

**Investigation of the Redistribution of Kinetic Energy in a  
Microgravity Complex (Dusty) Plasma**

by

Lori Christina Scott McCabe

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

Auburn, Alabama  
August 6, 2022

Keywords: Dusty Plasma, Low-Temperature Plasma, Microgravity,  
Energy Dissipation, Plasma Screening Length

Copyright 2022 by Lori McCabe

Approved by

Edward Thomas, Jr., Professor of Physics and  
Interim Dean of the College of Science and Mathematics  
Jeremiah Williams, Professor and Chair of Physics, Wittenberg University  
David Ennis, Associate Professor of Physics  
Stuart Loch, Professor of Physics

## **Abstract**

In the presence of gravity, the micron-sized charged dust particles in a complex plasma are compressed to thin layers, but under the microgravity conditions of the Plasma Kristall-4 (PK-4) experiment on the International Space Station (ISS), the particles fill the plasma, and we can investigate properties of a three-dimensional multi-particle system. This dissertation examines the change in the spatial ordering and thermal state of the particle system created when dust particles are stopped by periodic oscillations of the electric field, known as polarity switching, in a dc glow discharge plasma.

Data from the ISS is compared against experiments performed using a ground reference version of PK-4 and numerical MD code simulations. Initial results show substantive differences in the velocity distribution functions between experiments on the ground and in microgravity. The dust cloud in microgravity gains thermal energy at the application of polarity switching, a periodic oscillation of the electric field. This change in energy is seen in multiple plasma conditions (power, pressure) and whenever there is a change in the electric field direction, not just when polarity switching is applied. Simulation results suggest that this may be due to a modification in the dust screening length at the onset of polarity switching. Experimental measurements and simulations show that an extended time (much greater than the Epstein drag decay) is required to dissipate this energy back into the plasma. This larger timescale for dissipation than compared to Epstein drag is likely due to the ability of the interparticle structural energy to serve as an energy sink for the dust cloud.

## **Acknowledgments**

To my advisor, Dr. Edward Thomas Jr., thank you for everything throughout my graduate school experience. From the first email you sent me after I passed the GDEs saying “get your passport ready” I knew I was in for a great research experience. Even when you cause code to randomly malfunction during meetings with the advisor effect, I appreciate all of the guidance and support you provided the last several years. This would not have been possible without your help and dedication despite your ever-increasingly busy schedule.

To Dr. Jeremiah Williams, Dr. Saikat Chakraborty Thakur, and Dr. Uwe Konopka, thank you for your mentorship as secondary advisors through this project. Thank you to my committee members, Dr. Stuart Loch and Dr. David Ennis for your time, feedback, and insightful questions. Thank you to Dr. Lorin Matthews and Dr. Truell Hyde from Baylor University for getting me into dusty plasmas as an undergraduate student and continuing to be great collaborators and mentors through the years. You all helped guide and support me through my project and grow as a scientist and person.

To Dr. Misha Pustyl'nik, Dr. Hubertus Thomas, and the science team at DLR, Dr. Astrid Orr ESA manager for PK-4, and the entire team at CADMOS for quite literally making these experiments possible. The planning process for a campaign is extensive, and we appreciate the opportunities for letting outside scientists propose and use the experiment.

Thank you to all my physics friends, this grad school experience would not be the same without you. Thank you to my lab mates of PSL/MPRL past and present, Steve Williams, Jared

Powell, Dr. Misha Mickinlay, Dylan Funk, and Brandon Doyle, Dr. Taylor Hall and Dr. Spencer Leblanc, for always helping brainstorm ideas and enjoying our conference travels together. Thank you to our lab managers Darrick Artis and Cameron Royer for keeping us in organized and in line and the infinite troubleshooting skills. And thank you to Dr. Ricky Strom, Spenser Burrows, and Brady Unzicker for the infinite laughs from memes when we need a break at the office.

And finally thank you to my family, my dad David and mom Debra. While we still don't know where the physics brain came from, you always supported me with love and cheerful nods when you don't know what I am talking about, and yet you still always checked in on how my research was going. My fluffy coworkers in this work-from-home world, Allie and Maggie, who are the sweetest zoom interrupters making sure their presence known during every meeting. And finally, thank you to my incredibly supportive husband, Dr. Jacob McCabe. Both of us in grad school (long-distance) was quite a task but I am so glad we were there for each other every step of the way. All of the effort that went into this work would not have been possible without you by my side. Thank you.



## Table of Contents

Abstract .....	ii
Acknowledgments .....	iii
List of Figures .....	x
List of Tables .....	xvi
List of Abbreviations and Variables .....	xvii
<b>Chapter 1 : Introduction .....</b>	<b>1</b>
<b>1.1. Plasmas and Dusty Plasmas .....</b>	<b>2</b>
<b>1.2. Thermal Properties of Dusty Plasmas .....</b>	<b>4</b>
1.2.1. Maxwell-Boltzmann Distributions.....	6
<b>1.3. Charging of Dust Grains .....</b>	<b>7</b>
<b>1.4. Select Forces in a Dusty Plasma .....</b>	<b>10</b>
1.4.1. Interaction Forces.....	10
1.4.2. Debye Length.....	11
1.4.3. Epstein Drag.....	13
1.4.4. Electric Forces .....	14
1.4.5. Langevin Heater .....	15
1.4.6. System of Equations .....	16
<b>1.5. Outline of Dissertation .....</b>	<b>16</b>
<b>Chapter 2 : Experimental Setup and Analysis Techniques .....</b>	<b>18</b>

2.1. PK-4 Apparatus.....	18
2.1.1. PK-4 Chamber Description.....	20
2.1.2. Plasma Conditions .....	21
2.1.3. Camera Settings .....	22
2.1.4. Polarity Switching.....	24
2.2. PIV Analysis Techniques.....	27
2.2.1. Processing and Settings.....	28
2.2.2. Analysis Techniques .....	33
2.2.3. Validation for PK-4: DLR report.....	35
2.3. YOAK $\mu$ M- MD Simulation Code .....	38
<b>Chapter 3 : Experimental Results .....</b>	<b>41</b>
3.1. Experiment Development .....	41
3.2. Campaign 7 Results: Complete Analysis of a Single Dataset .....	47
3.2.1. Histogram Benchmarking .....	51
3.2.2. Subdividing into Regions.....	58
3.3. All Datasets from Campaign 7.....	61
3.3.1. Regions Analysis for Additional Datasets .....	63
3.4. Other Experimental Insights .....	66
3.4.1. Campaign 12 Reinjections .....	66
3.4.2. Campaign 7 Polarity Stepdown .....	69
3.5. Experiment Discussion .....	72
<b>Chapter 4 : Numeric Model Simulations.....</b>	<b>75</b>
4.1. YOAK $\mu$ M Step-by-step Replication of PK-4.....	76
4.1.1. Initialize Cloud.....	81

4.1.2. Injecting the Dust Particles .....	82
<b>4.2. Physics of Heating and Dissipation in the Dust Cloud .....</b>	<b>83</b>
<b>4.3. Validation of YOAK<math>\mu</math>M results .....</b>	<b>86</b>
4.3.1. Particle Positions and Velocities.....	88
4.3.2. Particle Energies .....	92
4.3.3. Cloud Size Validation .....	97
<b>4.4. MD Simulation Results.....</b>	<b>100</b>
<b>4.5. Simulation Discussion .....</b>	<b>103</b>
<b>Chapter 5 : Conclusions and Future Work .....</b>	<b>105</b>
<b>5.1. Conclusions.....</b>	<b>108</b>
<b>5.2. Future Work.....</b>	<b>109</b>
5.2.1. “Hook” Paths .....	109
5.2.2. Campaign 12 Full Analysis.....	110
5.2.3. Campaign 9- “Capture” to “Flowing” Reverse Experiment.....	111
5.2.4. All PO3 Data.....	113
5.2.5. YOAK $\mu$ M.....	115
References.....	117
Appendix A : PK-4 Documentation.....	126
A.1. Campaign 7 .....	126
A.1.1. Proposal.....	127
A.1.2. Flowchart .....	131
A.1.3. Lori’s Excel Timing Sheet .....	143
A.2. Campaign 12 .....	145
A.2.1. Proposal.....	146

A.2.2. Flowchart .....	150
A.3. Other Campaign Descriptions .....	162
A.3.1. Campaign 9 .....	162
A.3.2. Campaign 14 .....	181
A.3.3. Campaign 15 .....	181
A.4. Experiment Proposal Process and Timeline.....	182
Appendix B : Analysis Code Files.....	183
B.1. YOAK $\mu$ M Code .....	183
B.1.1. Driver.hpp .....	184
B.1.2. Driver.cpp.....	185
B.1.3. Basic_forces.hpp .....	193
B.1.4. Interaction_forces.hpp.....	215
B.1.5. Dust_state.hpp .....	228
B.1.6. Io_observers.hpp .....	231
B.1.7. Configure.hpp.....	232
B.1.8. System.hpp.....	233
B.1.9. Vector.hpp .....	234
B.2. Igor- Macro File .....	241
B.3. PIV Settings.....	242
B.4. MATLAB- Processing Codes .....	248
B.4.1. Camera Sample Data .....	248
B.4.2. Extract Data.....	249
B.4.3. Get Data.....	250
B.4.4. Single Fit Camera 2.....	252

B.4.5. Other Analysis Blocks.....	251
B.5. Mathematica- PK-4 All Calculations File.....	256

## List of Figures

Figure 1-1: Lab plasmas (a) without dust, showing the plasma glow, and (b) with dust and a green illumination laser to see the cloud. ....	3
Figure 2-1: Schematic of PK-4 experiment on board the International Space Station. ....	20
Figure 2-2: a) The field of view of a microgravity injection at 70 fps, with a vertical width of 480 pixels. Injection from Campaign 7, Injection 3. B) the field of view of a microgravity injection at 140 fps with a vertical width of 120 pixels. Injection from Campaign 12, Injection 2. ....	22
Figure 2-3: A sample VM3 frame from Campaign 7. ....	23
Figure 2-4: A sample microgravity PO1 frame during a) the injection process, or “flow” where the dust particles appear as streaks across many pixels and b) once the dust cloud is captured by polarity switching, or “capture”, where the dust particles appear more spherical. ....	25
Figure 2-5: Illustration of dc current for applying polarity switching to the dust cloud. ....	26
Figure 2-6: String formation in a microgravity PK-4 environment. ....	27
Figure 2-7: DaVis software vector processing window. ....	29
Figure 2-8: A cartoon of the PIV analysis technique. ....	29
Figure 2-9: A sample Ground-based PIV resultant vector field, overlapping the original data frame, to illustrate that the waves in the bottom of the field-of-view do not yield (reasonable) vectors when using the “flow” settings. ....	31

Figure 2-10: A sample total vector field reconstruction. ....	32
Figure 2-11: A sample resultant PIV frame of (a) total vectors returned and (b) representative vector field with 1/8 vector density for easier display. This is a frame from the injection of the $p = 0.6$ mBar, $I = 0.7$ mA microgravity dataset. ....	33
Figure 2-12: Histograms (red) of the (a) x-direction and (b) z-direction components of velocity for the sample PIV frame in Figure 2-11. ....	34
Figure 2-13: Resultant vector fields of a) square bins and b) 4:1 horizontal elliptical bins for ground-based data recorded at 70 fps.....	35
Figure 2-14: Total number of vectors returned by frame number for the square cells and the 4:1 elliptical regions as a function of time (image number). ....	36
Figure 2-15: Comparison of the average displacement as a function of time (image number) for 60, 70, and 80 fps. ....	37
Figure 2-16: Comparison of the average horizontal velocity (scaled, pixels/s) as a function of time. ....	37
Figure 3-1: Initial ground-based results taken in 2017. ....	42
Figure 3-2: FFT of the x-direction drift velocity as a function of time, for the same dataset as Figure 3-1, at polarity switching 100 Hz. ....	43
Figure 3-3: FFT of the x-direction drift velocity as a function of time for a polarity switching frequency of 500 Hz. There are no longer any significant peaks. ....	44
Figure 3-4: Comparison of velocity and dust kinetic temperature measurements from ground and microgravity PK-4 experiments for camera 1 (red). ....	47
Figure 3-5: Comparison of velocity and dust kinetic temperature measurements from ground and microgravity PK-4 experiments for camera 2 (blue). ....	48

Figure 3-6: A sample ground-based PIV resultant vector field, overlapping the original data frame, to illustrate that the waves in the bottom of the field-of-view do not yield (reasonable) vectors when using the “flow” settings. .... 50

Figure 3-7: Sample histograms of the  $v_x$  components yielded from PIV from ground (left) and flight (right) experiments pre-polarity switching. .... 52

Figure 3-8: Sample injection segment frames’ histograms (red) for the x-direction velocity components which yield “bad” MB fits (black). .... 53

Figure 3-9: Sample histograms of the  $v_x$  components yielded from PIV from ground (left) and flight (right) experiments post-polarity switching. .... 54

Figure 3-10: Invalid fit of a histogram during the captured segment of the microgravity  $p = 0.6$  mBar,  $I = 0.7$  mA dataset. .... 55

Figure 3-11: Improved fit after a redo of the same frame for the histogram in Figure 3-10. .... 56

Figure 3-12: Qualitative PIV analysis technique flowchart. We focus on confirmed the results of histograms are good fits before we do our further analysis. .... 57

Figure 3-13: A resultant velocity vector field from the  $p = 0.6$  mBar,  $I = 0.7$  mA dataset, combining PO1 and PO2 for a full Field of View. .... 58

Figure 3-14: a) The same dust cloud temperature data from camera 1 presented in Figure 3 b and c, reorganized to have  $T_x$  on the top, and  $T_z$  on the bottom. b) A sample frame divided into 4 regions, and the boxes marking the regions match the color to the region’s temperature data, directly below. .... 59

Figure 3-15: using the same regions shown in Figure 3-14, we can look at a single frame’s region histograms to further show the variation in regions’ results. .... 60

Figure 3-16: The whole-field x-direction temperatures for data from all 9 parameter spaces and both cameras. .... 62



Figure 3-17: Sample vector fields for before polarity switching (top) and after capture (bottom).....	64
Figure 3-18: Histograms in the captured frames' (bottom) 10 regions in Figure 3-17 for the x-direction (left) and z-direction (right). The numbers 1-10 for the regions in this figure correspond to the numbers at the top of Figure 3-17. ....	65
Figure 3-19: Campaign 12 whole field analysis for the reinjection turn around (flow from right to left) segment in microgravity. ....	67
Figure 3-20: Campaign 7 reinjection segment whole field. ....	68
Figure 3-21: $T_x$ and $T_z$ as a function of time for the PS stepdown section of the Campaign 7 microgravity experiment. ....	69
Figure 3-22: A closer look at the PS 500 Hz to 250 Hz stepdown heating and dissipation in the x-direction from Figure 3-21.....	71
Figure 4-1: Evolution of the dust cloud temperature during the preparation segments for the simulation, and one example "settling" segment. ....	77
Figure 4-2: The evolution of the average (mean) drift velocity during the preparation steps for the simulation, and one example "settling" segment. ....	78
Figure 4-3: The same settling data as shown in Figure 4-2 (yellow), zoomed in to show the oscillation of the drift velocity in the dust cloud. ....	79
Figure 4-4: Sample dust cloud structure of 1000 particles after initialization and allowing the particles to interact for 0.75s. ....	81
Figure 4-5: The simulated dust cloud's motion during the injection segment, equivalent to the injection of the dust cloud in PK-4. ....	82
Figure 4-6: A sample dust cloud experiencing a Coulomb explosion. ....	85
Figure 4-7: A YOAKUM cloud with the particle numbers (indices) labeled.....	86

