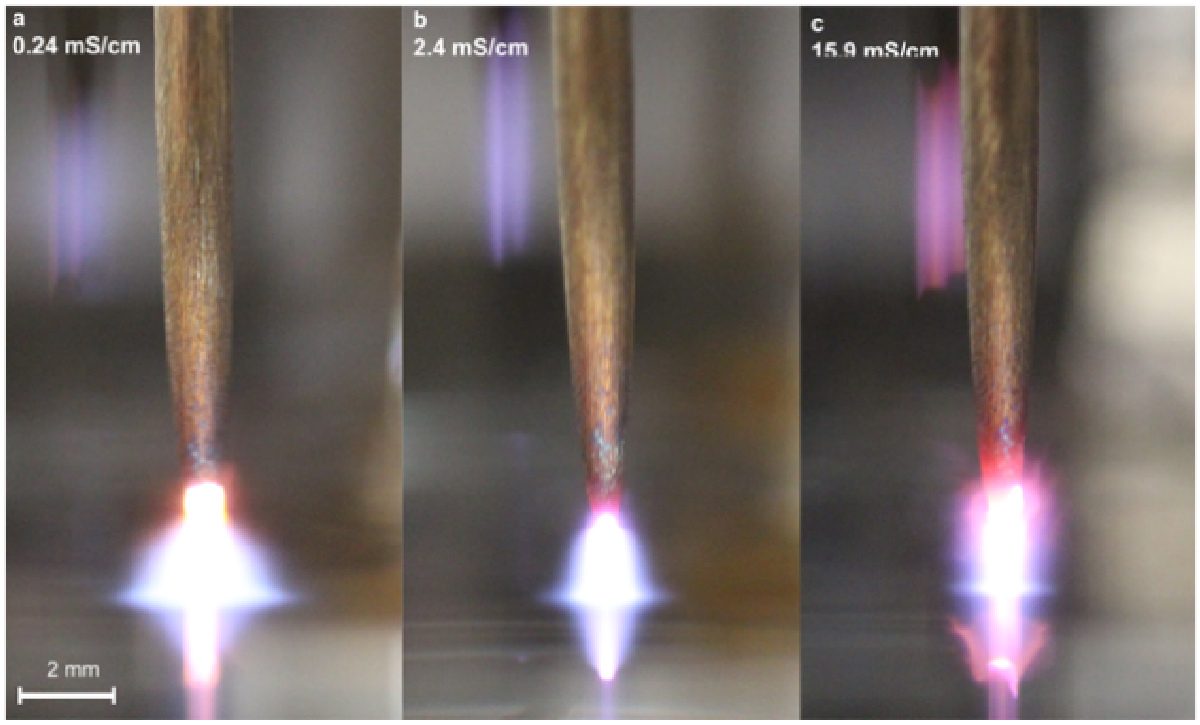


Figure 6.1 Figure taken from [2] showing the spreading of a DC discharge as solution conductivity is decreased.

pollutant degradation.

Zapdos can also be extended in a couple of other ways. To accurately describe electron behavior in the cathode of DC discharges and perhaps at the plasma-liquid interface, a kinetic instead of fluid model is required. MOOSE is capable of defining independent variables in addition to the traditional three spatial coordinates and time; the developers of RATTLESNAKE, another MOOSE application, use an energy variable as well as two streaming variables. It is conceivable that a kinetic model could be developed for the sheath regions and self-consistently coupled to a fluid model in the bulk regions, all underneath the Zapdos umbrella. In addition to kinetic models in the sheath it may be possible to self-consistently incorporate atomistic or molecular dynamic simulations of the interface to accurately capture interfacial electron behavior. There are current efforts in the MOOSE community to couple SPPARKS, a kinetic Monte Carlo code, and LAMMPS, a molecular dynamics simulator, to MOOSE applications.

Although not directly related to plasma-liquids, other students in our group are looking to

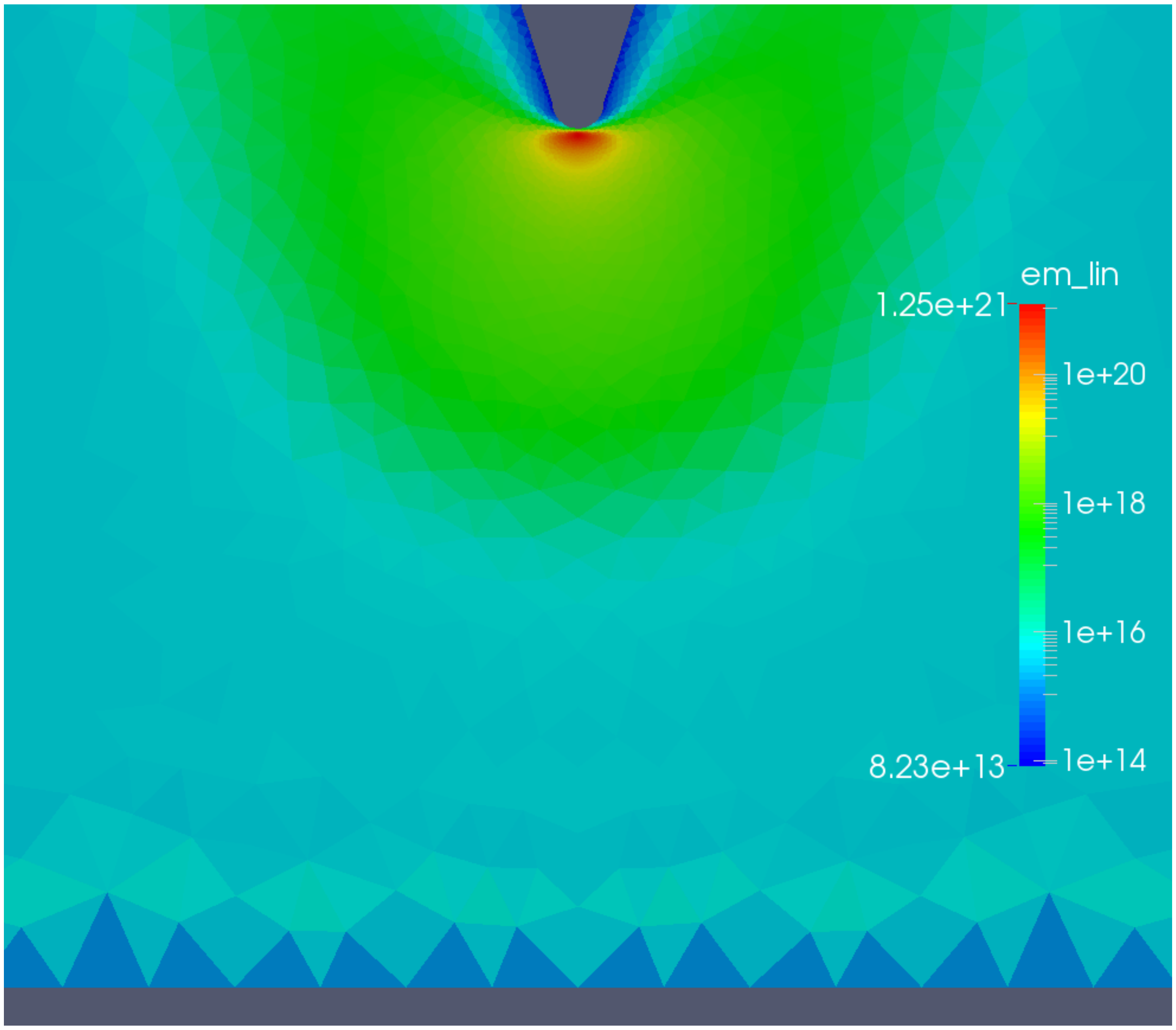


Figure 6.2 Initial 2-D axisymmetric modelling efforts for a needle-to-plane DC discharge. Figure shows electron density profile as space charge builds near the needle before propagating to the water anode. For future simulations in which the discharge is fully developed, additional mesh refinement will be needed at the anode.

extend Zapdos to modelling of RF antennaes on structures like ITER as well as simulation of low-pressure plasmas related to semiconductor manufacture. Additionally a group in Chengdu, China is interested in using Zapdos to model $CO_2$ to CO conversion with microwave and RF plasmas. With a host of potential projects, the future of Zapdos appears bright!

# BIBLIOGRAPHY

[1] Wei Tian and Mark J Kushner. Atmospheric pressure dielectric barrier discharges interacting with liquid covered tissue. *Journal of Physics D: Applied Physics*, 47(16):165201, April 2014.

[2] Paul Rumbach, David M Bartels, R Mohan Sankaran, and David B Go. The solvation of electrons by an atmospheric-pressure plasma. *Nature communications*, 6, 2015.

[3] `https://github.com/lindsayad/pythonScripts`.

[4] Joseph Flaherty. Finite element approximation.

[5] M G Kong, G Kroesen, G Morfill, T Nosenko, T Shimizu, J van Dijk, and J L Zimmermann. Plasma medicine: an introductory review. *New Journal of Physics*, 11(11):115012, November 2009.

[6] Mounir Laroussi. Low-temperature plasmas for medicine? *Plasma Science, IEEE Transactions on*, 37(6):714–725, 2009.

[7] K. Shimizu, H. Fukunaga, and M. Blajan. Biomedical applications of atmospheric microplasma. *Current Applied Physics*, pages 1–8, January 2014.

[8] Th. von Woedtke, H.-R. Metelmann, and K.-D. Weltmann. Clinical Plasma Medicine: State and Perspectives of in Vivo Application of Cold Atmospheric Plasma. *Contributions to Plasma Physics*, 54(2):104–117, February 2014.

[9] Th. von Woedtke, S. Reuter, K. Masur, and K.-D. Weltmann. Plasmas for medicine. *Physics Reports*, 530(4):291–320, September 2013.

[10] Vanessa Joubert, Cyril Cheype, Jean Bonnet, Denis Packan, Jean-Pierre Garnier, Justin Teissié, and Vincent Blanckaert. Inactivation of Bacillus subtilis var. niger of both spore

and vegetative forms by means of corona discharges applied in water. *Water research*, 47(3):1381–9, March 2013.

[11] Derek C Johnson, David S Dandy, and Vasgen a Shamamian. Development of a tubular high-density plasma reactor for water treatment. *Water research*, 40(2):311–22, January 2006.

[12] B. R. Locke, M. Sato, P. Sunka, M. R. Hoffmann, and J.-S. Chang. Electrohydraulic Discharge and Nonthermal Plasma for Water Treatment. *Industrial & Engineering Chemistry Research*, 45(3):882–905, February 2006.

[13] J Theron, J a Walker, and T E Cloete. Nanotechnology and water treatment: applications and emerging opportunities. *Critical reviews in microbiology*, 34(1):43–69, January 2008.

[14] Dayonna P. Park, Kevin Davis, Samid Gilani, Christal-Anne Alonzo, Danil Dobrynin, Gary Friedman, Alexander Fridman, Alexander Rabinovich, and Gregory Fridman. Reactive nitrogen species produced in water by non-equilibrium plasma increase plant growth rate and nutritional yield. *Current Applied Physics*, 13:S19–S29, March 2013.

[15] Alex Lindsay, Brandon Byrns, Wesley King, Asish Andhvarapou, Jeb Fields, Detlef Knappe, William Fonteno, and Steven Shannon. Fertilization of Radishes, Tomatoes, and Marigolds Using a Large-Volume Atmospheric Glow Discharge. *Plasma Chemistry and Plasma Processing*, 34(6):1271–1290, August 2014.

[16] P Lukes, E Dolezalova, I Sisrova, and M Clupek. Aqueous-phase chemistry and bactericidal effects from an air discharge plasma in contact with water: evidence for the formation of peroxynitrite through a pseudo-second-order post-discharge reaction of H 2 O 2 and HNO 2. *Plasma Sources Science and Technology*, 23(1):015019, February 2014.

[17] Matthew J Traylor, Matthew J Pavlovich, Sharmin Karim, Pritha Hait, Yukinori Sakiyama, Douglas S Clark, and David B Graves. Long-term antibacterial efficacy of air plasma-activated water. *Journal of Physics D: Applied Physics*, 44(47):472001, November 2011.

[18] Peter Bruggeman, Daan Schram, Manuel Á González, Robby Rego, Michael G Kong, and Christophe Leys. Characterization of a direct dc-excited discharge in water by optical emission spectroscopy. *Plasma Sources Science and Technology*, 18(2):025017, May 2009.

[19] Matthew J Pavlovich, Hung-Wen Chang, Yukinori Sakiyama, Douglas S Clark, and David B Graves. Ozone correlates with antibacterial effects from indirect air dielectric barrier discharge treatment of water. *Journal of Physics D: Applied Physics*, 46(14):145202, April 2013.

[20] Ippei Yagi, Ryo Ono, Tetsuji Oda, and Koichi Takaki. Two-dimensional lif measurements of humidity and oh density resulting from evaporated water from a wet surface in plasma for medical use. *Plasma Sources Science and Technology*, 24(1):015002, 2015.

[21] Peter Bruggeman, Jingjing Liu, Joris Degroote, Michael G Kong, Jan Vierendeels, and Christophe Leys. Dc excited glow discharges in atmospheric pressure air in pin-to-water electrode systems. *Journal of Physics D: Applied Physics*, 41(21):215201, 2008.

[22] C. Chen, D. X. Liu, Z. C. Liu, a. J. Yang, H. L. Chen, G. Shama, and M. G. Kong. A Model of Plasma-Biofilm and Plasma-Tissue Interactions at Ambient Pressure. *Plasma Chemistry and Plasma Processing*, 34(3):403–441, April 2014.

[23] Brian Lay, Richard S Moss, Shahid Rauf, and Mark J Kushner. Breakdown processes in metal halide lamps. *Plasma Sources Science and Technology*, 12(1):8, 2002.

[24] Zhongmin Xiong and Mark J Kushner. Surface corona-bar discharges for production of pre-ionizing uv light for pulsed high-pressure plasmas. *Journal of Physics D: Applied Physics*, 43(50):505204, 2010.

[25] Natalia Yu Babaeva, Wei Tian, and Mark J Kushner. The interaction between plasma filaments in dielectric barrier discharges and liquid covered wounds: electric fields delivered to model platelets and cells. *Journal of Physics D: Applied Physics*, 47(23):235201, June 2014.

[26] Matthias Heil, Andrew L Hazel, and Jonathan Boyle. Solvers for large-displacement fluid–structure interaction problems: segregated versus monolithic approaches. *Computational Mechanics*, 43(1):91–101, 2008.

[27] R Morrow and DR McKenzie. The time-dependent development of electric double-layers in pure water at metal electrodes: the effect of an applied voltage on the local ph. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, page rspa20110323. The Royal Society, 2011.

[28] Tatsuru Shirafuji, Akihiro Nakamura, and Fumiyoshi Tochikubo. Numerical simulation of electric double layer in contact with dielectric barrier discharge: Effects of ion transport parameters in liquid. *Japanese Journal of Applied Physics*, 53(3S2):03DG04, 2014.

[29] `https://www.comsol.com/`.

[30] Yukinori Sakiyama and David B Graves. Nonthermal atmospheric rf plasma in one-dimensional spherical coordinates: Asymmetric sheath structure and the discharge mechanism. *Journal of applied physics*, 101(7):073306, 2007.

[31] Y Sakiyama and DB Graves. Finite element analysis of an atmospheric pressure rf-excited plasma needle. *Journal of Physics D: Applied Physics*, 39(16):3451, 2006.

[32] `http://www.openfoam.com/`.

[33] `https://www.csc.fi/web/elmer`.

[34] `http://www.code-aster.org/spip.php?rubrique2`.

[35] `mooseframework.org`.

[36] `https://github.com/lindsayad/zapdos`.

[37] Claire Tendero, Christelle Tixier, Pascal Tristant, Jean Desmaison, and Philippe Leprince. Atmospheric pressure plasmas: A review. *Spectrochimica Acta Part B: Atomic Spectroscopy*, 61(1):2–30, 2006.

[38] Jaeyoung Park, I Henins, HW Herrmann, GS Selwyn, and RF Hicks. Discharge phenomena of an atmospheric pressure radio-frequency capacitive plasma source. *Journal of Applied Physics*, 89(1):20–28, 2001.