Figure 4-8: A closer look at the 4 reference particles selected. ....	88
Figure 4-9: Nearest neighbor differences relative to particle #40. ....	89
Figure 4-10: Difference between #40 and the furthest particle, #188. ....	91
Figure 4-11: Velocity phase-space evolution during the dissipation segment of the simulation.....	92
Figure 4-12: Average dust cloud interparticle distance (left) and average dust particle's velocities (right) vs. time. ....	93
Figure 4-13: Total potential energy and kinetic energy vs. time for the dust cloud in the YOAK $\mu$ M settling segment, $\lambda = 1.75 \mu\text{m}$ .....	94
Figure 4-14: Same energy data as Figure 4-13, but focusing on $t = 1.775 - 2 \text{ s}$ , with modified kinetic energy vertical axes limits to highlight the decay in both types of energy. Both values decay with time. ....	95
Figure 4-15: Comparison of a representative potential energy (from the average nearest neighbor distance) and the dust cloud temperature vs. time. ....	96
Figure 4-16: Dust cloud temperature evolution, for 1000, 2000, and 5000 particles during the three initialization segments.....	98
Figure 4-17: Comparison between 1000 particle 2D clouds and 1000 particle 3D clouds during the initialization process. The green, 2D cloud is the same temperature evolution shown in Figure 4-1, and the vertical axes is a smaller range to illustrate the subtle difference between the two clouds in the simulation. ....	99
Figure 4-18: The change in dust cloud temperature after the application of polarity switching (normalized to $t = 0 \text{ s}$ ) for varying screening lengths.....	101
Figure 4-19: The estimated characterization of dust screening length as a function of time at the application of polarity switching.....	102

Figure 5-1: Hook-like features that occur at the application of polarity switching in the ground experiment. ....	109
Figure 5-2: Using a square wave modulation on the manipulation laser, we can induce a change in the dust temperature in the X direction. ....	113
Figure 5-3: A line segment through the three views of PO3, in which we take pixel intensity values which correlate to the plasma glow strength. ....	114
Figure 5-4: Three line-integrated intensity traces and a line ratio of the filter views from VM3. ....	114
Appendix Figure 7: Vector calculation parameters. This is where the multi-pass settings are located. ....	246

## List of Tables

Table 3-1: Values or range of values for various plasma characteristics in our Campaign 7 experiment.....	45
Table 3-2: Summary of which plasma operating conditions show heating at the application of polarity switching in at least one camera in the microgravity campaign 7 experiment, shown in Figure 3-16. ....	63
Table 3-3: Summary of which plasma operating conditions show heating during a decrease of polarity switching frequency in at least one camera in the microgravity campaign 7 experiment.....	72
Table 3-4: Electron Debye length calculated based on plasma operating conditions. The boxes represent the results from Table 3-2 for the capture data, and the circles represent the results from Table 3-3 for the PS stepdown data. ....	73
Table 5-1: Injection dust cloud parameters for Campaign 12.....	111

## List of Abbreviations and Variables

CASPER	Center for Astrophysics, Space Physics and Engineering Research at Baylor University
DLR	German Aerospace Center, Deutsches Zentrum für Luft- und Raumfahrt
$e$	Elementary charge of an electron ( $1.602 \times 10^{-19}$ C)
ESA	European Space Agency
FFT	Fast Fourier Transform
FST	Facility Science Team, the group of scientists allowed to propose experiments on PK-4
FOV	Field of View
ISS	International Space Station
JIHT	Joint Institute for High Temperatures (Russia)
MB	Maxwell-Boltzmann Distribution
MD	Molecular Dynamics Simulation Code
MF	Melamine Formaldehyde ( $\rho = 1510$ kg/m <sup>3</sup> )
MPRL	Magnetized Plasma Research Laboratory, Auburn University
MSC	NASA Marshall Space Center
OML	Orbital Motion Limited Theory
PIV	Particle Image Velocimetry
PK-4	Plasmakristall-4 Experiment

PO 1,2,3	Particle Observation Camera 1, 2 and 3
PTV	Particle Tracking Velocimetry
VM 1,2,3	The file names for data from PO 1, 2, and 3, respectively
YOAK $\mu$ M	Yukawa Ordered and Kristallized Microgravity Model
Z <sub>d</sub>	Number of elementary charges on a dust particle

## **Chapter 1: Introduction**

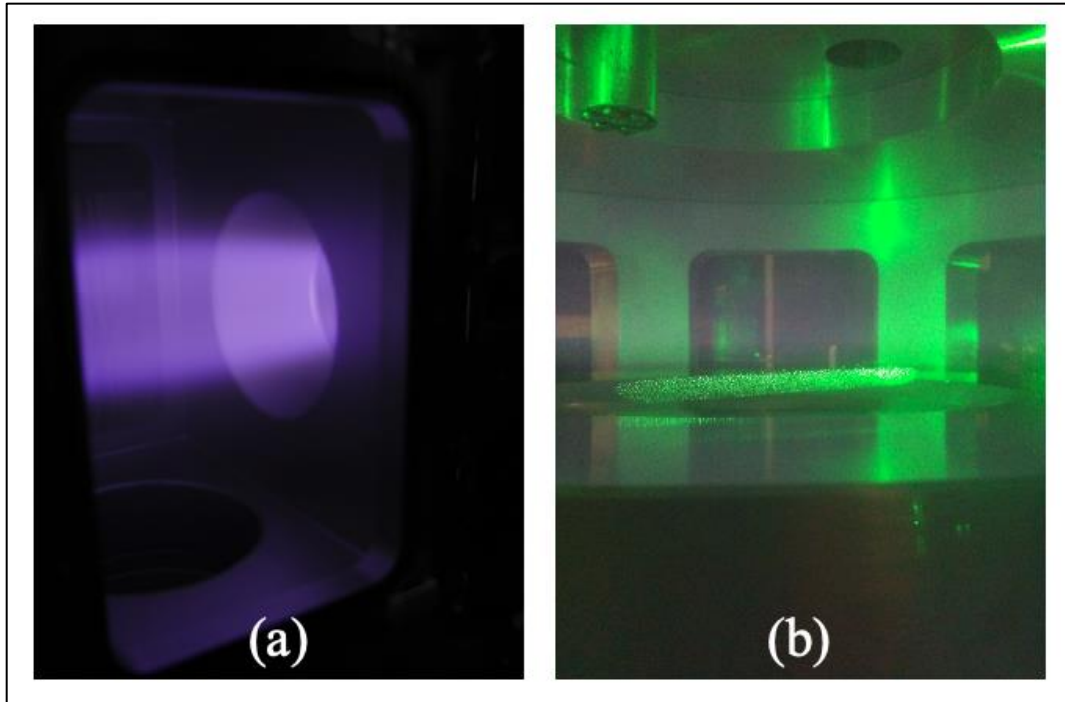
This dissertation investigates the differences in the thermal properties in a dusty plasma when a change in the background electric field modifies the flow or structural properties of the dust cloud. These studies use a direct comparison between a ground-based, gravity-dominated environment and a microgravity environment on the International Space Station. These experimental studies show that when the electric field changes we observe an instantaneous change in the dust kinetic temperature, and under the correct plasma conditions, this change is followed by an extended decay of the energy in microgravity-based experiments that is generally not seen in ground-based studies.

To further investigate the interactions between plasma and dust particles, we use a molecular dynamics (MD) simulation. Our simulation results support this extended timescale needed for the dissipation process. The simulations suggest that there is a modification in the effective screening length of the dust particles in the plasma environment on the microscopic scale and an ability to tap into the structural energy of the dust cloud. The combination of the changing screening length and accessing the structural energy creates an energy reservoir that is effectively a damped net dissipation of the thermal kinetic energy in a dust cloud system. The discussions presented in this dissertation will use a combination of experimental and numerical results to propose a common description for heating and extended dissipation in the dust cloud.

## 1.1. Plasmas and Dusty Plasmas

A plasma, typically referred to as the fourth state of matter, is created when there is enough energy (usually by electric field) to ionize a gaseous state, illustrated in Figure 1-1(a). This results in three distinct plasma components- the electrons stripped from the outer orbitals, ions, and the remaining neutral gas atoms. Because a plasma is a type of charged fluid, internal electromagnetic forces can arise in these systems, and they can be influenced by externally imposed electric and magnetic fields. A complex, or dusty, plasma is considered complex because we add a fourth charged component into the system, micro- or nano- sized particles, seen in Figure 1-1(b). In many experiments, these particles are typically spherical shaped grains introduced manually into the plasma which levitate in the plasma sheath region [1–4]. However, in the presence of chemically active plasmas, the dust particles can also be directly produced (i.e. grown or self-formed) inside a plasma [5–7]. Because the dust particles are more massive than the surrounding plasma particles (electrons, ions and neutral atoms), they will be subject to gravitational forces in ground-based experiments, often resulting in the dust cloud forming in a two-dimensional sheet. As a result, it is often the competition between gravity, neutral drag, and various electromagnetic forces on the dust particles that dominate many of the phenomena that we observe in the lab. Dusty plasmas in microgravity are not confined to levitate in just the plasma sheath region, they can expand into the bulk plasma region, creating large three-dimensional clouds. Therefore, it is of great scientific value to study the physics of dusty plasmas without the dominating role of gravity - thus the motivation to perform dusty plasma experiments under microgravity conditions.





**Figure 1-1: Lab plasmas (a) without dust, showing the plasma glow, and (b) with dust and a green illumination laser to see the cloud. Both of these experiments are part of the Auburn MPRL lab group, (a) ALEXIS (Auburn Linear EXperiment for Instability Studies), and (b) MDPX (Magnetized Dusty Plasma eXperiment).**

Beyond performing fundamental studies of dusty plasmas, the presence of charged dust in plasmas can occur in a wide variety of natural and human-made settings. Dusty plasmas exist in astrophysical settings, such as Saturn’s rings or even around blackholes [8–10]. Beside the astrophysical applications listed, there are other applications for dusty plasmas such as microchip etching for semiconductors and nanotechnology [11,12] and dust in a fusion plasma environment from wall ablation [13–15]. In these cases, dusty plasmas can be a hinderance to the primary goals. But by studying dusty plasmas, we can determine ways to improve the dust interference with these systems.

Dusty plasma is often used to investigate the fundamental physics in a plasma system. By using a dusty plasma, we can investigate waves, structures, ordering, and more fundamental

phenomena [16–21]. The plasma becomes a non-invasive diagnostic tool to the plasma environment and creates new fundamental physics from the interactions of the four-components in the plasma. Dusty plasmas also have soft matter applications, and can be used to investigate fluids or structural systems [22–25].

The work reported in this dissertation is focused on understanding the conversion of flow kinetic energy to thermal energy in a microgravity dusty plasma experiment. These studies are performed using the Plasmakristall-4 (PK-4) microgravity laboratory - both the ground-based PK-4 science reference module and the flight-based PK-4 facility that is on the International Space Station. The PK-4 facility produces linear, flowing dusty plasmas using a direct current (dc) generated plasma system with the application of oscillating electric fields to trap the dust particles - a technique called polarity-switching. For the physics discussions that follow, the experimental conditions that are used will be based on typical laboratory and PK-4 operating parameters. A full detailed description on the PK-4 laboratory setup will be given in Chapter 2.

## **1.2. Thermal Properties of Dusty Plasmas**

If measurements of the positions and velocities of a collection of objects (i.e., the dust particles in the plasma) can be made, then the velocity space distribution, and its moments, can potentially be computed. If so, then it is possible to use these measurements to extract the thermodynamic properties of a system. This process will be shown in detail in the next section

### **1.2.1, Maxwell-Boltzmann Distributions.**

Dusty plasmas are particularly useful for this technique because we can measure positions and velocities with high precision. Briefly, we use high speed cameras to capture the movement of the dust particles in both gravity and microgravity conditions using the PK-4 experiment. The resulting images are analyzed to obtain detailed positions and velocity vectors of the dust particles.

Hardware descriptions for the cameras and analysis techniques will be detailed further in sections 2.1.3 and 2.2.2, respectively.

The extraction of the positions and velocities enables measurements of a variety of energy-related properties of the dusty plasma. These include: the electrostatic potential energy from interactions of charged particles (charging of particles is described in Section 1.3), and the kinetic energy from the motion of the particles. Generally, as the system interacts and evolves it is possible to have an exchange between the types of energy in the system. By tracking the properties of the particles and calculating these energies, the evolution of energy in time, and how energy is converted from one form to another, it is possible to gain new insights into how microscopic processes (i.e., dust-dust and dust-plasma interactions) become coupled and lead to macroscopic changes in either global plasma properties (e.g., density, electron/ion temperatures, Debye screening, etc.) or spatial or temporal changes in the arrangements of the dust particles. Some examples of this occurring in dusty plasmas are when plasma crystals melt via modifications of the operating conditions (e.g., neutral gas pressure or input power) that lead to changes to underlying plasma parameters, [26–28], changing an external force by the application of a magnetic field that again, leads to changes in the underlying plasma parameters [29,30], in instabilities of mode coupling where changes in interaction potential between dust grains can lead to new spatial arrangements of the dust particles [31,32], or by introducing shock waves [33]. This transition of energy is also seen in other types of plasmas, such as atmospheric pressure, ultracold neutral, or cluster plasmas [34–37]. Therefore, the coupling between electrostatic potential energy and kinetic energy along with structural changes in physical systems can lead to the generation of kinetic energy, and this newly generated kinetic energy is converted into heating of the system (dust cloud kinetic temperature). In the work described in this dissertation, it will be postulated that the dust particles in these experiments will utilize these microscopic structural changes arising

from a change the background plasma will ultimately provide a source a energy that drives an extended energy dissipation process that is observed in microgravity experiments.

### 1.2.1. Maxwell-Boltzmann Distributions

“Statistical mechanics is a probabilistic approach to equilibrium macroscopic properties of large numbers of degrees of freedom” [38], and this probabilistic approach will be the basis of our analysis throughout this work. Distribution functions are a fundamental way to statistically determine the probability of a value occurring, given a larger set of possibilities, used frequently in statistical mechanics. More specifically for a velocity distribution function: “What is the probability that a particle will have a velocity,  $v$ , given the distribution,  $f(v)$ ?” Velocity distributions are used to describe particle speeds in a variety of environments, i.e., ideal gases, fluids, and of course, plasmas [39–41].

The 3D Maxwell-Boltzmann velocity distribution function is defined:

$$f_v(v_x, v_y, v_z) = \left( \frac{m}{2\pi k_B T} \right)^{3/2} \exp\left( \frac{-m(v_x^2 + v_y^2 + v_z^2)}{2k_B T} \right) \quad (1-1)$$

where  $v_{x,y,z}$  is the velocity of a particle in a given direction,  $m$  is the mass of the particle,  $k_B$  is the Boltzmann constant, and  $T$  is the equilibrium temperature of the particles in the system. Each of the three directions is a degree of freedom, and the velocities are independent, normally distributed variables, so the distribution can be described as the product of a 1D distributions in each of the three cartesian directions:

$$f_v(v_x, v_y, v_z) = f_v(v_x) f_v(v_y) f_v(v_z) \quad (1-2)$$

By combining the independence of each direction in Equation 1-2, and the 3D Maxwell-Boltzmann distribution in Equation 1-1, we can extract 1D Maxwell-Boltzmann distribution of velocity for each direction of our dust cloud in a plasma environment:

$$f(v_D) = \sqrt{\frac{m}{2\pi k_B T_D}} \exp\left(\frac{-m(v_D - v_{drift,D})^2}{2k_B T_D}\right) \quad (1-3)$$

where  $m$  is the mass of the dust particle,  $k_B$  is the Boltzmann constant,  $T_D$  is the dust cloud temperature,  $v_D$  is the velocity of an individual dust particle, and  $v_{Drift,D}$  is the drift velocity of the system of particles.

From the independent 1D distributions, we can further extract the drift velocity and kinetic temperature of the dust cloud. By fitting experimental results with a Maxwell-Boltzmann (MB) fit, we can extract the characteristic distribution function for the dust cloud. From the mean of this fit we can extract drift velocity, and from the width we can extract and convert into effective dust cloud kinetic temperature. These are the two fundamental values that we will use throughout this work to characterize the dust cloud in the plasma environment. A more detailed explanation of getting these drift and temperature values from the MB fit will be shown in the PIV Section, 2.2.2.