[39] M Moisan, G Sauve, Z Zakrzewski, and J Hubert. An atmospheric pressure waveguide-fed microwave plasma torch: the tia design. *Plasma Sources Science and Technology*, 3(4):584, 1994.

[40] BR Locke, M Sato, P Sunka, MR Hoffmann, and J-S Chang. Electrohydraulic discharge and nonthermal plasma for water treatment. *Industrial & engineering chemistry research*, 45(3):882–905, 2006.

[41] Brandon Byrns, Daniel Wooten, Alexander Lindsay, and Steven Shannon. A vhf driven coaxial atmospheric air plasma: electrical and optical characterization. *Journal of Physics D: Applied Physics*, 45(19):195204, 2012.

[42] Ulrich Kogelschatz. Atmospheric-pressure plasma technology. *Plasma Physics and Controlled Fusion*, 46(12B):B63, 2004.

[43] JA Bakken. High temperature processing and numerical modelling of thermal plasmas in norway. *Pure and applied chemistry*, 66(6):1239–1246, 1994.

[44] J Lelièvre, N Dubreuil, and J-L Brisset. Electrolysis processes in dc corona discharges in humid air. *Journal de Physique III*, 5(4):447–457, 1995.

[45] Matthew J Traylor, Matthew J Pavlovich, Sharmin Karim, Pritha Hait, Yukinori Sakiyama, Douglas S Clark, and David B Graves. Long-term antibacterial efficacy of air plasma-activated water. *Journal of Physics D: Applied Physics*, 44(47):472001, 2011.

[46] Božena Šerá, Iveta Gajdová, Mirko Černák, Bogdan Gavril, Eugen Hnatiuc, Dušan Kováčik, Vítězslav Kříha, Jan Sláma, Michal Šerỳ, and Petr Špatenka. How various plasma sources may affect seed germination and growth. In *Optimization of Electrical and Electronic Equipment (OPTIM), 2012 13th International Conference on*, pages 1365–1370. IEEE, 2012.

[47] Edward Bormashenko, Roman Grynyov, Yelena Bormashenko, and Elyashiv Drori. Cold radiofrequency plasma treatment modifies wettability and germination speed of plant seeds. *Scientific reports*, 2, 2012.

[48] Zhuwen Zhou, Yanfen Huang, Size Yang, and Wei Chen. Introduction of a new atmospheric pressure plasma device and application on tomato seeds*. *Agricultural Sciences*, 2(1):23, 2011.

[49] Ming-jing Huang and Mei-qiang Yin. Effect of radio frequency discharge plasma on wheatâĂŹs seed germination and seedling growth. *Journal of Shanxi Agricultural University (Natural Science Edition)*, 5:006, 2010.

[50] Irina Filatova, Viktor Azharonok, Alexander Shik, Alexandra Antoniuk, and Natalia Terletskaya. Fungicidal effects of plasma and radio-wave pre-treatments on seeds of grain crops and legumes. In *Plasma for Bio-Decontamination, Medicine and Food Security*, pages 469–479. Springer, 2012.

[51] K Takaki, J Takahata, S Watanabe, N Satta, O Yamada, T Fujio, and Y Sasaki. Improvements in plant growth rate using underwater discharge. In *Journal of Physics: Conference Series*, volume 418, page 012140. IOP Publishing, 2013.

[52] Dayonna P Park, Kevin Davis, Samid Gilani, Christal-Anne Alonzo, Danil Dobrynin, Gary Friedman, Alexander Fridman, Alexander Rabinovich, and Gregory Fridman. Reactive nitrogen species produced in water by non-equilibrium plasma increase plant growth rate and nutritional yield. *Current Applied Physics*, 13:S19–S29, 2013.

[53] R Ingels, D Graves, S Anderson, R Koller, et al. Modern plasma technology for nitrogen fixation-new opportunities? proceedings of the international fertiliser society 771, london, uk, 24th june, 2015. In *Proceedings-International Fertiliser Society*, number 771. International Fertiliser Society, 2015.

[54] Rhododendronbusch. [Online; accessed February 22, 2016].

[55] Jung Ho Suh and Madjid Mohseni. A study on the relationship between biodegradability enhancement and oxidation of 1, 4-dioxane using ozone and hydrogen peroxide. *Water Research*, 38(10):2596–2604, 2004.

[56] Wim TM Audenaert, Yoshi Vermeersch, Stijn WH Van Hulle, Pascal Dejans, Ann Dumoulin, and Ingmar Nopens. Application of a mechanistic uv/hydrogen peroxide model at full-scale: Sensitivity analysis, calibration and performance evaluation. *Chemical Engineering Journal*, 171(1):113–126, 2011.

[57] Fernando J Beltran. *Ozone reaction kinetics for water and wastewater systems*. CRC Press, 2003.

[58] Kitirote Wantala, Pongtanawat Khemthong, Jatuporn Wittayakun, and Nurak Grisdanurak. Visible light-irradiated degradation of alachlor on fe-tio2 with assistance of h2o2. *Korean Journal of Chemical Engineering*, 28(11):2178–2183, 2011.

[59] Miguel Pelaez, Nicholas T Nolan, Suresh C Pillai, Michael K Seery, Polycarpos Falaras, Athanassios G Kontos, Patrick SM Dunlop, Jeremy WJ Hamilton, J Anthony Byrne,

Kevin O'shea, et al. A review on the visible light active titanium dioxide photocatalysts for environmental applications. *Applied Catalysis B: Environmental*, 125:331–349, 2012.

[60] Ki-Chang Lee and Kwang-Ho Choo. Hybridization of tio 2 photocatalysis with coagulation and flocculation for 1, 4-dioxane removal in drinking water treatment. *Chemical engineering journal*, 231:227–235, 2013.

[61] *The UV/Oxidation Handbook*. Solarchem Environmental Systems, 1994.

[62] Su no G. [Online; accessed February 22, 2016].

[63] Chuyang Y Tang, Q Shiang Fu, AP Robertson, Craig S Criddle, and James O Leckie. Use of reverse osmosis membranes to remove perfluorooctane sulfonate (pfos) from semiconductor wastewater. *Environmental science & technology*, 40(23):7343–7349, 2006.

[64] Jie Cheng, Chad D Vecitis, Hyunwoong Park, Brian T Mader, and Michael R Hoffmann. Sonochemical degradation of perfluorooctane sulfonate (pfos) and perfluorooctanoate (pfoa) in landfill groundwater: environmental matrix effects. *Environmental science & technology*, 42(21):8057–8063, 2008.

[65] Chuyang Y Tang, Q Shiang Fu, Craig S Criddle, and James O Leckie. Effect of flux (transmembrane pressure) and membrane properties on fouling and rejection of reverse osmosis and nanofiltration membranes treating perfluorooctane sulfonate containing wastewater. *Environmental science & technology*, 41(6):2008–2014, 2007.

[66] Kellyn S Betts. Chemical exposures: Not immune to pfos effects? *Environmental health perspectives*, 116(7):A290, 2008.

[67] Hisao Hori, Ari Yamamoto, Etsuko Hayakawa, Sachi Taniyasu, Nobuyoshi Yamashita, Shuzo Kutsuna, Hiroshi Kiatagawa, and Ryuichi Arakawa. Efficient decomposition of environmentally persistent perfluorocarboxylic acids by use of persulfate as a photochemical oxidant. *Environmental Science & Technology*, 39(7):2383–2388, 2005.

[68] Hisao Hori, Yumiko Nagaoka, Ari Yamamoto, Taizo Sano, Nobuyoshi Yamashita, Sachi Taniyasu, Shuzo Kutsuna, Issey Osaka, and Ryuichi Arakawa. Efficient decomposition of environmentally persistent perfluorooctanesulfonate and related fluorochemicals using zerovalent iron in subcritical water. *Environmental science & technology*, 40(3):1049–1054, 2006.

[69] Hiroshi Moriwaki, Youichi Takagi, Masanobu Tanaka, Kenshiro Tsuruho, Kenji Okitsu, and Yasuaki Maeda. Sonochemical decomposition of perfluorooctane sulfonate and perfluorooctanoic acid. *Environmental science & technology*, 39(9):3388–3392, 2005.

[70] David B Graves. The emerging role of reactive oxygen and nitrogen species in redox biology and some implications for plasma applications to medicine and biology. *Journal of Physics D: Applied Physics*, 45(26):263001, 2012.

[71] Klaus Dieter Weltmann, Ronny Brandenburg, Thomas von Woedtke, J Ehlbeck, R Foest, M Stieber, and E Kindel. Antimicrobial treatment of heat sensitive products by miniaturized atmospheric pressure plasma jets (appjs). *Journal of Physics D: Applied Physics*, 41(19):194008, 2008.

[72] Michael Keidar, Alex Shashurin, Olga Volotskova, Mary Ann Stepp, Priya Srinivasan, Anthony Sandler, and Barry Trink. Cold atmospheric plasma in cancer therapya). *Physics of Plasmas (1994-present)*, 20(5):057101, 2013.

[73] Andrei Vasile Nastuta, Ionut Topala, Constantin Grigoras, Valentin Pohoata, and Gheorghe Popa. Stimulation of wound healing by helium atmospheric pressure plasma treatment. *Journal of Physics D: Applied Physics*, 44(10):105204, 2011.

[74] David B Graves. Oxy-nitroso shielding burst model of cold atmospheric plasma therapeutics. *Clinical Plasma Medicine*, 2014.

[75] Alexander Lindsay, Carly Anderson, Elmar Slikboer, Steven Shannon, and David Graves. Momentum, heat, and neutral mass transport in convective atmospheric pressure plasma-liquid systems and implications for aqueous targets. *Journal of Physics D: Applied Physics*, 48(42):424007, 2015.

[76] Ryo Ono and Tetsuji Oda. Measurement of hydroxyl radicals in an atmospheric pressure discharge plasma by using laser-induced fluorescence. In *Industry Applications Conference, 1998. Thirty-Third IAS Annual Meeting. The 1998 IEEE*, volume 3, pages 1777–1783. IEEE, 1998.

[77] Ryo Ono and Tetsuji Oda. Oh radical measurement in a pulsed arc discharge plasma observed by a lif method. *Industry Applications, IEEE Transactions on*, 37(3):709–714, 2001.

[78] Yusuke Nakagawa, Ryo Ono, and Tetsuji Oda. Density and temperature measurement of oh radicals in atmospheric-pressure pulsed corona discharge in humid air. *Journal of Applied Physics*, 110(7):073304, 2011.

[79] T Verreycken, RM Van der Horst, AHFM Baede, EM Van Veldhuizen, and PJ Bruggeman. Time and spatially resolved lif of oh in a plasma filament in atmospheric pressure he–h2o. *Journal of Physics D: Applied Physics*, 45(4):045205, 2012.

[80] K Niemi, V Schulz-Von Der Gathen, and HF Döbele. Absolute atomic oxygen density measurements by two-photon absorption laser-induced fluorescence spectroscopy in an rf-excited atmospheric pressure plasma jet. *Plasma Sources Science and Technology*, 14(2):375, 2005.

[81] Seiji Kanazawa, Yasuyuki Shuto, Naruaki Sato, Toshikazu Ohkubo, Yukiharu Nomoto, Jerzy Mizeraczyk, and Jen-Shih Chang. Two-dimensional imaging of no density profiles by

lif technique in a pipe with nozzles electrode during no treatment. *Industry Applications, IEEE Transactions on*, 39(2):333–339, 2003.

[82] C Hibert, I Gaurand, O Motret, and JM Pouvesle. [oh (x)] measurements by resonant absorption spectroscopy in a pulsed dielectric barrier discharge. *Journal of applied physics*, 85(10):7070–7075, 1999.

[83] L. Zhao and K. Adamiak. EHD flow in air produced by electric corona discharge in pinâĂŞplate configuration. *Journal of Electrostatics*, 63(3-4):337–350, March 2005.

[84] R Byron Bird, Warren E Stewart, and Edwin N Lightfoot. *Transport phenomena*. John Wiley & Sons, 2007.

[85] Edward Lansing Cussler. *Diffusion: mass transfer in fluid systems*. Cambridge university press, 2009.

[86] C Antoine. Thermodynamic vapor pressures: New relation between the pressures and the temperatures (thermodynamique, tensions des vapeurs: Novelle relation entre les tensions et les temperatures). *CR Hebd. Seances Acad. Sci*, 107(681):836, 1888.

[87] Alexandre C Dimian, Costin S Bildea, and Anton A Kiss. *Integrated design and simulation of chemical processes*, volume 35. Elsevier, 2014.

[88] Ding-Xin Liu, Peter Bruggeman, Felipe Iza, Ming-Zhe Rong, and Michael G Kong. Global model of low-temperature atmospheric-pressure he+ h2o plasmas. *Plasma Sources Science and Technology*, 19(2):025018, 2010.

[89] Barry Halliwell. *Free Radicals in Biochemistry and Medicine*. Wiley-VCH Verlag GmbH & Co. KGaA, 2006.

[90] Yukinori Sakiyama, David B Graves, Hung-Wen Chang, Tetsuji Shimizu, and Gregor E Morfill. Plasma chemistry model of surface microdischarge in humid air and dynamics of

reactive neutral species. *Journal of Physics D: Applied Physics*, 45(42):425201, October 2012.

[91] Julia M Khalack and Alexander P Lyubartsev. Solvation structure of hydroxyl radical by car-parrinello molecular dynamics. *The Journal of Physical Chemistry A*, 109(2):378–386, 2005.

[92] RL McMurtrie and FG Keyes. A measurement of the diffusion coefficient of hydrogen peroxide vapor into air. *Journal of the American Chemical Society*, 70(11):3755–3758, 1948.

[93] Ian G Zacharia and William M Deen. Diffusivity and solubility of nitric oxide in water and saline. *Annals of biomedical engineering*, 33(2):214–222, 2005.

[94] Brian T Skinn and William M Deen. A nitrogen dioxide delivery system for biological media. *Free Radical Biology and Medicine*, 56:44–53, 2013.

[95] George B Wills and Hsi-Sheng Yeh. Diffusion coefficient of aqueous nitric acid at 25. deg. as function of concentration from 0.1 to 1.0 m. *Journal of Chemical & Engineering Data*, 16(1):76–77, 1971.

[96] Jan-Ulrich Kreft, Cristian Picioreanu, Julian WT Wimpenny, and Mark CM van Loosdrecht. Individual-based modelling of biofilms. *Microbiology*, 147(11):2897–2912, 2001.

[97] Noam Agmon. The grotthuss mechanism. *Chemical Physics Letters*, 244(5):456–462, 1995.

[98] N Kitamura, K Sasaki, H Misawa, H Masuhara, and H Masuhara. Microchemistry: Spectroscopy and chemistry in small domains. *Elsevier Science, Amsterdam*, 1994.

[99] Lowell G Wayne and Don M Yost. Kinetics of the rapid gas phase reaction between no, no2, and h2o. *The Journal of Chemical Physics*, 19(1):41–47, 1951.

[100] Rajesh Dorai. *Modeling of atmospheric pressure plasma processing of gases and surfaces*. PhD thesis, University of Illinois at Urbana-Champaign, 2002.

[101] John W Coddington, James K Hurst, and Sergei V Lymar. Hydroxyl radical formation during peroxynitrous acid decomposition. *Journal of the American Chemical Society*, 121(11):2438–2443, 1999.

[102] Jong Yoon Park and Yin Nan Lee. Solubility and decomposition kinetics of nitrous acid in aqueous solution. *The Journal of Physical Chemistry*, 92(22):6294–6302, 1988.

[103] Sara Goldstein, Johan Lind, and Gábor Merényi. Chemistry of peroxynitrites as compared to peroxynitrates. *Chemical reviews*, 105(6):2457–2470, 2005.

[104] A JohnáElliot et al. Estimation of rate constants for near-diffusion-controlled reactions in water at high temperatures. *Journal of the Chemical Society, Faraday Transactions*, 86(9):1539–1547, 1990.

[105] Sara Goldstein, Gidon Czapski, Johan Lind, and Gabor Merényi. Tyrosine nitration by simultaneous generation ofâŃĚ no and oâĹł 2 under physiological conditions how the radicals do the job. *Journal of Biological Chemistry*, 275(5):3031–3036, 2000.