### 1.3. Charging of Dust Grains

In a plasma environment, the electrons are typically more energetic and mobile than the ions. This means that the dust particles have interactions with the electrons more frequently and those electrons can initially arrive at the dust particles' surfaces faster and in larger net amounts than the ions. As the dust particle becomes negatively charged, the particles acquire a floating negative potential relative to the plasma to equalize the electron and ion fluxes. This eventually reaches a steady-state and we can determine the charge of a dust grain. After the charging is at this steady state, the dust particles are considered to be a third charged particle component of the plasma system.

Dust particle charge is often estimated using Orbital-Motion Limited (OML) theory [42–44] for spherical particles. OML is a modification of Langmuir probe theory [45,46], and the

derivations can be found in many dusty plasma textbooks [47,48]. Since this dissertation will look at the interaction between the plasma environment and dust particles, in part due to charging, the derivation will be expanded on below.

By looking at a plasma charged particle species (electron or ion) interacting with a dust particle, we can use a cross-sectional collision approach. A collision, even between charged particles, is governed by conservation of angular momentum and energy (and just includes a potential energy term into the energy equation). By assuming that the dust particle is at a floating potential within the plasma environment,  $q_d = r_d \phi_d$ , it can be shown that the cross-sectional relation for a plasma particle species,  $s$ , (ions and electrons) and dust particle,  $d$ , collision is:

$$\sigma_s = \pi r_d^2 \left( 1 - \frac{2q_s \phi_d}{m_s v_s^2} \right) \quad (1-4)$$

where  $r_d$  is the radius of the dust grain,  $q_{s(d)}$  is the charge of the plasma (dust) particle,  $m_{s(d)}$  is the mass of the plasma (dust) particle, and  $v_s$  is the initial velocity of the plasma particle at an infinite distance away.

By using a Maxwell Boltzmann distribution (see Section 1.2.1) and this cross section above, we can solve for the currents of the ions and electrons in the plasma using the integral:

$$I_s = q_s \int_{v_{min}}^{\infty} v_s \sigma_s f(v_s) dv_s \quad (1-5)$$

and evaluating Equation 1-5 for both plasma species (electrons and ions) to get:

$$I_e = \left( 4 \pi r_d^2 n_e e \sqrt{\frac{k_b T_e}{2 \pi m_e}} \right) \exp\left(-\frac{e |\phi_d|}{k_B T_e}\right) \quad (1-6)$$

$$I_i = \left( 4 \pi r_d^2 n_i q_i \sqrt{\frac{k_b T_i}{2 \pi m_i}} \right) \left( 1 + \frac{e |\phi_d|}{k_B T_i} \right)$$

where  $n_{e,i}$  is the electron (ion) density in the plasma,  $e$  is the fundamental charge of an electron,  $q_i$  is the charge of an ion,  $T_{e,i}$  is the temperature of the electrons (ions) in the system,  $m_{e,i}$  is the mass of the electrons (ions),  $\phi_d$  is the potential of the dust, and  $k_B$  is the Boltzmann constant.

Since we assume that the dust particle is at a floating potential in the plasma environment, this means that the net current (flux) of the plasma species' particles (ions and electrons) are at a steady state:

$$I_i + I_e = 0 \quad (1-7)$$

and by inserting the current equations of 1-6 and simplifying, we show:

$$\sqrt{\frac{m_e T_i}{m_i T_e}} \left( 1 + \frac{e|\phi_d|}{k_B T_i} \right) - \exp\left(-\frac{e|\phi_d|}{k_B T_e}\right) = 0 \quad (1-8)$$

which is an equation we can use to solve for  $\phi_d$  numerically. By assuming the dust particle is a spherical capacitor in the plasma system, we can use a value of  $\phi_d$  from Equation 1-8 to find the dust particles' charge:

$$q_d = 4 \pi \epsilon_0 r_d \phi_d = Z_d e \quad (1-9)$$

where  $\epsilon_0$  is the permittivity of free space, and  $Z_d$  is the number of elementary charges,  $e$ , that are gathered on the dust grain's surface.

Using this definition of the dust grain charge, we can expand the quasineutrality condition, a fundamental characteristic for plasmas. Quasineutrality is a fundamental characteristic for plasmas where the charge of all components is conserved in the bulk plasma system. With the addition of dust particles, this can be expressed as:

$$\sum_s n_s q_s = e (n_i - n_e - Z_d n_d) = 0 \quad (1-10)$$

where  $s$  is a plasma species,  $n$  is the number density of the species in the plasma. This characteristic of charging in a plasma will be used later in this chapter for the derivation of the fundamental screening parameter for a plasmas, the Debye length.

To help provide context for the charge of a dust particle in a complex plasma, consider the typical plasma we use in our Auburn experiments. A typical gas we use is Argon ( $\sim 40$  amu), where the electrons have a thermal temperature of 4 eV, and the ions are at room temperature, or 1/40 eV. If we use a melamine formaldehyde dust particle with a diameter of 3  $\mu\text{m}$ , we can estimate the dust charge of  $\sim 9300 e$ , or  $- 1.5 \times 10^{-19} \text{ C}$ .

#### **1.4. Select Forces in a Dusty Plasma**

There are many forces that the dust particles experience in a dusty plasma system, depending on the experimental environment. The forces we will be focusing on that are relevant to both the ground and microgravity experiment using the PK-4 experiment are interaction forces between dust particles, drag forces, electric forces, and gravity. Additionally, for modeling purposes we incorporate a randomized thermal heater, or Brownian motion. When combined, this represented the force-balance system of equations for the dust cloud in the PK-4 experiment.

##### **1.4.1. Interaction Forces**

Since dust particles are typically negatively charged in a plasma environment, as previously shown, they have electrostatic interactions with each other. The simple definition of electrostatic potential for a charged particle is a Coulombic potential:

$$\Phi_{Coulomb} = \frac{q_i}{4 \pi \epsilon_0 r} \quad (1-11)$$

where  $q$  is the charge of the particle, and  $r$  is the distance between the particle and the reference point. When applying this to two charged particles interacting, you get a force:



$$\vec{F}_{Coulomb} = \frac{q_i q_j}{4\pi\epsilon_0 r_{ij}^2} \hat{r} \quad (1-12)$$

where  $q$  is the charge of the individual particles, and  $r$  is the distance between the two particles. This is considered valid for infinite interaction lengths, meaning it is a long-range effect.

In a plasma environment with ions and electrons charged species, the Coulomb interaction is not an accurate representation of the entire system's interaction with a dust particle. Since the ion and electron components of the plasma populations are also charged, there is a finite region of interaction lengths where the dust particles would feel the force of the other dust particles. This region is defined by a Debye-Huckel or Yukawa Interaction Potential [49–51]:

$$\Phi_{Yukawa} = \frac{q}{4\pi\epsilon_0 r} e^{-r/\lambda_D} \quad (1-13)$$

where  $q$  is the charge of the plasma particle,  $r$  is the distance from the dust particle, and  $\lambda_D$  is the Debye length (described further in Section 1.4.2). When taking this shielded Yukawa potential into account, an interaction force can be defined as:

$$\vec{F}_{interaction} = \frac{q_i q_j}{4\pi\epsilon_0} e^{-r_{ij}/\lambda_D} \left( \frac{1}{r_{ij}^2} + \frac{1}{r_{ij} \lambda_D} \right) \hat{r} \quad (1-14)$$

where  $q_{i,j}$  are the charges of the two dust particles,  $r$  is the distance between the particles, and  $\lambda_D$  is the effective screening length of the dust particle in the plasma environment. This modified interaction force decays exponentially as the distance goes to infinity, and therefore the dust-dust interactions will be spatially limited by the properties of the background plasma.

### 1.4.2. Debye Length

The Debye length is numerically defined as the distance where the bare electrostatic potential of a charged particle has dropped in magnitude by a factor of  $e$ . Outside of a sphere with a radius of this distance, the electric potential of the other charged particles in a system is screened to the charge particle of reference [52]. For negatively charged dust particles, some ions move

closer to the dust particle and some electrons are repelled. By Gauss's law, particles outside the Debye length would see a neutral total charge, and after several Debye radiuses distance dust- dust interactions can be considered neutral, hence the fundamental plasma characteristic of quasineutrality described above. This allows the plasma to shield the potentials from longer-range charged plasma species from each other.

While the Debye length makes sense conceptually, it arises from important derivations for a plasma system using Poisson's equation for the four-component (three-charged species) plasma:

$$\nabla^2 \Phi = \frac{-e}{\epsilon_0} (-n_e + n_i - Z_d n_d) \quad (1-15)$$

where  $e$  is the fundamental charge of an electron,  $n_s$  is the number density of a plasma species (electrons,  $e$ ; ions,  $i$ ; dust particles,  $d$ ), and  $Z_d$  is the number of fundamental charges accumulated on the dust particle's surface, Equation 1-9.

Assuming a Boltzmann equilibrium for the ion and electrons, and performing a Taylor series of the electrostatic potential, we can define the ion and electron densities as:

$$n_s \approx n_{s_0} \left( 1 - \frac{q_s \Phi}{k_B T_s} \right) \quad (1-16)$$

Combining Equations 1-15 and 1-16 along with quasineutrality definition from Equation 1-10, it yields a differential equation for  $\Phi$  of the ions and electron species:

$$\nabla^2 \Phi - \left( \frac{e^2 n_{e_0}}{\epsilon_0 k_B T_e} + \frac{q_i^2 n_{i_0}}{\epsilon_0 k_B T_i} \right) \Phi = 0 \quad (1-17)$$

To solve this differential equation, we can define the Debye length for a single plasma species (ion or electron):

$$\lambda_s = \sqrt{\frac{\epsilon_0 k_B T_s}{n_s q_s^2}} \quad (1-18)$$

which allows the differential equation to be rewritten as:

$$\nabla^2 \Phi - \left( \frac{1}{\lambda_i^2} + \frac{1}{\lambda_e^2} \right) \Phi = 0 \quad (1-19)$$

By adding our third charged particle component, dust, into the equation, and due to quasineutrality, we can say the dust particles' Debye length depends on both the ion and electron Debye lengths to be able to solve our differential equation for the dust particles:

$$\frac{1}{\lambda_D^2} = \frac{1}{\lambda_i^2} + \frac{1}{\lambda_e^2} \quad (1-20)$$

By taking limits of the dust Debye length equation ( $T_e \gg T_i$  therefore  $\lambda_e \gg \lambda_i$  in Equation 1-18), we can assume that the ion Debye length of the plasma system is an upper limit for the dust particle Debye length. As will be shown throughout this work, we are unable to know exact measurements of the plasma parameters (particularly for the ions) necessary to calculate a value for the dust Debye length. While this is a precise definition of screening, experimental uncertainties place limits on a precise calculation of the Debye length. Therefore, throughout this work, the main objective will be to determine an “effective screening length” for the complex plasmas that is constrained by the ion and electron Debye lengths.

### 1.4.3. Epstein Drag

The next fundamental force in a dusty plasma environment is Epstein drag [53]. This is a specific kind of drag for spherical particles in a gas environment and is commonly used in plasma environments as well [54]. Epstein drag is defined:

$$\vec{F}_{epstein} = -\frac{8}{3} \sqrt{2\pi r_d^2 m_n N_n v_{Tn} v_{rel}} \hat{v} \quad (1-21)$$

where  $r_d$  is the radius of the dust particle,  $m_n$  is the mass of the neutral gas particles,  $N_n$  is the density of the neutral gas,  $v_{Tn}$  is the thermal velocity of the neutrals, and  $v_{rel}$  is the dust particles' relative velocity compared to the neutrals. We also refer to Epstein drag as neutral drag, since it arises from interactions with the gas environment, not the charged particle species in the plasma

system. Those ionized particles can also create drag on the dust particles [54–56], but we will be accounting for these interactions in other ways.

By setting the basic equation of a drag force,  $\vec{F} = -\gamma m_D \vec{v}$ , equal to  $F_{\text{epstein}}$ , we can solve for the drag coefficient:

$$\gamma = \frac{\frac{8}{3} \sqrt{2\pi} r_D^2 m_N \frac{P}{k_B T} v_{T_N}}{m_D} \quad (1-22)$$

where  $r_d$  is the radius of the dust particle,  $m_n$  is the mass of the neutral gas particles with substituting  $P/k_B T$  for  $N_n$  as the density of the neutral gas (from the ideal gas law),  $v_{Tn}$  is the thermal velocity of the neutrals, and  $m_d$  is the dust particle's mass. This coefficient will be used extensively for our simulations, described in section 2.3, and utilized in Chapter 4.

A further note for thermal velocity is that we use the three-dimensional definition of thermal velocity:

$$\vec{v}_{T_N} = \sqrt{\frac{8k_B T}{\pi m}} \hat{v} \quad (1-23)$$

which is based on using the mean magnitude of velocity of the neutral particles in a three-dimensional system, as compared to other definitions which are from a root mean square approach or probability-based approach. Depending on the thermal velocity definition used, there are some differences in the constants seen in Equation 1-22, but the substitution of variables is consistent for all definitions and therefore the dependence of the drag constant on the other plasma characteristic variables, such as mass and temperature, is the same.

#### 1.4.4. Electric Forces

To create a dc discharge plasma, as in PK-4, a high voltage between two electrodes is used to create a large electric potential to ionize the gas within the vacuum chamber. Because of

quasineutrality and plasma Debye shielding, in dc plasmas most of the potential drop occurs near the electrodes (i.e., anode and cathode), giving rise to substantial electric fields. However, because there is a need to maintain a net current to maintain the plasma, there exists a residual electric field in dc discharge plasmas. This electric field can be manipulated through oscillations in dc potential applied to the electrodes. Since the dust particles are (negatively) charged, they respond to this electric field with a force,  $\vec{F} = q\vec{E}_d$ . This force can be modified to become oscillating with the addition of a sinusoidal component:

$$\vec{F} = q\vec{E}_d \sin(f_{ps} t) \quad (1-24)$$

where  $f_{ps}$  is the frequency of the oscillation, and  $t$  is time. Depending on the mass of the plasma charged particle species and the frequency of this oscillation, some particles (electrons and ions) gain a net acceleration, but at high frequencies ( $f_{ps} > 200$  Hz), the dust particles remain stationary because of their large inertia ( $m_d > 10^{12} m_{ion}$ ). This stationary response due to an oscillating electric field is the basis of polarity switching for our experiment (hence the subscript  $ps$  on the frequency in Equation 1-24), which will be discussed in detail in Section 2.1.4.

#### 1.4.5. Langevin Heater

There are random fluctuations in motion for a dust particle in a fluid-like system at thermal equilibrium, due to many interactions at a given instance. When two gas particles collide (or come near enough to repel), this can change their direction of motion. Since all particles' motion cannot be accounted for individually, this must be calculated statistically and can change the total internal energy of a system. Due to the large size of the dust particles compared to the other plasma species, the plasma can be considered a fluid-like background system, and we have previously shown dusty plasma systems are considered to be at thermal equilibrium. Therefore, we can use a Langevin heater [57,58] to describe the background plasma "bath" of the dust particles when investigating

the particles' motion in the complex plasma system. While there is not a specific force definition for this interaction (only conservation of momentum), this is implemented in simulations by:

$$\vec{F}_{Thermal} = \vec{\alpha}_{rng} * F_{magnitude} \quad (1-25)$$

where  $\alpha_{rng}$  is a random number multiplier of value (0-1) for all vector directions, and  $F_{magnitude}$  is the maximum force that would be applied to a dust particle, typically calculated to create a cloud at room temperature.

#### 1.4.6. System of Equations

The final force for this work is the most fundamental, gravity,  $F = mg$ . Gravity is the attractive interaction between objects with mass, in this case, the dust particles and the Earth. Gravity provides subtle differences in dusty plasma experiments that will be further described in detail in Chapter 3.

Finally, all of these forces can combine into a force balance equation for a dust particle:

$$m_i \frac{d\vec{v}_i}{dt} = \vec{F}_{Interaction} + \vec{F}_E + \vec{F}_{Epstein} + \vec{F}_{Thermal} + (\vec{F}_{gravity}) \quad (1-26)$$

where this can be set into a system of N equations and solved for the entire dust cloud population. This will be the system of equations solved for all dust particles by our MD simulation, discussed in further detail in Section 2.3.