[106] S Solar, W Solar, and N Getoff. Reactivity of hydroxyl with tyrosine in aqueous solution studied by pulse radiolysis. *The Journal of Physical Chemistry*, 88(10):2091–2095, 1984.

[107] James J Carberry. *Chemical and catalytic reaction engineering*. Courier Dover Publications, 2001.

[108] William M Deen. *Analysis of Transport Phenomena (Topics in Chemical Engineering)*, volume 3. Oxford University Press, New York, 1998.

[109] GJM Hagelaar and LC Pitchford. Solving the boltzmann equation to obtain electron transport coefficients and rate coefficients for fluid models. *Plasma Sources Science and Technology*, 14(4):722, 2005.

[110] GJM Hagelaar, FJ De Hoog, and GMW Kroesen. Boundary conditions in fluid models of gas discharges. *Physical Review E*, 62(1):1452–1454, 2000.

[111] C Yamabe, SJ Buckman, and AV Phelps. Measurement of free-free emission from low-energy-electron collisions with ar. *Physical Review A*, 27(3):1345, 1983.

[112] `http://fr.lxcat.net/home/`.

[113] `https://github.com/aluque/bolos`.

[114] Albert D Richards, Brian E Thompson, and Herbert H Sawin. Continuum modeling of argon radio frequency glow discharges. *Applied physics letters*, 50(9):492–494, 1987.

[115] Emi Kawamura. Personal communication.

[116] Yong Jun Hong, Seung Min Lee, Gyoo Cheon Kim, and Jae Koo Lee. Modeling high-pressure microplasmas: Comparison of fluid modeling and particle-in-cell monte carlo collision modeling. *Plasma Processes and Polymers*, 5(6):583–592, 2008.

[117] Jun Choi, Felipe Iza, Jae Koo Lee, and Chang-Mo Ryu. Electron and ion kinetics in a dc microplasma at atmospheric pressure. *Plasma Science, IEEE Transactions on*, 35(5):1274–1278, 2007.

[118] GJM Hagelaar and GMW Kroesen. A monte carlo modelling study of the electrons in the microdischarges in plasma addressed liquid crystal displays. *Plasma Sources Science and Technology*, 9(4):605, 2000.

[119] `http://libmesh.github.io/`.

[120] `http://www.mcs.anl.gov/petsc/`.

[121] Dana A Knoll and David E Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[122] John E Dennis Jr and Robert B Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Siam, 1996.

[123] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[124] Todd Dupont, Richard P Kendall, and HH Rachford, Jr. An approximate factorization procedure for solving self-adjoint elliptic difference equations. *SIAM Journal on Numerical Analysis*, 5(3):559–573, 1968.

[125] Yousef Saad. *Iterative methods for sparse linear systems*. Siam, 1996.

[126] Larry A Schoof and Victor R Yarberry. Exodus ii: a finite element data model. Technical report, Sandia National Labs., Albuquerque, NM (United States), 1994.

[127] Frank-Rene Schaefer. Getpot version 1.0: Powerful input file and command line parser.

[128] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

[129] `http://www.paraview.org/`.

[130] R Morrow, DR McKenzie, and MMM Bilek. The time-dependent development of electric double-layers in saline solutions. *Journal of Physics D: Applied Physics*, 39(5):937, 2006.

[131] Mark M Benjamin. *Water chemistry*. Waveland Press, 2014.

[132] Norman Neill Greenwood and Alan Earnshaw. Chemistry of the elements. 1984.

[133] Jean-Louis Brisset and Eugen Hnatiuc. Peroxynitrite: a re-examination of the chemical properties of non-thermal discharges burning in air over aqueous solutions. *Plasma Chemistry and Plasma Processing*, 32(4):655–674, 2012.

[134] R Burlica, RG Grim, K-Y Shih, D Balkwill, and BR Locke. Bacteria inactivation using low power pulsed gliding arc discharges with water spray. *Plasma Processes and Polymers*, 7(8):640–649, 2010.

[135] D Moussa, F Abdelmalek, B Benstaali, A Addou, E Hnatiuc, and J-L Brisset. Acidity control of the gliding arc treatments of aqueous solutions: application to pollutant abatement and biodecontamination. *The European Physical Journal Applied Physics*, 29(02):189–199, 2005.

[136] Anoop Shankar, Jie Xiao, and Alan Ducatman. Perfluoroalkyl chemicals and chronic kidney disease in us adults. *American journal of epidemiology*, page kwr171, 2011.

[137] `https://github.com/lindsayad/programming`.

# APPENDIX

APPENDIX

# RELEVANT CODE SNIPPETS

## A.1 Zapdos Input File

Below is a Zapdos input file that is exemplary of the input files used to produce the simulation results in section 2.2. For descriptions of many of the input blocks and the classes used in them, see section 3.1.

```
dom0Scale=1e-3
dom1Scale=1e-7

[GlobalParams]
  offset = 20
  # offset = 0
  potential_units = kV
  # potential_units = V
[]

[Mesh]
  # type = GeneratedMesh
  # nx = 2
  # xmax = 1.1
  # dim = 1
  type = FileMesh
  file = 'liquidNew.msh'
[]

[MeshModifiers]
  [./interface]
    type = SideSetsBetweenSubdomains
    master_block = '0'
    paired_block = '1'
    new_boundary = 'master0_interface'
    # depends_on = 'box'
  [../]
  [./interface_again]
    type = SideSetsBetweenSubdomains
    master_block = '1'
    paired_block = '0'
    new_boundary = 'master1_interface'
    # depends_on = 'box'
  [../]
  [./left]
    type = SideSetsFromNormals
    normals = '-1 0 0'
    new_boundary = 'left'
  [../]
  [./right]
    type = SideSetsFromNormals
    normals = '1 0 0'
    new_boundary = 'right'
  [../]
  # [./box]
  #   type = SubdomainBoundingBox
  #   bottom_left = '0.55 0 0'
  #   top_right = '1.1 1. 0'
  #   block_id = 1
  # [../]
[]

[Problem]
  type = FEProblem
  # kernel_coverage_check = false
[]
```

```

[Preconditioning]
  [./smp]
    type = SMP
    full = true
  [../]
[]

[Executioner]
  type = Transient
  end_time = 1e-1
  # end_time = 10
  petsc_options = '-snes_converged_reason -snes_linesearch_monitor'
  # petsc_options = '-snes_test_display'
  solve_type = NEWTON
  petsc_options_iname = '-pc_type -pc_factor_shift_type -pc_factor_shift_amount -
    ksp_type -snes_linesearch_minlambda'
  petsc_options_value = 'lu NONZERO 1.e-10 preonly 1e-3'
  # petsc_options_iname = '-snes_type'
  # petsc_options_value = 'test'
 nl_rel_tol = 1e-4
 nl_abs_tol = 5.4e-5
  dtmin = 1e-12
  [./TimeStepper]
    type = IterationAdaptiveDT
    cutback_factor = 0.4
    dt = 1e-9
    # dt = 1.1
    growth_factor = 1.2
   optimal_iterations = 15
  [../]
[]

[Outputs]
  print_perf_log = true
  print_linear_residuals = false
  [./out]
    type = Exodus
  [../]
[]

[Debug]
  show_var_residual_norms = true
```

```
[]

[UserObjects]
  [./data_provider]
    type = ProvideMobility
    electrode_area = 5.02e-7 # Formerly 3.14e-6
    ballast_resist = 1e6
    e = 1.6e-19
    # electrode_area = 1.1
    # ballast_resist = 1.1
    # e = 1.1
  [../]
[]

[Kernels]
  [./em_time_deriv]
    type = ElectronTimeDerivative
    variable = em
    block = 0
  [../]
  [./em_advection]
    type = EFieldAdvectionElectrons
    variable = em
    potential = potential
    mean_en = mean_en
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./em_diffusion]
    type = CoeffDiffusionElectrons
    variable = em
    mean_en = mean_en
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./em_ionization]
    type = ElectronsFromIonization
    variable = em
    potential = potential
    mean_en = mean_en
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./em_log_stabilization]
    type = LogStabilizationMoles
    variable = em
    block = 0
  [../]
 # [./em_advection_stabilization]
 #   type = EFieldArtDiff
 #   variable = em
 #   potential = potential
 # [../]

  [./emliq_time_deriv]
    type = ElectronTimeDerivative
    variable = emliq
    block = 1
  [../]
  [./emliq_advection]
    type = EFieldAdvection
    variable = emliq
    potential = potential
    block = 1
    position_units = ${dom1Scale}
  [../]
  [./emliq_diffusion]
    type = CoeffDiffusion
    variable = emliq
    block = 1
    position_units = ${dom1Scale}
  [../]
  [./emliq_reactant_first_order_rxn]
    type = ReactantFirstOrderRxn
    variable = emliq
    block = 1
  [../]
  [./emliq_water_bi_sink]
    type = ReactantAARxn
    variable = emliq
    block = 1
  [../]
  [./emliq_log_stabilization]
    type = LogStabilizationMoles
    variable = emliq
    block = 1
  [../]

  [./potential_diffusion_dom1]
    type = CoeffDiffusionLin
    variable = potential
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./potential_diffusion_dom2]
    type = CoeffDiffusionLin
    variable = potential
    block = 1
    position_units = ${dom1Scale}
  [../]
  [./Arp_charge_source]
    type = ChargeSourceMoles_KV
    variable = potential
    charged = Arp
    block = 0
  [../]
  [./em_charge_source]
    type = ChargeSourceMoles_KV
    variable = potential
    charged = em
    block = 0
  [../]
  [./emliq_charge_source]
    type = ChargeSourceMoles_KV
    variable = potential
    charged = emliq
    block = 1
  [../]
  [./OHm_charge_source]
    type = ChargeSourceMoles_KV
    variable = potential
    charged = OHm
    block = 1
  [../]

  [./Arp_time_deriv]
    type = ElectronTimeDerivative
    variable = Arp
    block = 0
  [../]
  [./Arp_advection]
    type = EFieldAdvection
    variable = Arp
    potential = potential
    position_units = ${dom0Scale}
    block = 0
  [../]
  [./Arp_diffusion]
    type = CoeffDiffusion
    variable = Arp
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./Arp_ionization]
    type = IonsFromIonization
    variable = Arp
    potential = potential
    em = em
    mean_en = mean_en
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./Arp_log_stabilization]
    type = LogStabilizationMoles
    variable = Arp
    block = 0
  [../]
 # [./Arp_advection_stabilization]
 #   type = EFieldArtDiff
 #   variable = Arp
 #   potential = potential
 # [../]

  [./OHm_time_deriv]
    type = ElectronTimeDerivative
    variable = OHm
    block = 1
  [../]
  [./OHm_advection]
    type = EFieldAdvection
    variable = OHm
    potential = potential
    block = 1
    position_units = ${dom1Scale}
  [../]
  [./OHm_diffusion]
    type = CoeffDiffusion
    variable = OHm
    block = 1
    position_units = ${dom1Scale}
  [../]
  [./OHm_log_stabilization]
    type = LogStabilizationMoles
    variable = OHm
    block = 1
  [../]
 # [./OHm_advection_stabilization]
 #   type = EFieldArtDiff
 #   variable = OHm
 #   potential = potential
 #   block = 1
 # [../]
  [./OHm_product_first_order_rxn]
    type = ProductFirstOrderRxn
    variable = OHm
    v = emliq
    block = 1
  [../]
  [./OHm_product_aabb_rxn]
    type = ProductAABBRxn
    variable = OHm
    v = emliq
    block = 1
  [../]

  [./mean_en_time_deriv]
    type = ElectronTimeDerivative
    variable = mean_en
    block = 0
  [../]
  [./mean_en_advection]
    type = EFieldAdvectionEnergy
    variable = mean_en
    potential = potential
    em = em
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_diffusion]
    type = CoeffDiffusionEnergy
    variable = mean_en
    em = em
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_joule_heating]
    type = JouleHeating
    variable = mean_en
    potential = potential
    em = em
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_ionization]
    type = ElectronEnergyLossFromIonization
    variable = mean_en
    potential = potential
    em = em
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_elastic]
    type = ElectronEnergyLossFromElastic
    variable = mean_en
    potential = potential
    em = em
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_excitation]
    type = ElectronEnergyLossFromExcitation
    variable = mean_en
    potential = potential
    em = em
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_log_stabilization]
```

```
    type = LogStabilizationMoles
    variable = mean_en
    block = 0
    offset = 15
  [../]
  # [./mean_en_advection_stabilization]
  #   type = EFieldArtDiff
  #   variable = mean_en
  #   potential = potential
  # [../]
[]

[Variables]
  [./potential]
  [../]
  [./em]
    block = 0
  [../]
  [./emliq]
    block = 1
    # scaling = 1e-4
  [../]

  [./Arp]
    block = 0
  [../]

  [./mean_en]
    block = 0
    # scaling = 1e-1
  [../]

  [./OHm]
    block = 1
    # scaling = 1e-4
  [../]
[]

[AuxVariables]
  [./e_temp]
    block = 0
  [../]
  [./x]
```

```
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./rho]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./rholiq]
    block = 1
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./em_lin]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./emliq_lin]
    order = CONSTANT
    family = MONOMIAL
    block = 1
  [../]
  [./Arp_lin]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./OHm_lin]
    block = 1
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./Efield]
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./Current_em]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./Current_emliq]
```

```
    order = CONSTANT
    family = MONOMIAL
    block = 1
  [../]
  [./Current_Arp]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./Current_OHm]
    block = 1
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./tot_gas_current]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./tot_liq_current]
    block = 1
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./tot_flux_OHm]
    block = 1