### 1.5. Outline of Dissertation

This dissertation is divided into five segments: Chapter 1 described the relevant physics topics involved in the project, including distribution functions, charging of dust particles, and dusty plasma forces, chapter 2 will discuss the experiment setup and the relevant analysis techniques used for this work, chapter 3 will show the analysis process of one experimental dataset and the

results for all other datasets, chapter 4 will discuss our numeric model and supporting simulation results, and chapter 5 will summarize the discussion of the project and future work opportunities.

## **Chapter 2: Experimental Setup and Analysis Techniques**

In the previous chapter, the most important relevant physics concepts for this work were introduced. This chapter will introduce the experiment apparatus, Plasmakristall-4, in section 2.1, the analysis techniques used for experimental data in section 2.2, and a simulation code description we will use to support our results in section 2.3. These three topics will be used throughout the scope of this dissertation and are described here to serve as a further introduction to the project before presenting the analysis and results.

Several parts of Chapter 2 - 4 are adapted from a manuscript that is in preparation for this work and under internal review by the PK-4 science team. In particular, sections for this adaptation include Sections 2.1.2 and 2.1.4 when describing the experimental settings; Sections 3.2 and 3.2.2 for an overview of experimental results; and Section 4.4 as an overview to our simulation comparison to experimental results.

### **2.1. PK-4 Apparatus**

All experimental work for this dissertation was completed on an experiment apparatus called Plasmakristall-4 (PK-4), which is an experiment that utilizes the microgravity conditions of the International Space Station (ISS). PK-4 is a multi-user apparatus located in the European Space Agency (ESA) Columbus module of the ISS, which allows for a community of scientific users to propose and conduct experiments for each campaign of the microgravity experiments. This work



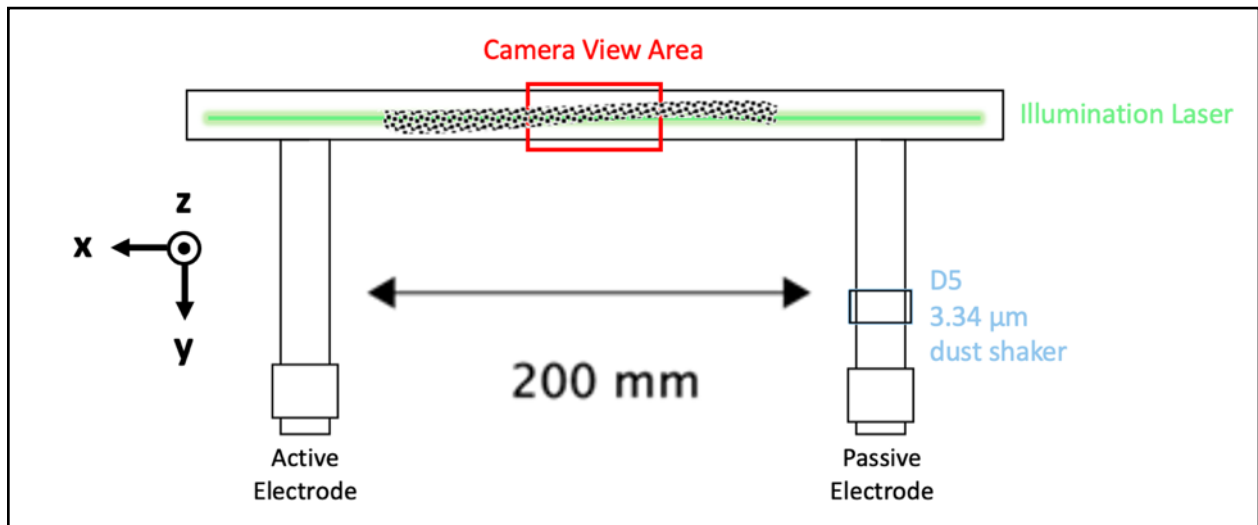
would not be possible without the scientific development team and the team at CADMOS, who work to make each campaign (15 and counting) a success for all scientists involved.

The development of microgravity dusty plasma experiments has spanned several decades. The most important benefit of microgravity complex plasma research is that the microparticles are unconstrained from gravity's influence and can freely expand in the plasma to form a 3D system that can fill the entire plasma volume. This allows for investigations into fluid and solid like structures, that can aid in studies related to soft matter materials [22,59,60].

For the ISS-based experiments, the Plasmakristall Series began in 2001 with PKE-Nefedov [61]. PK-3 Plus was commissioned on the ISS in 2006 [61]. These experiments focused on stationary structures in a complex plasma system. The next experiment in the series wanted to investigate dynamic dusty plasmas at a kinetic level, which eventually led to Plasmakristall-4's experiment design, launching in 2014, and beginning user experiments in 2017. The Plasma Kristall series is lead by German scientists at The German Aerospace Center (DLR, Deutsches Zentrum für Luft- und Raumfahrt), with collaborations from the Russian Academy of Sciences (JIHT) and the European Space Agency (ESA), of varying degrees for each iteration of the PK series. The PK-4 instrument on the ISS has identical modules located in Toulouse, France (CADMOS, ESA mission control), Germany (DLR), and Russia (JIHT) that are used for ground-based testing and validation. Ground-based results that are presented in this dissertation are performed using the PK-4 Science Reference Module that is located in Oberpfaffenhofen, Germany. An extensive report on the development and specifications of the PK-4 experiment can be found in *Pustlynik, et al.* [62].

### 2.1.1. PK-4 Chamber Description

PK-4 is a U-shaped, glass vacuum chamber, with about 200 mm of visible working area. There are a variety of electrodes that can produce dc or rf plasmas, or even a combination plasma, as we use in campaign 14, described further in Appendix Section A.3.3. PK-4 has six different dust shakers to introduce spherical, melamine formaldehyde (MF) particles ranging in size from 1.31 to 10.41  $\mu\text{m}$  in diameter, with the most utilized sizes being 3.34  $\mu\text{m}$  and 6.86  $\mu\text{m}$  diameter particles. There are a variety of diagnostics systems including cameras, lasers, and a spectrometer, which will be described in further detail below.



**Figure 2-1: Schematic of PK-4 experiment on board the International Space Station. The dc current to the active and passive electrodes creates the oscillating electric field for polarity switching. The coordinate system is shown on the left, and therefore the cameras show the x-direction horizontally and z-direction vertically. In both ground-based and microgravity experiments, the x-direction corresponds to the direction of the axial electric field. In ground-based experiments, the vector direction z corresponds to the direction of gravity.**

A schematic of the apparatus, adapted from *Pustyl'nik, et al.* [62], is shown in Figure 2-1. The PK-4 apparatus has working capabilities for neon and argon gases, with the additional

capability to generate oxygen plasmas for cleaning the vacuum chamber. There is a total pressure range of 0.1 – 2 mBar (75 – 1500 mTorr) for the system with voltages up to 2.7 kV, creating a wide variety of plasma operating conditions. For the experiments presented in this work, neon dc glow discharge plasmas in PK-4 are generated using the active electrode shown in Figure 2-1. In addition, modulation of the dc current on this electrode is used for polarity switching, to be discussed in greater detail in Section 2.1.4 below.

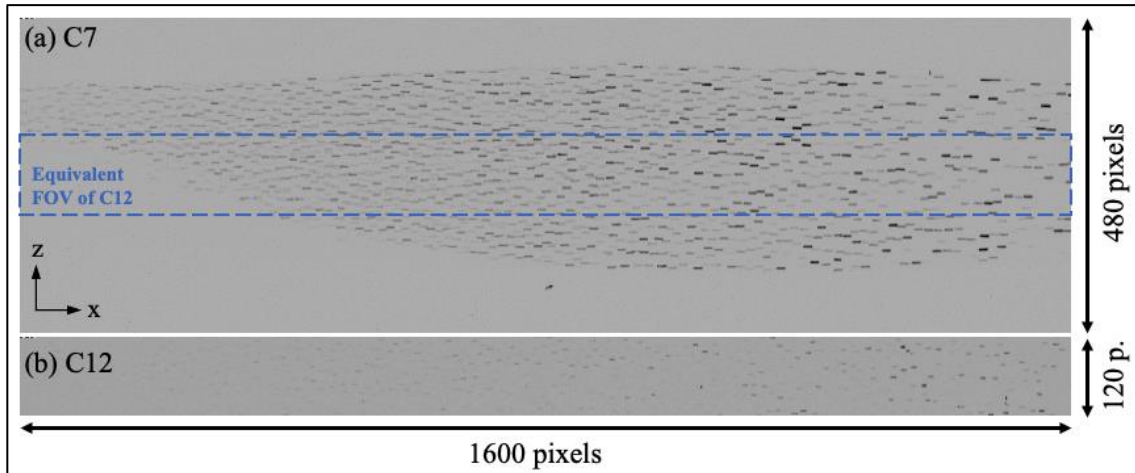
PK-4 experiments are performed using scripts that are written using a C-scripting language of libraries developed by the DLR group. Because both the PK-4 Science Reference Module instrument and the PK-4 flight instrument on the ISS use the same scripts in both locations, all operating conditions are equal, except gravity and camera height (for sheath/ dust-levitation purposes on the ground). These scripts ensure that the timings of the application of polarity switching as well as experimental conditions (e.g. plasma conditions and electric field) are identical in both locations, allowing for the direct comparison of gravity and microgravity experiments. The flowchart for the scripts that operate each of our experiments can be found in Appendix A, with Campaign 7's information in Section A.1.

### **2.1.2. Plasma Conditions**

The experiments discussed here used neon dc glow discharge plasma generated with neutral gas pressures,  $p = 0.2 - 0.6$  mBar (150 – 450 mTorr) and discharge currents,  $I_{DC} = 0.35 - 1.0$  mA. Following the parameter characterization from empirical models described in *Pustylnik, et al.* [62], the plasma conditions in PK-4 for our experiment have the following ranges: electron density,  $n_e = (0.9 - 2.8) \times 10^8 \text{ cm}^{-3}$ , electron temperature,  $T_e = 8.3 - 8.5$  eV, and an axial electric field,  $E_x = 210 - 250$  V/m.

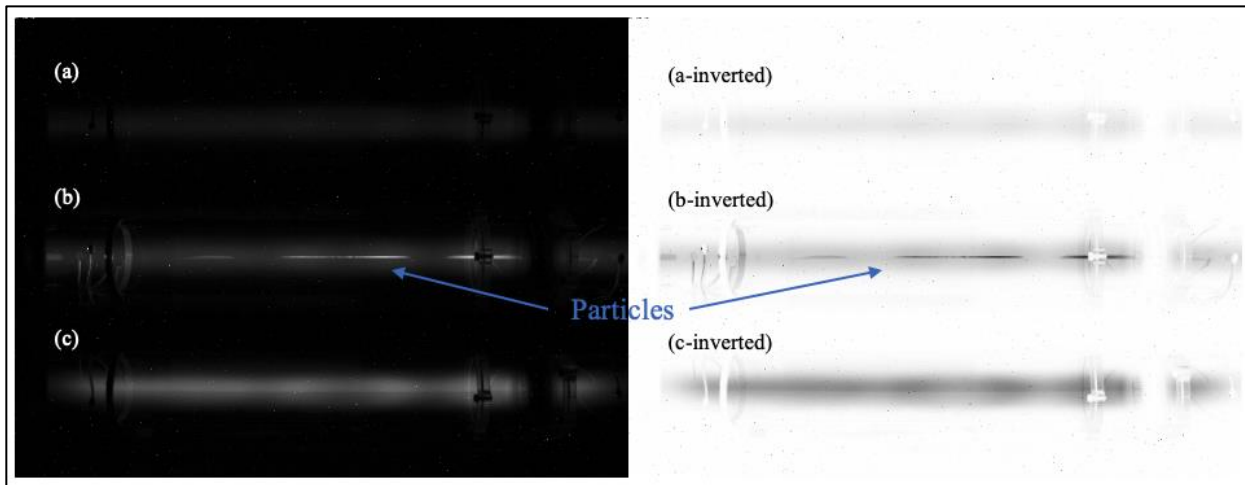
For the studies described here, a single particle dispenser (D5) is used to introduce  $3.34 \mu\text{m}$  diameter MF particles into the experiment (density,  $\rho_d = 1510 \text{ kg/m}^3$  and mass,  $m_d = 2.95 \times 10^{-14} \text{ kg}$ ). When injected into PK-4, the dust particles generally form an ellipsoid cloud that is flowing through the field of view of the experiment with peak axial velocities of up to  $20 \text{ mm/s}$ . During both the “flowing” and “capture” phases of the experiment, the interparticle spacing typically varies from  $200$  to  $300 \mu\text{m}$ , roughly corresponding to dust number densities  $n_d \sim 10^5 \text{ cm}^{-3}$ . Examples of flowing and capture data can be seen in Figure 2-4a and Figure 2-4b, respectively. Using the plasma parameters given above and assuming that the particle charge can be computed using an orbit-motion-limited model [63], the dust grain charge can be approximated to be in the range  $Z_d = 15,000\text{-}17,000$  elementary charges. All of these can be combined to estimate the dust plasma frequency,  $\omega_{dp} = 103 \text{ rad/sec}$ .

### 2.1.3. Camera Settings



**Figure 2-2:** a) The field of view of a microgravity injection at 70 fps, with a vertical width of 480 pixels. Injection from Campaign 7, Injection 3. B) the field of view of a microgravity injection at 140 fps with a vertical width of 120 pixels. Injection from Campaign 12, Injection 2. Both frames are from PO1, at  $p = 0.6 \text{ mBar}$ ,  $I = 0.7 \text{ mA}$  operating conditions, and the blue overlap box shows where the focus of the C12 FOV is with respect to the dust cloud. The image colors are inverted for visibility.

PK-4 uses two Basler cameras, denoted as Particle Observation camera 1 and 2 (PO1 and PO2, respectively), with an overlapping horizontal field-of-view region of  $\sim 2$  mm. This produces data from a region of  $\Delta x \sim 40$  mm. The typical camera framerate used for PK-4 is 70 fps, but the vertical field of view (FOV) can be reduced to run the cameras at a higher framerate (up to 140 fps). A sample frame from campaign 7 and campaign 12 are shown below in Figure 2-2a and b, respectively, to show this field-of-view to framerate tradeoff. Both frames shown are from the same microgravity plasma operating conditions,  $p = 0.6$  mBar, and  $I = 0.7$  mA. We utilized this tradeoff for campaign 12 to get a higher temporal resolution (or fps) to further investigate our campaign 7 results, and the Campaign 12 data will be discussed in Section 3.4.1. The reduced field-of-view is centered in the middle of the larger field-of-view, so we are looking at the middle of the cloud in both instances. Each camera has a slightly different scaling conversion (all  $\sim 14 \mu\text{m} / \text{pixel}$ ) in the vertical and horizontal directions, and the specific conversion values are listed in Table IV of *Pustynnik, et al.* [62].