    order = CONSTANT
    family = MONOMIAL
  [../]
  [./EFieldAdvAux_em]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./DiffusiveFlux_em]
    order = CONSTANT
    family = MONOMIAL
    block = 0
  [../]
  [./EFieldAdvAux_emliq]
    order = CONSTANT
    family = MONOMIAL
    block = 1
  [../]
  [./DiffusiveFlux_emliq]
    order = CONSTANT
    family = MONOMIAL
    block = 1
  [../]
  [./PowerDep_em]
   order = CONSTANT
   family = MONOMIAL
   block = 0
  [../]
  [./PowerDep_Arp]
   order = CONSTANT
   family = MONOMIAL
   block = 0
  [../]
  [./ProcRate_el]
   order = CONSTANT
   family = MONOMIAL
   block = 0
  [../]
  [./ProcRate_ex]
   order = CONSTANT
   family = MONOMIAL
   block = 0
  [../]
  [./ProcRate_iz]
   order = CONSTANT
   family = MONOMIAL
   block = 0
  [../]
[]

[AuxKernels]
  [./PowerDep_em]
    type = PowerDep
    density_log = em
    potential = potential
    art_diff = false
    potential_units = kV
    variable = PowerDep_em
  [../]
  [./PowerDep_Arp]
```

```
    type = PowerDep
    density_log = Arp
    potential = potential
    art_diff = false
    potential_units = kV
    variable = PowerDep_Arp
  [../]
  [./ProcRate_el]
    type = ProcRate
    em = em
    potential = potential
    proc = el
    variable = ProcRate_el
  [../]
  [./ProcRate_ex]
    type = ProcRate
    em = em
    potential = potential
    proc = ex
    variable = ProcRate_ex
  [../]
  [./ProcRate_iz]
    type = ProcRate
    em = em
    potential = potential
    proc = iz
    variable = ProcRate_iz
  [../]
  [./e_temp]
    type = ElectronTemperature
    variable = e_temp
    electron_density = em
    mean_en = mean_en
    block = 0
  [../]
  [./x]
    type = Position
    variable = x
  [../]
  [./rho]
    type = ParsedAux
    variable = rho
    args = 'em_lin Arp_lin'
    function = 'Arp_lin - em_lin'
    execute_on = 'timestep_end'
    block = 0
  [../]
  [./rholiq]
    type = ParsedAux
    variable = rholiq
    args = 'emliq_lin OHm_lin' # H3Op_lin OHm_lin'
    function = '-emliq_lin - OHm_lin' # 'H3Op_lin - em_lin - OHm_lin'
    execute_on = 'timestep_end'
    block = 1
  [../]
  [./tot_gas_current]
    type = ParsedAux
    variable = tot_gas_current
    args = 'Current_em Current_Arp'
    function = 'Current_em + Current_Arp'
    execute_on = 'timestep_end'
    block = 0
  [../]
  [./tot_liq_current]
    type = ParsedAux
    variable = tot_liq_current
    args = 'Current_emliq Current_OHm' # Current_H3Op Current_OHm'
    function = 'Current_emliq + Current_OHm' # + Current_H3Op + Current_OHm'
    execute_on = 'timestep_end'
    block = 1
  [../]
  [./em_lin]
    type = Density
    variable = em_lin
    density_log = em
    block = 0
  [../]
  [./emliq_lin]
    type = Density
    variable = emliq_lin
    density_log = emliq
    block = 1
  [../]
  [./Arp_lin]
    type = Density
    variable = Arp_lin
    density_log = Arp
    block = 0
  [../]
  [./OHm_lin]
    type = Density
    variable = OHm_lin
    density_log = OHm
    block = 1
  [../]
  [./Efield]
    type = Efield
    potential = potential
    variable = Efield
  [../]
  [./Current_em]
    type = Current
    potential = potential
    density_log = em
    variable = Current_em
    art_diff = false
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./Current_emliq]
    type = Current
    potential = potential
    density_log = emliq
    variable = Current_emliq
    art_diff = false
    block = 1
    position_units = ${dom1Scale}
  [../]
  [./Current_Arp]
    type = Current
    potential = potential
    density_log = Arp
    variable = Current_Arp
    art_diff = false
    block = 0
    position_units = ${dom0Scale}
  [../]
  [./Current_OHm]
    block = 1
    type = Current
    potential = potential
    density_log = OHm
    variable = Current_OHm
    art_diff = false
    position_units = ${dom1Scale}
  [../]
  [./tot_flux_OHm]
    block = 1
    type = TotalFlux
    potential = potential
    density_log = OHm
    variable = tot_flux_OHm
  [../]
  [./EFieldAdvAux_em]
    type = EFieldAdvAux
    potential = potential
    density_log = em
    variable = EFieldAdvAux_em
    block = 0
  [../]
  [./DiffusiveFlux_em]
    type = DiffusiveFlux
    density_log = em
    variable = DiffusiveFlux_em
    block = 0
  [../]
  [./EFieldAdvAux_emliq]
    type = EFieldAdvAux
    potential = potential
    density_log = emliq
    variable = EFieldAdvAux_emliq
    block = 1
  [../]
  [./DiffusiveFlux_emliq]
    type = DiffusiveFlux
    density_log = emliq
    variable = DiffusiveFlux_emliq
    block = 1
  [../]
[]

[InterfaceKernels]
```

```
  [./em_advection]
    type = InterfaceAdvection
    mean_en_neighbor = mean_en
    potential_neighbor = potential
    neighbor_var = em
    variable = emliq
    boundary = master1_interface
    position_units = ${dom1Scale}
    neighbor_position_units = ${dom0Scale}
  [../]
  [./em_diffusion]
    type = InterfaceLogDiffusionElectrons
    mean_en_neighbor = mean_en
    neighbor_var = em
    variable = emliq
    boundary = master1_interface
    position_units = ${dom1Scale}
    neighbor_position_units = ${dom0Scale}
  [../]
[]

[BCs]
  [./potential_left]
    type = NeumannCircuitVoltageMoles_KV
    variable = potential
    boundary = left
    function = potential_bc_func
    ip = Arp
    data_provider = data_provider
    em = em
    mean_en = mean_en
    r = 0
    position_units = ${dom0Scale}
  [../]
  [./potential_dirichlet_right]
    type = DirichletBC
    variable = potential
    boundary = right
    value = 0
  [../]
  [./em_physical_right]
    type = HagelaarElectronBC
    variable = em
    boundary = 'master0_interface'
    potential = potential
    ip = Arp
    mean_en = mean_en
    r = 0
    position_units = ${dom0Scale}
  [../]
  # [./em_physical_right]
  #   type = MatchedValueLogBC
  #   variable = em
  #   boundary = 'master0_interface'
  #   v = emliq
  #   H = 1e7
  # [../]
  [./Arp_physical_right]
    type = HagelaarIonBC
    variable = Arp
    boundary = 'master0_interface'
    potential = potential
    r = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_physical_right]
    type = HagelaarEnergyBC
    variable = mean_en
    boundary = 'master0_interface'
    potential = potential
    em = em
    ip = Arp
    r = 0
    position_units = ${dom0Scale}
  [../]
  [./em_physical_left]
    type = HagelaarElectronBC
    variable = em
    boundary = 'left'
    potential = potential
    ip = Arp
    mean_en = mean_en
    r = 0
    position_units = ${dom0Scale}
  [../]
  [./Arp_physical_left]
```

```
    type = HagelaarIonBC
    variable = Arp
    boundary = 'left'
    potential = potential
    r = 0
    position_units = ${dom0Scale}
  [../]
  [./mean_en_physical_left]
    type = HagelaarEnergyBC
    variable = mean_en
    boundary = 'left'
    potential = potential
    em = em
    ip = Arp
    r = 0
    position_units = ${dom0Scale}
  [../]
  [./emliq_right]
    type = DCIonBC
    variable = emliq
    boundary = right
    potential = potential
    position_units = ${dom1Scale}
  [../]
  [./OHm_physical]
    type = DCIonBC
    variable = OHm
    boundary = 'right'
    potential = potential
    position_units = ${dom1Scale}
  [../]
[]

[ICs]
  [./em_ic]
    type = ConstantIC
    variable = em
    value = -26
    block = 0
  [../]
  [./emliq_ic]
    type = ConstantIC
    variable = emliq
    value = -21
    block = 1
  [../]
  [./Arp_ic]
    type = ConstantIC
    variable = Arp
    value = -26
    block = 0
  [../]
  [./mean_en_ic]
    type = ConstantIC
    variable = mean_en
    value = -25
    block = 0
  [../]
  # [./potential_ic]
  #   type = ConstantIC
  #   variable = potential
  #   value = 0
  # [../]
  [./potential_ic]
    type = FunctionIC
    variable = potential
    function = potential_ic_func
  [../]
  [./OHm_ic]
    type = ConstantIC
    variable = OHm
    value = -15.6
    block = 1
  [../]
  # [./em_ic]
  #   type = RandomIC
  #   variable = em
  #   block = 0
  # [../]
  # [./emliq_ic]
  #   type = RandomIC
  #   variable = emliq
  #   block = 1
  # [../]
  # [./Arp_ic]
  #   type = RandomIC
  #    variable = Arp
  #    block = 0
  # [../]
  # [./mean_en_ic]
  #    type = RandomIC
  #    variable = mean_en
  #    block = 0
  # [../]
  # [./potential_ic]
  #    type = RandomIC
  #    variable = potential
  # [../]
  # [./OHm_ic]
  #    type = RandomIC
  #    variable = OHm
  #    block = 1
  # [../]
[]

[Functions]
  [./potential_bc_func]
    type = ParsedFunction
    # value = '1.25*tanh(1e6*t)'
    value = 1.25
  [../]
  [./potential_ic_func]
    type = ParsedFunction
    value = '-1.25 * (1.0001e-3 - x)'
  [../]
[]

[Materials]
  [./gas_block]
    type = Gas
    interp_trans_coeffs = true
    interp_elastic_coeff = true
    em = em
    potential = potential
    ip = Arp
    mean_en = mean_en
    block = 0
 [../]
 [./water_block]
```

```
    type = Water
    block = 1
    potential = potential
 [../]
# [./jac]
#    type = JacMat
#    mean_en = mean_en
#    em = em
#    emliq = emliq
#    block = '0 1'
#  [../]
[]
```

Listing A.1 Exemplary Zapdos input file

## A.2 Python Methods

A couple of useful python methods we developed are described below. Interested readers can navigate to the python directory of [137] for more python scripts (at various levels of code maturity).

### A.2.1 load_data method

The load_data method shown in listing A.2 handles a raw Exodus data file output by Zapdos. It uses reader and writer modules from Paraview's [129] python interface to write solution variable data from the last simulation time step to a CSV file. Then numpy is used to load node and element data into separate arrays for later processing. To run the load_data method, the user specifies the following arguments, all of which are lists and must have the same size (elements correspond to a particular job):

- short_names (type list of strings): The strings passed in this list should summarize the simulation jobs. They will be used as the label strings later when solution variables like electron density, potential, etc. are plotted
- labels (type list of strings): List of colors that corresponds to the list of jobs that will be used in coloring variable plots, e.g. labels = ['blue','red'] would mean that job1 plots would be blue and job2 plots would be red

- mesh_struct (type list of strings): The elements in this list must either equal 'scaled' or 'physical'. 'Physical' means that the simulation did not use a scaled mesh.
- styles (type list of strings): Options for elements in this list include 'solid', 'dashed', 'dashdot', etc. A job with a style of 'dashed' will have dashed line plots.

```
def load_data(short_names, labels, mesh_struct, styles):
    global job_names, name_dict, cellGasData, cellLiquidData, pointGasData,
    pointLiquidData, label_dict, style_dict, mesh_dict
    data = OrderedDict()
    cellData = OrderedDict()

    name_dict = {x:y for x,y in zip(job_names, short_names)}
    label_dict = {x:y for x,y in zip(job_names, labels)}
    style_dict = {x:y for x,y in zip(job_names, styles)}
    mesh_dict = {x:y for x,y in zip(job_names, mesh_struct)}

    index = 0
    GasElemMax = 0
    path = "/home/lindsayad/gdrive/MooseOutput/"
    for job in job_names:
        file_sans_ext = path + job + "_gold_out"
        inp = file_sans_ext + ".e"
        out = file_sans_ext + ".csv"

        reader = ExodusIIReader(FileName=inp)
        tsteps = reader.TimestepValues
        writer = CreateWriter(out, reader)
        writer.Precision = 16
        writer.UpdatePipeline(time=tsteps[len(tsteps)-1])
        del writer

        for i in range(2,6):
            os.remove(file_sans_ext + str(i) + ".csv")

        new_inp0 = file_sans_ext + "0.csv"
        data[job] = np.genfromtxt(new_inp0,delimiter=',', names=True)
        pointGasData[job] = data[job]

        # Use for coupled gas-liquid simulations
        new_inp1 = file_sans_ext + "1.csv"
        data1 = np.genfromtxt(new_inp1, delimiter=',', names=True)
        pointLiquidData[job] = data1
        data[job] = np.concatenate((data[job],data1), axis=0)

        writer = CreateWriter(out, reader)
        writer.FieldAssociation = "Cells"
        writer.Precision = 16
        writer.UpdatePipeline(time=tsteps[len(tsteps)-1])
        del writer

        for i in range(2,6):
            os.remove(file_sans_ext + str(i) + ".csv")

        new_inp0 = file_sans_ext + "0.csv"
        cellData[job] = np.genfromtxt(new_inp0,delimiter=',', names=True)
        cellGasData[job] = cellData[job]
        if index == 0:
            GasElemMax = np.amax(cellData[job]['GlobalElementId'])

        # Use for coupled gas-liquid simulations
        new_inp1 = file_sans_ext + "1.csv"
        data1 = np.genfromtxt(new_inp1, delimiter=',', names=True)
        cellData[job] = np.concatenate((cellData[job],data1), axis=0)
        cellLiquidData[job] = data1
```

Listing A.2 Python method for turning raw Exodus file data into numpy arrays using Paraview's [129] python interface

### A.2.2 Plotting methods

The following two methods are for plotting elemental and nodal variables respectively. The user specifies the following parameters:

- save (type bool): Whether to output an eps figure
- variable (type str): The variable to be plotted, e.g. the electron density, potential, electron temperature, etc.
- pos_scaling (type float): The amount of scaling of the mesh. This option will be deprecated now that the Position AuxKernel has been updated

- ylabel (type str): Label to apply to the y-axis
- tight_plot (type bool): Whether to tighten the figure area. In general this should be True; True generally prevents cutting off of x- and y-axis labels that may occur if False
- xticks (type list of floats): Where to apply tick marks on the axes
- xticklabels (type list of strings): List of tick labels being applied to the ticks passed to xticks
- xlabel (type str): Label to apply to the x-axis
- xmin (type float): Optional argument. Specifies x-axis minimum
- xmax (type float): Optional argument. Specifies x-axis maximum
- ymin (type float): Optional argument. Specifies y-axis minimum
- ymax (type float): Optional argument. Specifies y-axis maximum
- yscale (type str): Optional argument. Specifies the format of the y-axis. Can pass 'log' to specify logarithmic scaling
- save_string (type str): Optional argument. If save=True, the user should specify a string here to create unique name for the figure saved. Otherwise the string 'dummy' will be used.

The elemental variable plotting method is given in listing A.3.

```python
def cell_gas_generic(save, variable, pos_scaling, ylabel, tight_plot, xticks,
    xticklabels, xlabel, xmin=None, xmax=None, ymin=None, ymax=None, yscale=None,
     save_string="dummy"):
    fig = plt.figure()
    ax1 = plt.subplot(111)
    for job in job_names:
        if mesh_dict[job] == "phys":
            ax1.plot(cellGasData[job]['x'], cellGasData[job][variable], color =
    label_dict[job], linestyle = style_dict[job], label = name_dict[job],
    linewidth=2)
        elif mesh_dict[job] == "scaled":
            ax1.plot(cellGasData[job]['x'] / pos_scaling, cellGasData[job][
    variable], color = label_dict[job], linestyle = style_dict[job], label =
    name_dict[job], linewidth=2)
    ax1.legend(loc='best', fontsize = 16)
    ax1.set_xticks(xticks)
    ax1.set_xticklabels(xticklabels)
    ax1.set_xlabel(xlabel)
    ax1.set_ylabel(ylabel)
    if xmin is not None:
        ax1.set_xlim(left=xmin)
    if xmax is not None:
        ax1.set_xlim(right=xmax)
    if ymin is not None:
        ax1.set_ylim(bottom=ymin)
    if ymax is not None:
        ax1.set_ylim(top=ymax)
    if yscale is not None:
        ax1.set_yscale(yscale)
    if tight_plot:
        fig.tight_layout()
    if save:
        fig.savefig('/home/lindsayad/Pictures/' + save_string + '_' + variable +
    '.eps', format='eps')
    plt.show()
```

Listing A.3 Generic elemental variable plotting method. Argument description given in appendix A.2.2

The nodal variable plotting method is given in listing A.4.

```
def point_gas_generic(save, variable, pos_scaling, ylabel, tight_plot, xticks,
    xticklabels, xlabel, xmin=None, xmax=None, ymin=None, ymax=None, yscale=None,
     save_string="dummy"):
    fig = plt.figure()
    ax1 = plt.subplot(111)
    for job in job_names:
        if mesh_dict[job] == "phys":
            ax1.plot(pointGasData[job]['Points0'], pointGasData[job][variable],
    color = label_dict[job], linestyle = style_dict[job], label = name_dict[job],
     linewidth=2)
        elif mesh_dict[job] == "scaled":
            ax1.plot(pointGasData[job]['Points0'] / pos_scaling, pointGasData[job
    ][variable], color = label_dict[job], linestyle = style_dict[job], label =
    name_dict[job], linewidth=2)
    ax1.legend(loc='best', fontsize = 16)
    ax1.set_xticks(xticks)
    ax1.set_xticklabels(xticklabels)
    ax1.set_xlabel(xlabel)
    ax1.set_ylabel(ylabel)
    if xmin is not None:
        ax1.set_xlim(left=xmin)
    if xmax is not None:
        ax1.set_xlim(right=xmax)
    if ymin is not None:
        ax1.set_ylim(bottom=ymin)
    if ymax is not None:
        ax1.set_ylim(top=ymax)
    if yscale is not None:
        ax1.set_yscale(yscale)
    if tight_plot:
        fig.tight_layout()
    if save:
        fig.savefig('/home/lindsayad/Pictures/' + save_string + '_' + variable +
    '.eps', format='eps')
    plt.show()
```

Listing A.4 Generic nodal variable plotting method. Argument description given in appendix A.2.2