**Figure 2-3: A sample VM3 frame from Campaign 7. The left is the normal view, and the right is color-inverted, to make the plasma glow show easier in print. a) the 703.2 nm filter view, b) the unfiltered plasma glow view, and c) the 585.2 nm filter view.**

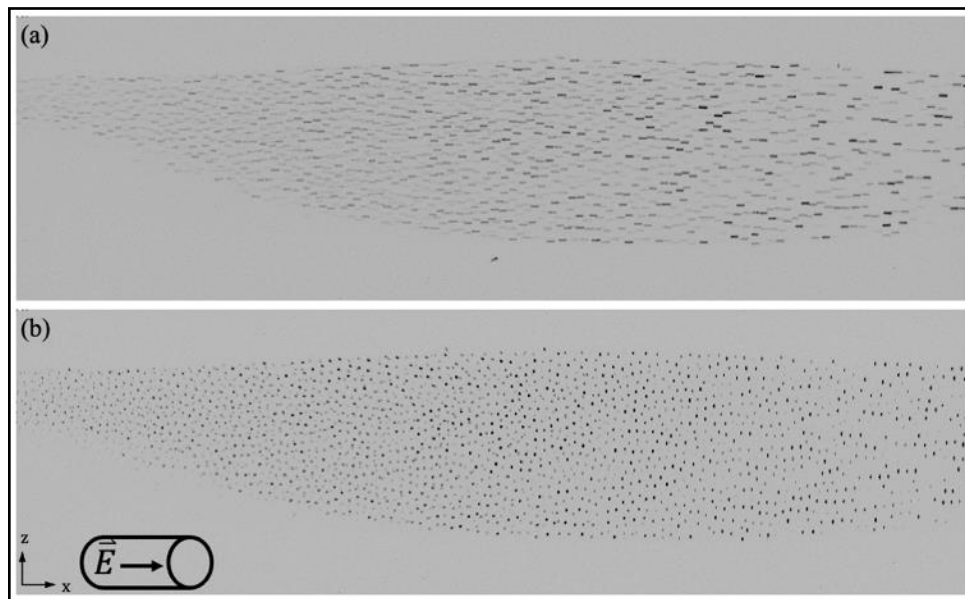
There is also an overview camera, VM3, that allows you to see the full x-direction and width of the chamber as shown in Figure 2-3 (left). A duplicated but color-inverted frame is shown on the right side to better see subtleties of the plasma glow. This view focuses on the plasma glow, but you are still able to see the location of the dust cloud when dust is present. A careful examination of the middle view (b) shows that there are three brighter (or darker in inverted form) thin lines in the middle of the chamber. This is the location of the dust cloud, which is split into at least three clouds for this experiment. For this run, the PO camera data focused on the biggest cloud slightly to the right of the middle of the chamber.

This camera has a kaleidoscopic mirror system that splits the view into three channels, as seen in Figure 2-3. The top has a filter of the 703.2 nm line, and the bottom has a filter of the 585.2 nm line, which allows for more specific information from the plasma than the visible spectrum view in the middle. From this split view, we can gain additional information about the plasma background environment for the dust cloud by looking at plasma glow intensities and taking intensity ratios of the two filtered views. The downside of this camera is that it operates at only 35 fps, so there is not sufficient temporal resolution to closely investigate the changes in the plasma system. We initially considered a detailed analysis using the data from VM3, but difficulties encountered in the synchronization among the various video cameras as well as the slower frame rate introduced a number of analysis challenges, so the initial results will be presented in Section 5.2.4, Future Work.

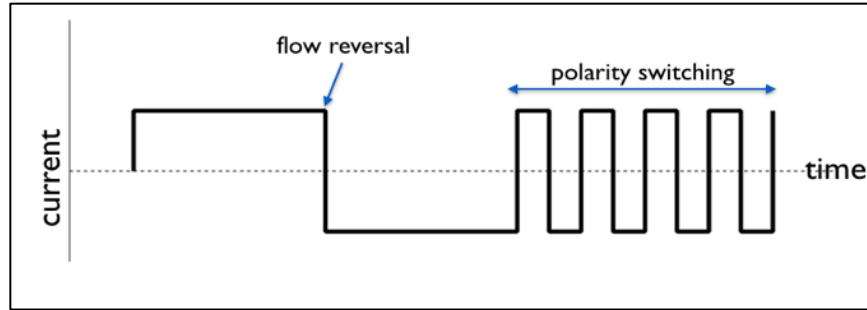
#### **2.1.4. Polarity Switching**

The primary technique to capture a dust cloud within the PK-4 apparatus when operating in dc mode is polarity switching, a rapid oscillation of the axial (x-direction) electric field through a periodic modulation of the dc current on the plasma generating electrode, as illustrated in Figure

2-5. When the direction of the current is reversed, either once or periodically, it causes a change in the direction of the axial electric field in the plasma. This, in turn, leads to a reversal of the flow direction of the dust particles. If the oscillation frequency of the current is fast enough ( $f_{ps} > 100$  Hz) then the particles are captured. While we refer to the resulting oscillations in frequency, this is actually set by time intervals ( $t_{0,1,2,3}$ ) to the pulse generator and duty cycle. The time intervals are described in detail in section 2 of *Pustylnik, et al* [62], and is shown in the settings of our experiment flow chart in Appendix A.



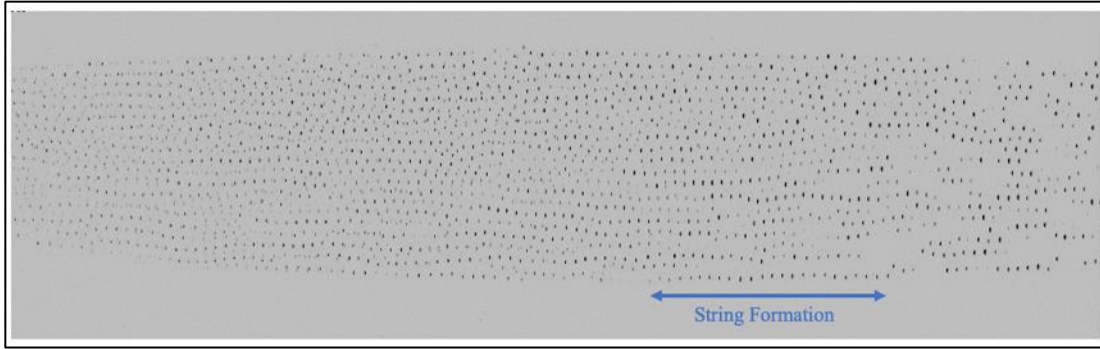
**Figure 2-4: A sample microgravity PO1 frame during a) the injection process, or “flow” where the dust particles appear as streaks across many pixels and b) once the dust cloud is captured by polarity switching, or “capture”, where the dust particles appear more spherical. Both frames are color inverted for easier viewing. Note, a) is the same frame as shown in the FOV trade off in Figure 2-2a, for 70 fps.**



**Figure 2-5: Illustration of dc current for applying polarity switching to the dust cloud. When alternating slowly, the dust cloud can respond and have a flow reversal (Figure 2-4a). When alternating rapidly, the dust cloud is “captured” for the camera (Figure 2-4b).**

For all of the experiments described in this initial work, the polarity switching frequency is  $f_{ps} = 500$  Hz or  $\omega_{ps} = 3140$  rad/sec. When compared to the dust plasma frequency ( $\omega_{dp} = 103$  rad/sec), the applied polarity switching frequency is over a factor of 30 higher. This means that the charged particle species of the plasma (e.g. the ions and electrons) can respond to this oscillation, but the dust particles cannot, due to their large inertia. As a result, at the application of polarity switching, the flowing motion of the dust cloud is halted and the dust is “captured” for camera recording purposes. Once the dust cloud is captured, the particles begin to form string-like structures after a few seconds [64–67], which can be seen in Figure 2-6. These “strings” consist of  $N > 5$  particles all aligned (mostly) horizontally throughout the dust cloud, with most particles throughout the volume being in a string, and this formation hold throughout the entire polarity switching segment. The goal of this work is to investigate this initial redistribution of the kinetic energy of the dust cloud immediately after ( $t < 1$  s) the application of polarity switching.





**Figure 2-6: String formation in a microgravity PK-4 environment. This dataset is  $p = 0.6$  mBar,  $I = 0.7$  mA from Campaign 7. The image color is inverted for visibility.**

To optimize the dust particles and plasma conditions, we also utilized a PK-4 technique called reinjections. This is where the electric field is set to a negative constant value to send the dust cloud out of the field-of-view in the  $-x$ -direction, then set to a positive constant value similar to an initial injection to bring the cloud back into the field of view, and then finally captured using the same oscillation settings for polarity switching as a “standard” injection capture. This allows us to “reset” the cloud and repeat experiments without having to flush the system of particles in order to inject and form a new cloud. We use this reinjection technique when we change the plasma discharge current, while keeping the same gas pressure. Our analysis indicates that the phenomena later described in this work associated with the redistribution of the flow kinetic energy is consistent for both the initial “injection” and “reinjection” processes, which will be shown in the analysis of Section 3.4.2.

## **2.2. PIV Analysis Techniques**

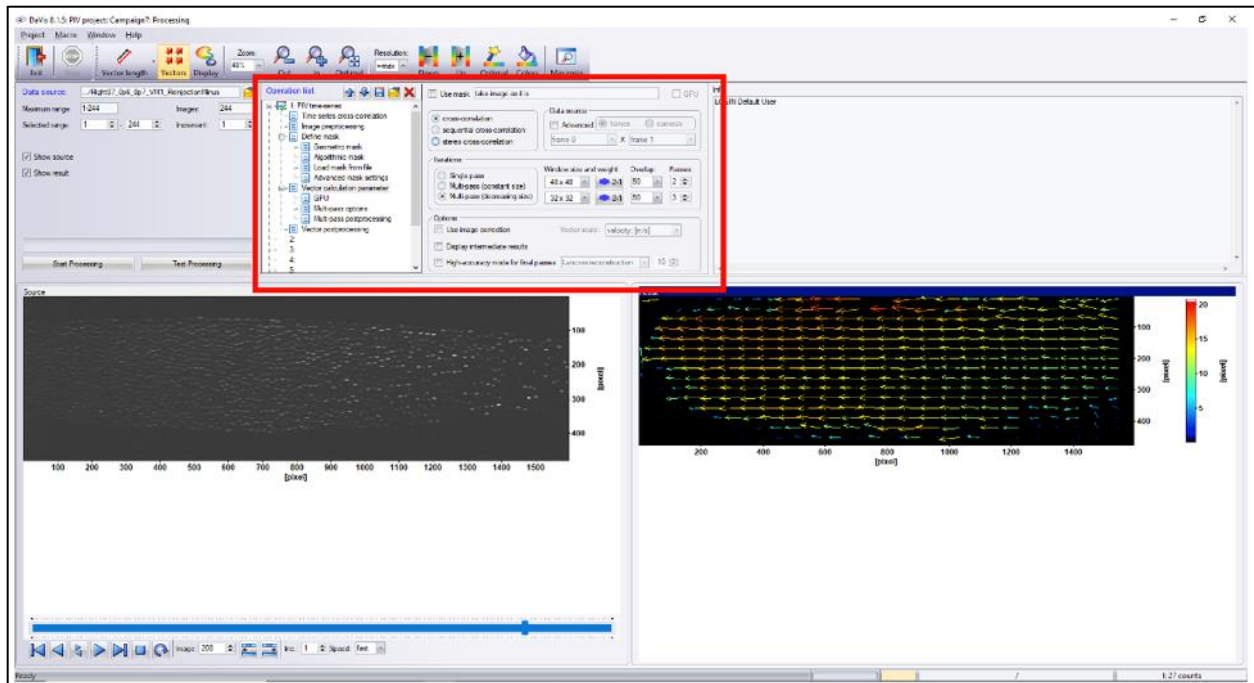
To analyze the particle motion from the PO cameras, Particle Image Velocimetry (PIV) techniques are used. We use the software *DaVis* by LaVision for our PIV analysis [68]. PIV is a particle analysis technique in which an image is decomposed into interrogation cells typically containing three or more particles and a cross-correlation between two consecutive images in a

sequence is performed to determine the average velocity vector that corresponds to the group of particles.

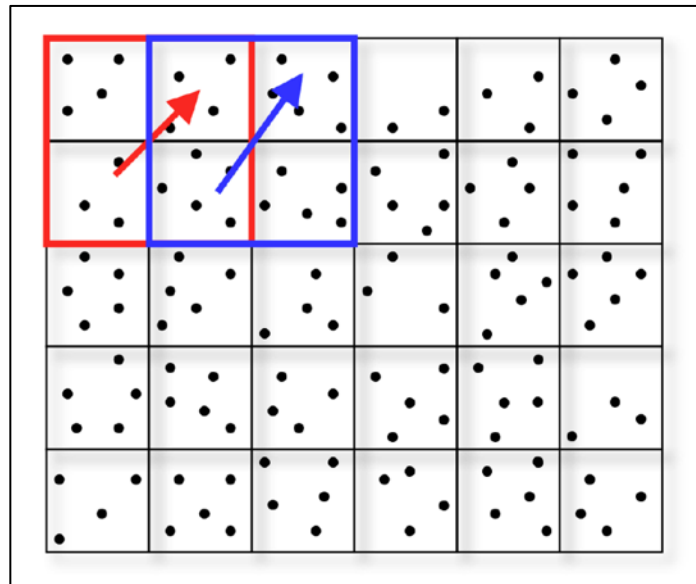
Dedicated PIV hardware systems can be used to capture images, but image sequences from “high speed” cameras can also be used with PIV analysis techniques. The PIV technique was first applied to fluids, automotive, and aerospace systems [69–72]. However, PIV techniques have been used extensively for dusty plasma studies and earlier work by *Thomas, et al.*, *Williams, et al.* [73], and *Fisher, et al.* [74], specifically for determining the thermodynamic properties of dusty plasmas. Additionally, PIV techniques have been specifically benchmarked against ground-based PK-4 experiments [75] and PK-4 simulation [76]. For the PK-4 measurements discussed here, the high-speed imaging technique is used to obtain 2D-2V (two spatial dimensions, two velocity components) vectors and then used to extract representative thermodynamics quantities.

### **2.2.1. Processing and Settings**

The PIV software has many settings to optimize the vector results, a representative screenshot of the software interface is seen in Figure 2-7, and further images of the settings described in the walkthrough of this technique can be found in Appendix B.2. This shows a sample image and the resultant test vector field at the bottom of the screen, with the file settings (i.e. total images in sequence) on the far left. The various settings for the PIV sequence processing that are discussed below are all found within the area highlighted by the red box, and each file is shown in better detail in the appendix.



**Figure 2-7:** DaVis software vector processing window. The settings for PIV discussed are in the upper middle, and a sample frame and resultant vector field are shown at the bottom of the screen.



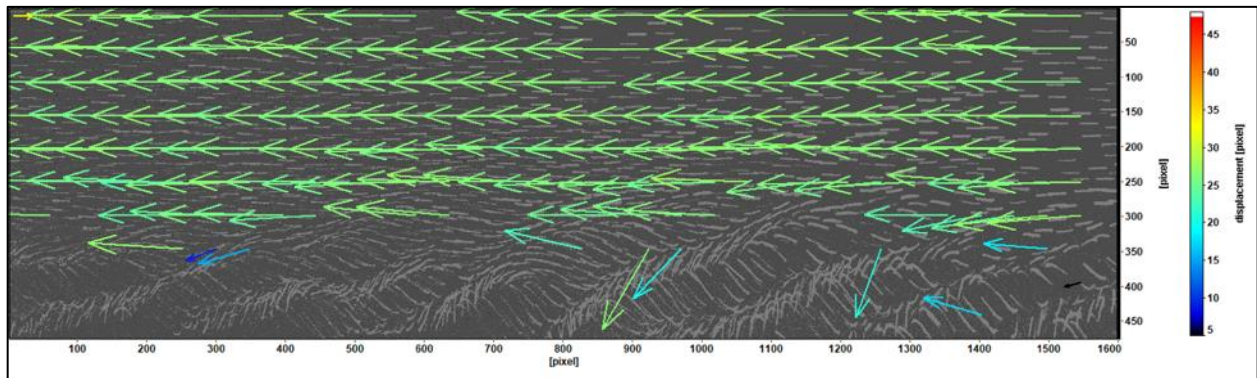
**Figure 2-8:** A cartoon of the PIV analysis technique. The dots represent the dust cloud, and the black grid illustrates the bin regions. The red and blue boxes show two interrogation regions of 2x2 bin size, with a 50% overlap. *Reproduced from Dr. Jeremiah William's dissertation; Auburn University, 2006.*

A cartoon of the PIV technique is seen in Figure 2-8. The first group of settings is focused on the data and algorithm process. The first setting is bin size, usually set in number of pixels. We select a bin size so that there are at least a few dust particles in every bin. For PK-4 PIV, this is typically at least 32 x 32 pixel bin sizes. Along with the bin size, we can select a gaussian weighting function shape for the bins, which we utilize to account for the shape of streaking particles during high flow speeds (see Figure 2-4a). The weighting function shape options are circles, ellipsoids at a 2:1 width to height ratio, or ellipsoids at a 4:1 ratio, and the ellipsoids can be at varying angles of orientation as needed. For flowing injections in PK-4 we typically use a 2:1 or 4:1 horizontal oriented ellipsoid. And for captured cloud segments, we typically use circles on one pass, and 2:1 horizontal ellipsoids on a second pass. We can also select the % overlap of the bins (usually 50%), and how many passes over the two consecutive frames that the algorithm uses (usually 2 or 3 passes). There are also masking functions where you can ignore part of the frame, which can be important if there is a stationary object in view, such as probes.

After the processing is complete for a set of consecutive frames, we can then apply postprocessing to refine the resultant vector field. The settings here focus on finding “bad” vectors to improve our results. The most common setting we utilize is for if the magnitude of vectors exceeds a maximum value that we determine (usually 25 mm/s for flowing PK-4 data). Other settings are if the signal to noise ratio for a cross-correlation is too large, or a vector’s magnitude compared to its neighbors is inconsistent (with a few different parameters to determine this). There are also interpolation and smoothing settings, but we do not use those as to not impose artificial data in the vector field results.

All of these settings play an effect into the resultant vector field, but ultimately, we want to have a large enough number of vectors returned ( $N > 1000$  / frame) that have a statistically significant number of vectors to determine the dust cloud characteristics. A large enough sample

size minimizes the artificial effects that can arise in a PIV dataset [77] and therefore more accurately represents the true velocity results of a dust cloud in a plasma environment.



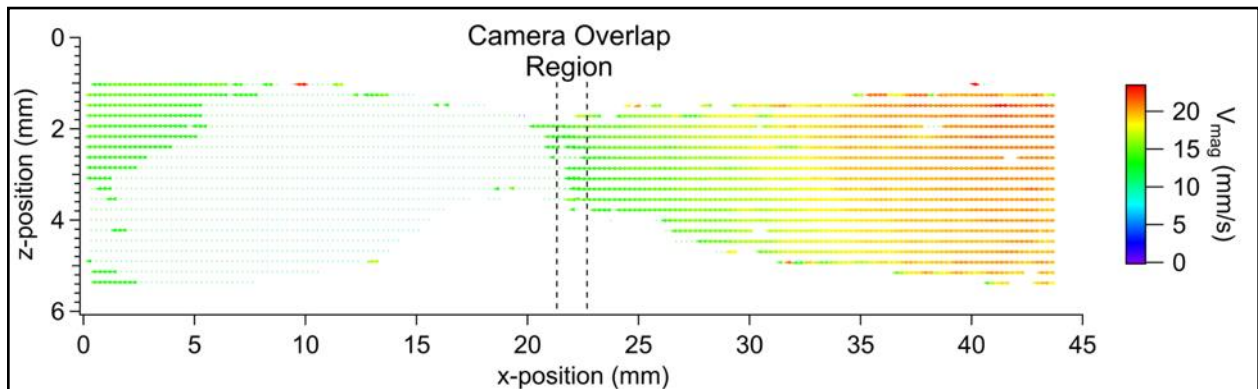
**Figure 2-9: A sample Ground-based PIV resultant vector field, overlapping the original data frame, to illustrate that the waves in the bottom of the field-of-view do not yield (reasonable) vectors when using the “flow” settings. Note, the background image contrast is saturated to make the waves more visible in the lower part of the image.**

A sample result of PIV is shown in Figure 2-9. This is a ground-based dataset where waves are typically present in the lower portion of the field-of-view. This vector field illustrates the recreation of the flowing portion of the dust cloud, and typically returns a “poor” reconstruction of the waves (with the exception of a few vectors with these settings). This resultant vector field is created using our “flow” PIV settings. PIV is also capable of reproducing the wave dynamics in a dust cloud [74,78], but requires different settings [79], and they are not able to be recreated at the same time, hence the blank area over the waves in Figure 2-9.

Because PIV measures the motion of groups of particles, it is particularly well-suited for higher particle number densities and higher speed particle flows. Therefore, PIV is ideally suited for the analysis of dust particles in PK-4, as illustrated in the image of flowing particles shown in Figure 2-4a. While Particle Tracking Velocimetry (PTV) techniques do work well for the “capture” portion of the experiment (Figure 2-4b) when drift velocities are low, we seek to measure

the velocity space distribution before and after polarity switching, so the PIV technique is used both portions of the data in order to obtain self-consistent results. PIV is a statistical approach to determining a vector field for a system, whereas PTV is an individual, aggregate approach. When the dust cloud drift speeds are large enough, the particles “streak” in a frame, and PTV is unable to determine their motion. PIV can also adjust for varying resultant image particle shapes (i.e. how many pixels are illuminated from a particle) as part of the motion calculations, such as the ellipsoids described in Section 2.2.1.

We process each PO camera individually, but in the overlapping region we can obtain the duplicate vectors and produce a resultant vector field that encompasses the entire field of view in the PK-4 experiment, as seen in Figure 2-10. Finding overlap regions in PIV is easier than the true view of the frames because there is less noise. There are also a few subtle angle differences between cameras as well, that can be resolved easier with vector fields instead of aligning individual particles [80].

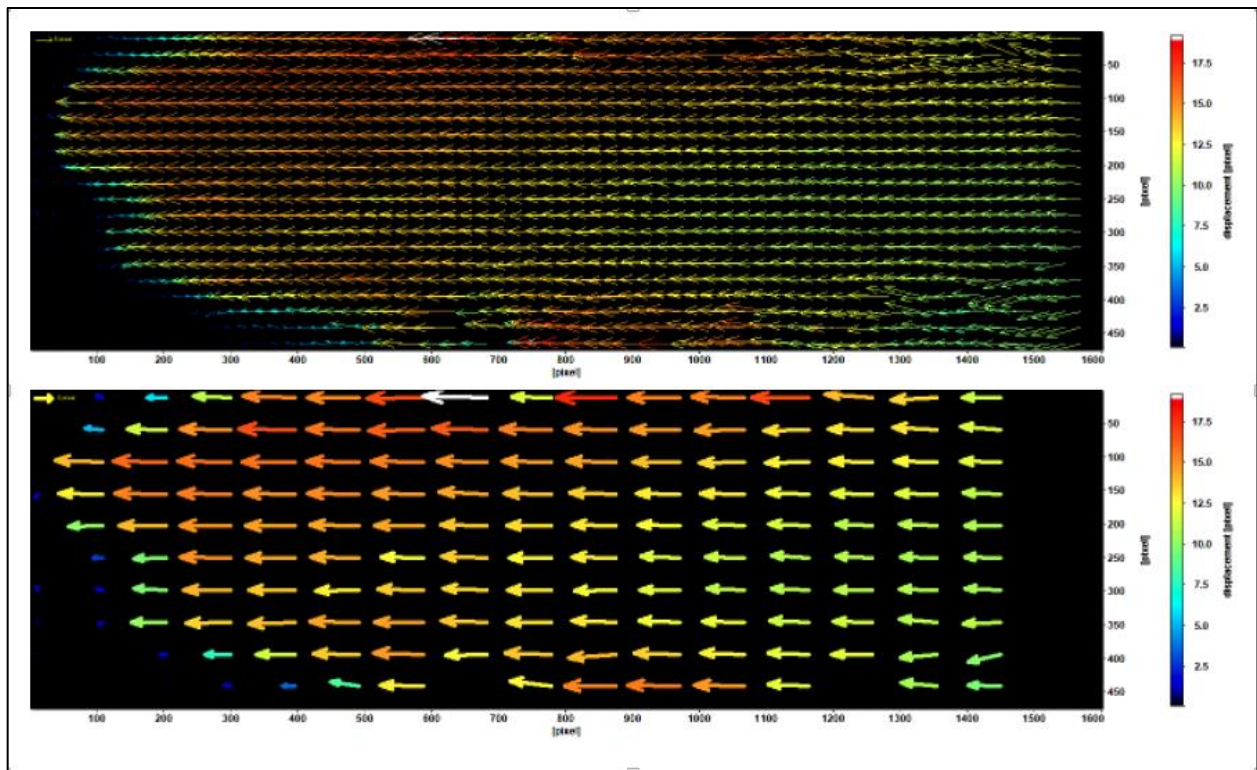


**Figure 2-10: A sample total vector field reconstruction. This is a microgravity dataset at  $p = 0.6$  mBar,  $I = 0.7$  mA, the dataset we will focus our analysis on in Section 3.2. The cloud was split into two smaller clouds during this run, (which can be seen by the dim middle section in the cloud of Figure 2-9), hence the low reconstruction of vectors in  $x = 10 - 20$  mm.**



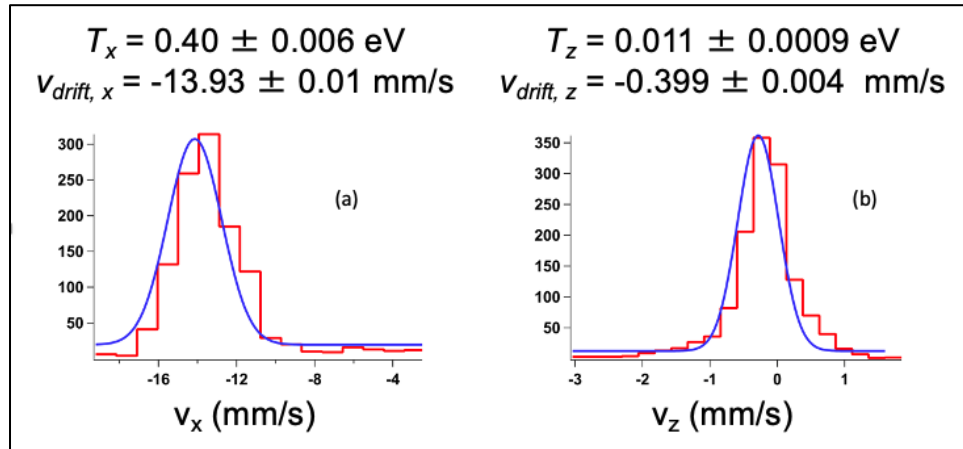
### 2.2.2. Analysis Techniques

PIV techniques are used to characterize the time evolution of the two-dimensional velocity distribution. PIV returns a vector field (see Figure 2-9 and Figure 2-10) for the frame and the components of the vectors are binned into histograms and fit using a Maxwell-Boltzmann distribution (detailed in Section 1.2.1). This technique is applied before and after polarity switching and can be plotted dynamically in time to see the evolution of the system. This is done for both the ground and microgravity experiments in which identical scripts are used to perform both experiments. Most of the analysis of the PIV vectors throughout this work is processed in Igor, and the analysis code can be found in Appendix **Error! Reference source not found.**



**Figure 2-11: A sample resultant PIV frame of (a) total vectors returned and (b) representative vector field with 1/8 vector density for easier display. This is a frame from the injection of the  $p = 0.6$  mBar,  $I = 0.7$  mA microgravity dataset.**

A sample single frame from PIV is shown in Figure 2-11, where the vector color is displacement in pixels. The upper frame shows the total vectors returned, and the lower frame is at a lower display resolution (1/8 vector density) for easier viewing of the same field. After this data is exported, we can convert the displacement in pixels to velocity in mm/s using the camera resolution ( $\sim 14 \mu\text{m} / \text{pixel}$ ) and framerate (70 fps, or  $\Delta t = 0.014 \text{ s}$ ). Each vector can be split into its horizontal (x-direction) and vertical (z-direction) components, and we can produce histograms of all components in a single frame, as seen in Figure 2-12. The histograms shown (red) can be fit with a Maxwell-Boltzmann distribution (blue), using Equation 1-3, and from the fit, we can extract the drift velocity and kinetic temperature of the dust cloud for this frame, with the values listed above each histogram. This can be done for a series of images to investigate the evolution of drift velocity and temperature for the dust cloud throughout the experimental run.

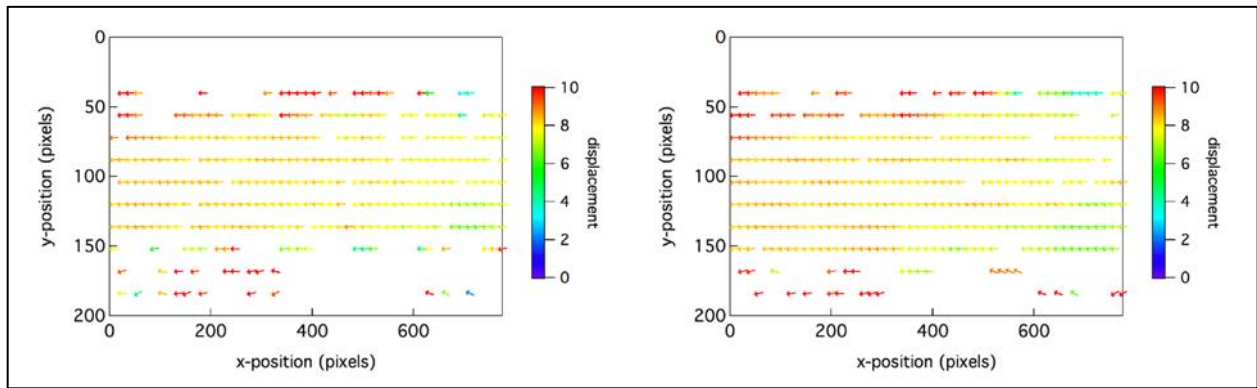


**Figure 2-12: Histograms (red) of the (a) x-direction and (b) z-direction components of velocity for the sample PIV frame in Figure 2-11. When we fit the histograms with a Maxwell-Boltzmann distribution (blue), we can extract the drift velocity and temperature of the dust cloud using Equation 1-3, and the values are listed above each histogram.**



### 2.2.3. Validation for PK-4: DLR report

Our group has extensively tested the reconstruction of PIV vectors on the PK-4 experiment. In 2015, at the earliest stages of this project, Professors Thomas and Williams submitted a report to DLR [75] demonstrating the various PIV settings' (shapes, passes, etc.) yield on vectors returned and resultant drift velocities for several framerates. This section will show the highlights from this report to confirm the validity of using PIV on the PK-4 experiments throughout this dissertation.

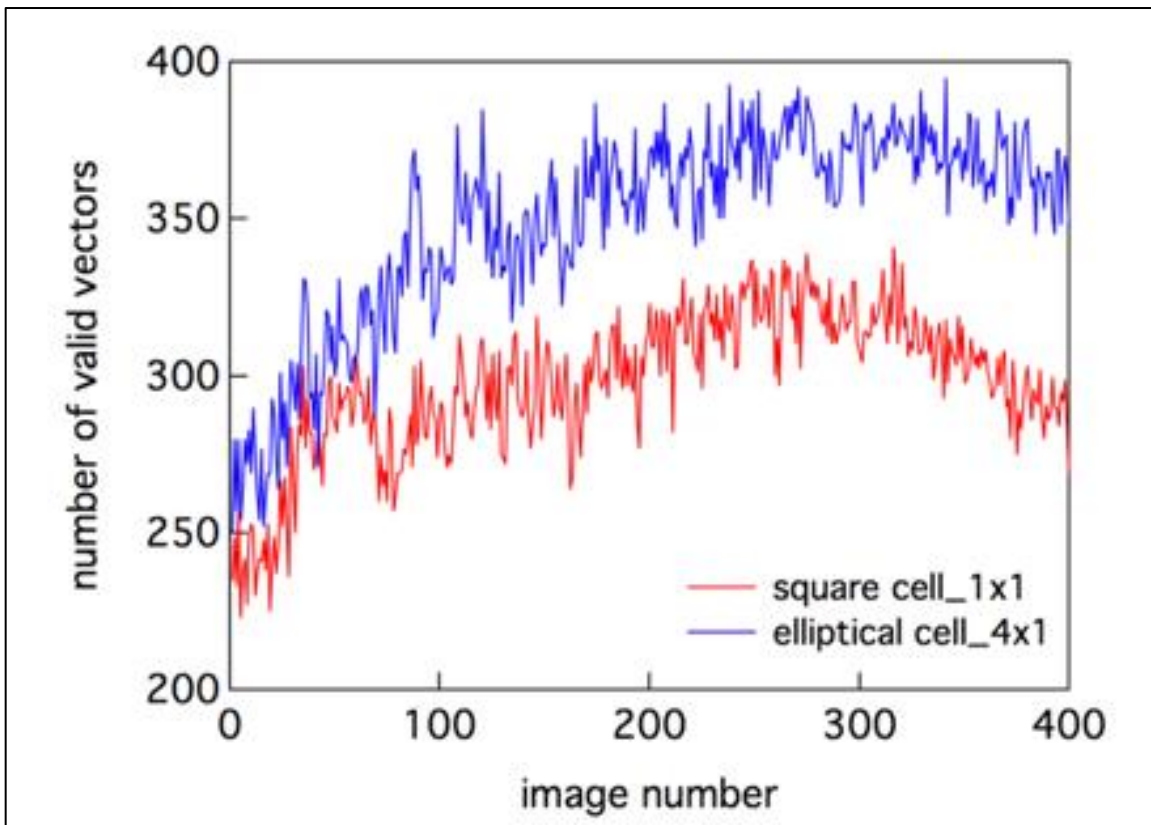


**Figure 2-13: Resultant vector fields of a) square bins and b) 4:1 horizontal elliptical bins for ground-based data recorded at 70 fps. There are additional vectors returned for the elliptical interrogation region at the top and bottom of the cloud, but overall the reconstruction of the vector field looks similar.**

The first test was to optimize the processing settings, described in 2.2.1. These settings help find particles in the video frames and determine statistically how the particles move between two consecutive frames using cross-correlation analysis. Figure 2-13 shows the difference in vector field for a) square bins, and b) 4:1 horizontal elliptical interrogation region for a ground-based dataset recorded at 70 fps. While the middle of the cloud might look similar, there are more vectors returned with the elliptical interrogation at the top and bottom of the cloud.

Subsequently, Figure 2-14 compares the total number of vectors returned for all frames in these datasets. The elliptical interrogation (blue) consistently returns more total vectors than the

square interrogation (red). Since our analysis on this processed data requires a large number of vectors in order to obtain good velocity distributions, we want to create as many vectors as possible with PIV to yield accurate statistical results from the distribution functions, and therefore we use the elliptical interrogation regions for our PIV processing.



**Figure 2-14: Total number of vectors returned by frame number for the square cells and the 4:1 elliptical regions as a function of time (image number). The elliptical interrogations consistently return more valid vectors.**

The next step of validating PIV use for PK-4 is to determine which framerates can be properly reconstructed. The average displacement for each framerate dataset varies slightly in Figure 2-15, but when the displacement is converted into velocity, as shown in Figure 2-16, there is good agreement between all framerates. A framerate of 140 fps is not typically used on the ground-based experiments because the FOV is so thin compared to the dust levitating in the sheath,

but we can confirm that the typical average horizontal velocities are comparable to the 3 framerates here, including 70 fps which is our primary framerate setting.

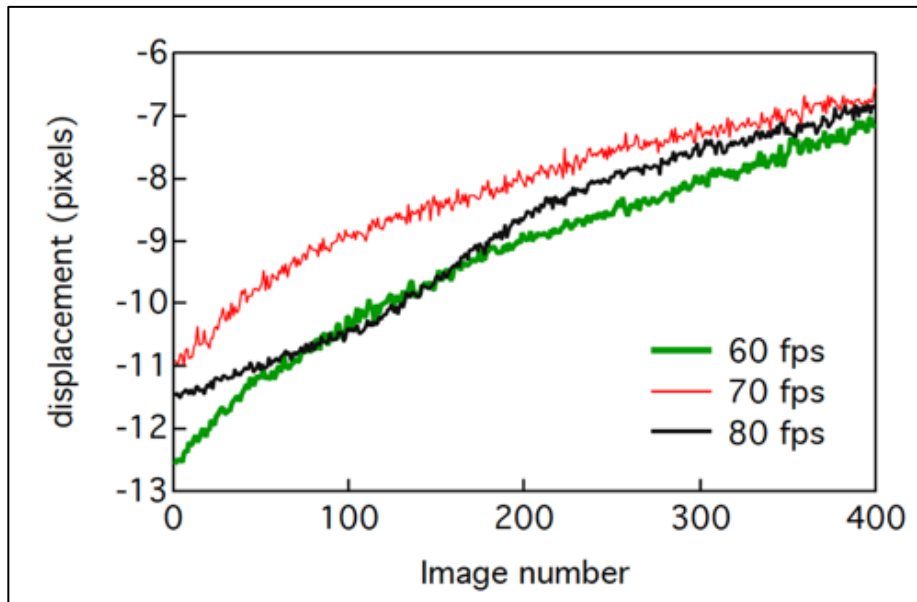


Figure 2-15: Comparison of the average displacement as a function of time (image number) for 60, 70, and 80 fps.

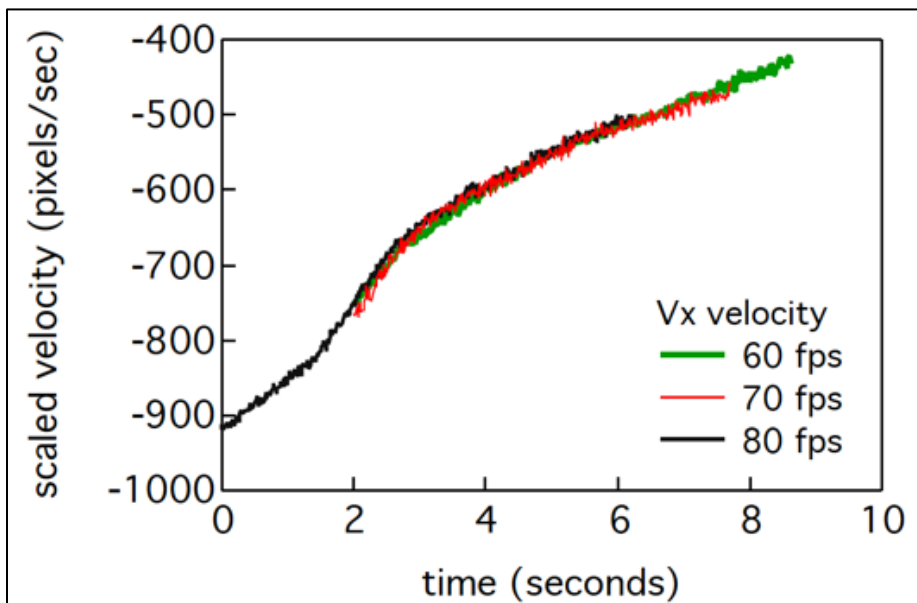


Figure 2-16: Comparison of the average horizontal velocity (scaled, pixels/s) as a function of time. The data shows good agreement between the three framerates at the same plasma operating conditions.

Based on this thorough validation of the PIV technique on PK-4 by Drs. Thomas and Williams, we are confident that the PIV analysis technique can be used on the flowing environment of the PK-4 experiment. And therefore, PIV is our primary analysis technique for the PK-4 experiments throughout this dissertation.

### **2.3. YOAK $\mu$ M- MD Simulation Code**

YOAK $\mu$ M (Yukawa-Ordered, Kristallized And microparticle( $\mu$ ) Model) is a molecular dynamics (MD) simulation code written in C++2017 coding language. The script of the code can be found in Appendix B.1. Molecular dynamics codes are a simulation type that utilizes balance of forces to calculate physical movements and interactions of particles. Each particle's state is accounted for individually by the code with position, velocity, and when needed, charge values. To evaluate the time evolution of a system, the equations of motion are evaluated for each particle at each time step- using a library of possible forces (as described in Section 1.4.6), including interactions among the particles. MD simulations are widely used in a variety of physical systems ranging from plasmas, to biophysics, to a wide variety of material systems, particularly soft-matter systems [81–83].

Since we know the particles' positions and velocities within the dust cloud, we can analyze the results in the same way as we do the vectors from the PIV results, which allows us to directly compare results between experiments and simulations. We are also able to use the simulated data to calculate other values, such as energies, to gain key insights into the system that we are unable to easily obtain from experimental measurements.

The primary limitation of an MD code simulating dusty plasmas is determining the correct timestep. With a large number of interacting particles,  $N$ , the number of calculations scales as  $N^2$ . To resolve the dynamics of the dust cloud requires a specific timestep between calculations as to

not accidentally mask the physics (i.e., Coulomb collisions of charged particles may not occur and simply pass through each other if the step is too large). Dust dynamics can typically be calculated on a millisecond timescale in many MD simulations, as determined by  $\Delta t \sim \frac{1}{\omega_{pd}} = 0.6$  ms. However, this work will frequently focus on microsecond timescales. The microsecond timescales allow for more information to pass between timesteps, just as the plasma interacts with the electric field and the dust particles more frequently in an experiment.

An important repercussion of the timestep limitations occurs when transitioning between two calculation timescales. Larger timestep sizes for the initial steps of a simulation are often used to establish equilibrium conditions in the simulation, but smaller timesteps may be needed to ensure to make sure all physics is properly incorporated. If you decrease the timestep between segments, that inherently gives a larger acceleration and therefore velocity kick to all particles, which in turn creates artificial heating in the system. This timestep difference becomes important in Section 4.1.2 for the YOAK $\mu$ M simulation results.

The framework of this code was done by an REU student in the summer of 2018, Dustin Samford from Baylor University. (He arbitrarily chose the code's name based on a county in Texas and we created the backronym to match our group's research interests). This code was modeled on our group's previous 2010 C++ MD simulation code, DEMON [84], but updated with the latest computer language (at the time) to be more time and memory computationally efficient. The basic forces of a plasma environment on a dust particle were benchmarked as part of Dustin's REU work, and we have continued to add forces to the YOAK $\mu$ M database as our work requires, such as adding electron beams as a manipulation force [85].

For the context of this work, YOAK $\mu$ M uses a fourth order Runge-Kutta [86] algorithm to solve either two- or three-dimensional particle dynamics for up to several thousand charged dust

particles in a background plasma. While the particle dynamics are evaluated self-consistently, the charge on the particles is calculated from the plasma parameters and is held at a fixed value. The particle interactions are governed via interparticle forces (Coulomb and Yukawa-screened), constant and oscillating electric fields, neutral drag, and a thermal (Langevin) heater, all as described in Section 1.4 above. The specifics of replicating a PK-4 experiment using this MD simulation will be the focus of Chapter 4.

## **Chapter 3: Experimental Results**

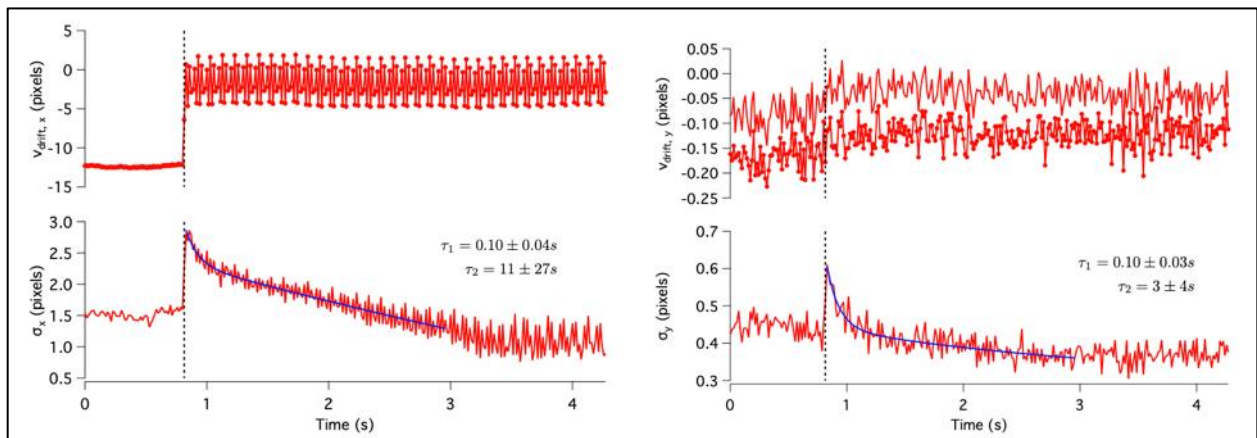
The goal of the experiments on PK-4 presented throughout this dissertation is to investigate the thermal energy evolution when the dust particles transition from a flowing dust cloud to a captured dust cloud at the application of polarity switching. Our first microgravity experiment was meant to be a test to simply see what happens in microgravity so that we could develop further in-depth experiments for our collaborative group. However, we identified much more rich phenomena than we were expecting in this experiment. Effectively, the dust dynamics occur during one second of data turned into my entire dissertation project. This chapter discusses the experimental proposal and development of our microgravity experiments in Section 3.1, a fully detailed analysis approach to our results for one dataset in Section 3.2, the results for all nine campaign 7 datasets in Section 3.3, supporting results from campaign 12 and other segments of campaign 7 in Section 3.4, and a final experimental results discussion in Section 3.5.

### **3.1. Experiment Development**

This work began in ~2012 to verify that the PIV technique can be applied to the PK-4 experiment. After some initial analysis (further detailed in the PIV Analysis Section 2.2.3), this approach was confirmed to yield accurate results for the PK-4 environment, and additional ground experiments were performed in 2016 and 2017 at DLR to begin our specific experiment design to investigate the dissipation of kinetic energy at the application of polarity switching [87].

Specifically, we wanted to look at dissipation and possible conversion of the kinetic energy when you go from a “flowing” to a “captured” dust cloud (see Figure 2-4).

Initial ground-based results from 2016, presented in Figure 3-1, showed that there was occasional evidence of dust particle heating. Data from these ground-based experiments showed that at the onset of polarity switching, there was experimental evidence of dust heating, i.e., an increase in the dust kinetic temperature, that occurs in both the direction along the flow (x-direction) and perpendicular to the flow (z-direction). In these studies, gravity points in the -z-direction.

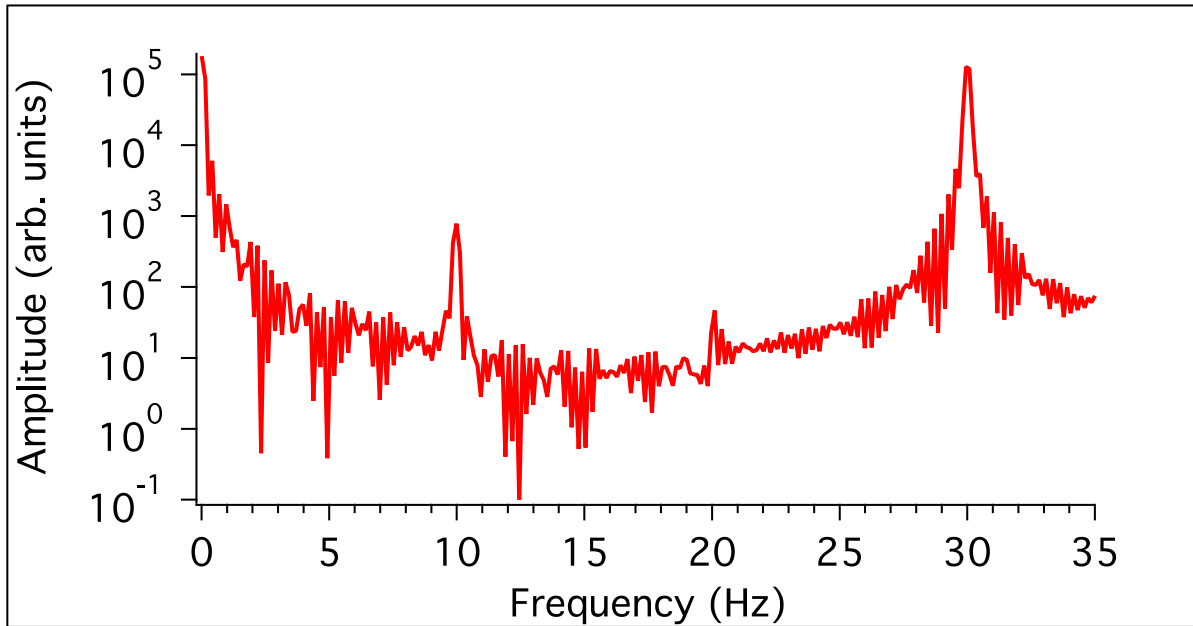


**Figure 3-1: Initial ground-based results taken in 2017. There is heating and extended decay in both directions. Note, the units for the vertical axes are pixels. The x-direction has larger heating and a longer decay than the y- (z) direction. This data was taken at  $p = 0.4$  mBar,  $I = 1.0$  mA. The polarity switching frequency was 100 Hz, which is why we can observe the drift velocity oscillations.**

To construct Figure 3-1, the particle velocities are measured using the PIV technique. For the uncalibrated data presented here, the particle velocities are reported as pixel displacements and the effective kinetic temperatures are reported in terms of the width of the velocity distribution function, again, in terms of pixels. Here, we consider the relative change in the width of the distribution. The dataset for Figure 3-1 was captured at  $p = 0.4$  mBar,  $I = 1.0$  mA, using a polarity

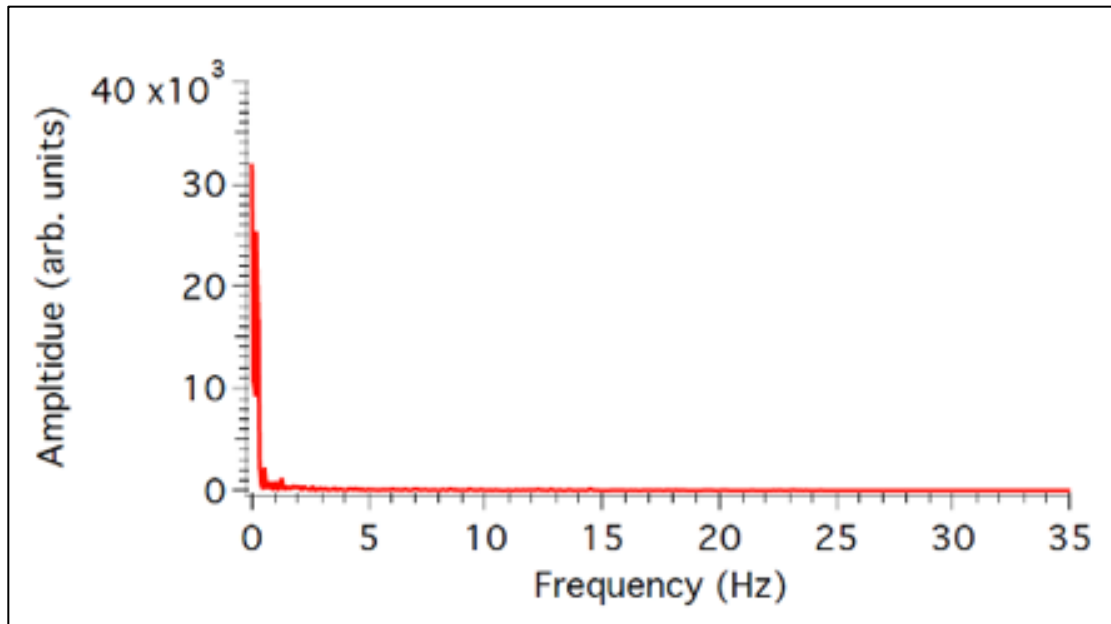


switching frequency of 100 Hz, which is why the x-direction drift velocity exhibits oscillatory motion. The heating and extended dissipation after the application of polarity switching (occurs at the dashed line) is still quite pronounced.



**Figure 3-2: FFT of the x-direction drift velocity as a function of time, for the same dataset as Figure 3-1, at polarity switching 100 Hz. There are peaks at 30 and 10 Hz. This indicates beating between the polarity switching and the 70 fps camera rate.**

When polarity switching is at 100 Hz, we see evidence of sampling bias, or beats, in the dust particles with respect to the camera framerate (70 fps). If we look at a Fourier Transform (Fast Fourier Transform method, FFT) of the drift velocity, as seen in Figure 3-2, the peak at 30 Hz corresponds to the difference between polarity switching and framerate ( $100 - 70 = 30$  Hz). As the polarity switching frequency increases, this is no longer an issue, as seen in Figure 3-3 which shows a dataset captured at 500 Hz and 70 fps. This led us to choose 500 Hz as our primary polarity switching frequency for our first microgravity experiment.



**Figure 3-3: FFT of the x-direction drift velocity as a function of time for a polarity switching frequency of 500 Hz. There are no longer any significant peaks.**

Based on these initial ground results, we wanted to run a similar experiment in the microgravity environment to see how the dust cloud dissipation of energy changes without the influence of gravity. In order to best utilize the experiment slot (~30 minutes), we collaborated with the CASPER dusty plasma group at Baylor University, who was investigating the string formations that occur at the application of polarity switching (shown in Figure 2-6). We were able to optimize the time of an experiment slot in Campaign 7 to develop an experiment where we all benefited from the data collection- Auburn from the initial captures of the cloud, and Baylor utilizing the long wait times to see the string formations develop. The proposal to the PK-4 science team and script description and flowchart of this accepted experiment are in Appendix A.1.1 and A.1.2, respectively. This experiment was proposed in February of 2019, ground testing was performed in May 2019, and the experiment was performed in July 2019.

For our campaign 7 experiment, we used a neon dc plasma. In both the ground-based and microgravity experiments, the MF particles (3.34 $\mu\text{m}$  diameter) are injected into the PK-4 plasma volume, flow along the axial electric field, and then are trapped using polarity switching, at a frequency of 500 Hz. We allow the particles to remain trapped for  $\sim 10$  s to form chains, and then we perform a y-scan (in and out of the FOV) to gain more knowledge about the 3D structure of the dust cloud. In the higher-pressure datasets, we included a polarity switching “stepdown” section where we lowered the frequency every 10 s,  $f_{\text{ps}} = (500, 250, 150, 100, 50, 25)$  Hz, so Baylor could investigate the changes in the string formation, and so we can investigate the sampling biases between polarity switching frequency and camera framerate (as shown above).

Parameter	Values or Ranges
Pressure, p (mBar) (mTorr) (Pa)	0.2, 0.4, 0.6 150, 300, 450 20, 40, 60
Current, $I_{\text{DC}}$ (mA)	0.5, 0.7, 1.0
Dust Diameter, $d_d$ ( $\mu\text{m}$ )	3.34
Plasma Density, $N_e$ ( $10^8 \text{ cm}^{-3}$ )	0.9 – 2.8
Electron Temperature, $T_e$ (eV)	8.3 – 8.5
Electric Field, $E$ (V/m)	211.0 - 311.3
Electron Debye Length, $\lambda_{\text{De}}$ (mm)	1.24 – 3.07
Epstein Drag Coefficient, $\gamma$ ( $\text{s}^{-1}$ )	57.1 – 171.4

**Table 3-1: Values or range of values for various plasma characteristics in our Campaign 7 experiment.**

Our first experiment was a success, and we obtained data over 9 different plasma parameter settings, or all combinations of the pressure and current listed in Table 3-1. Plasma Density, electron temperature, electric field, and the electron Debye length values listed are all calculated using the linear model values from *Pustylnik, et. al*, [62] for the plasma operating conditions. Debye length is defined in Equation 1-18, and the Epstein drag coefficient is defined in Equation 1-22. For a fully detailed analysis approach investigation throughout this dissertation, this work focuses on the  $p = 0.6$  mBar,  $I = 0.7$  mA dataset, and mainly the data from camera PO1, for the analysis walk through discussed in Section 3.2. Results comparing all 9 datasets analyzed with these techniques will be presented in full detail in Section 3.3. Data presented in this section uses spatially calibrated measurements so that velocities are reported in mm/s and the dust kinetic temperature will reported in electron-volts (eV).

The results shown throughout the rest of Chapter 3 will compare velocity and temperature measurements extracted using the PIV technique (Section 2.2) from the ground-based and microgravity experiments. In the presence of gravity, in the experiments performed using the ground reference module, the particles must be levitated in the sheath region of the PK-4 chamber and are therefore 0.5 to 1.5 cm below the centerline of the cylindrical glass tube. As a result, the particle flows are less stable and vertically propagating waves (parallel to gravity) are often observed as shown in Figure 2-9. These features are not observed in the microgravity experiments. Despite these observational differences, the use of the identical scripts ensures that a direct comparison can be made using studies on both experimental platforms for the portion of the cloud not exhibiting these waves.

### 3.2. Campaign 7 Results: Complete Analysis of a Single Dataset

An example of the time evolution of the particle flow velocity and the dust kinetic temperature are shown in Figure 3-4. In this case, the operating pressure is  $p = 0.6$  mBar, the dc discharge current is  $I_{DC} = 0.7$  mA, and the polarity switching frequency is 500 Hz. For all of the data shown for both the ground experiments (and later for the microgravity experiments), the horizontal axis is adjusted so that  $t = 0$  s corresponds to the video frame when polarity switching is applied to the experiment, determined by when the drift velocity magnitude decreases to  $\sim 0$  mm/s. As noted previously, the camera frame rate for all of these experiments is set at 70 frames per second, which corresponds to a  $\Delta t = 0.014$  second interval between each measured data point.

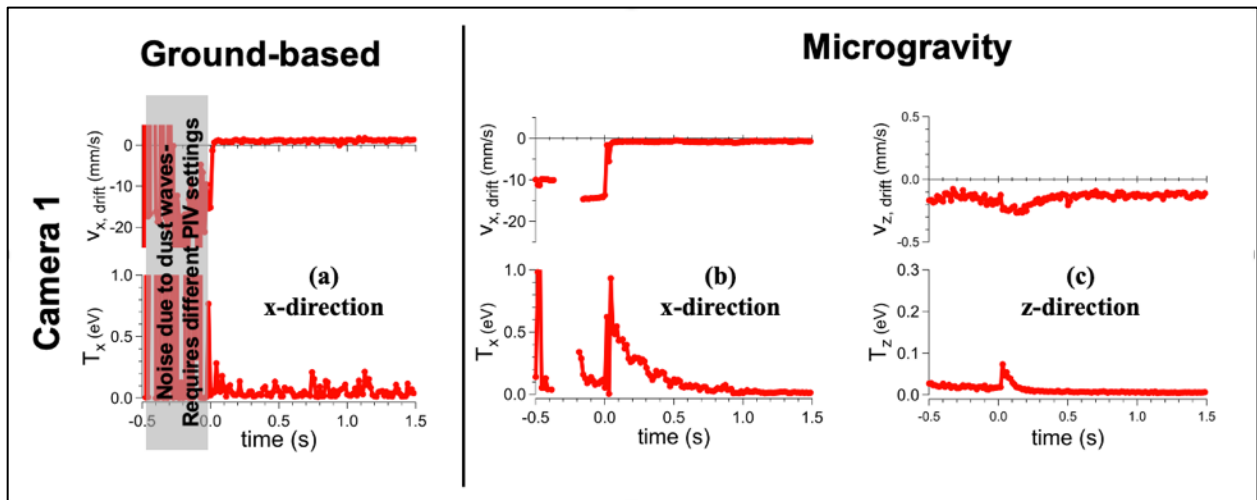


Figure 3-4: Comparison of velocity and dust kinetic temperature measurements from ground and microgravity PK-4 experiments for camera 1 (red). (a) is the ground-based x-direction, (b) is the microgravity x-direction, and (c) is the microgravity z-direction; top graphs are the drift velocity, and bottom graphs are the dust cloud effective temperatures. All x-axes are normalized to  $t = 0$  s corresponding to the application of polarity switching.

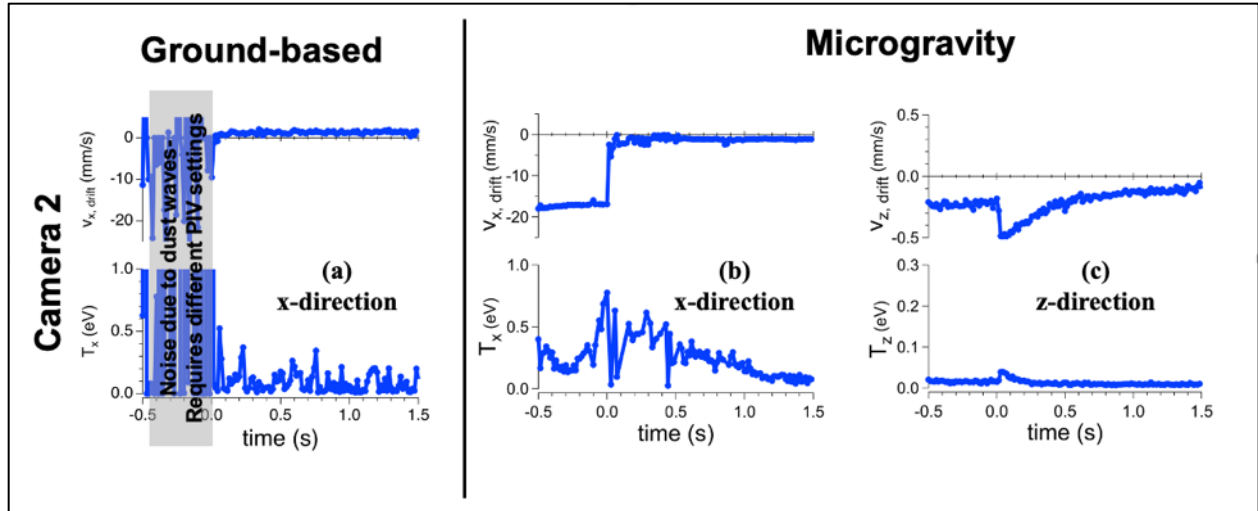


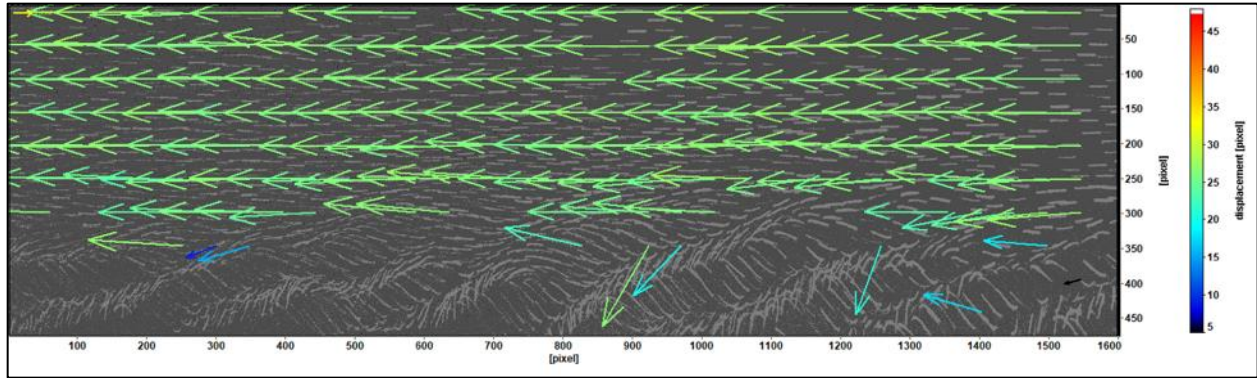
Figure 3-5: Comparison of velocity and dust kinetic temperature measurements from ground and microgravity PK-4 experiments for camera 2 (blue). (a) is the ground-based x-direction, (b) is the microgravity x-direction, and (c) is the microgravity z-direction; top graphs are the drift velocity, and bottom graphs are the dust cloud effective temperatures. All x-axes are normalized to  $t = 0$  s corresponding to the application of polarity switching.

The vectors returned from PIV are binned into histograms and fit using a Maxwell-Boltzmann distribution. From this distribution, we extract the drift velocity,  $v_{drift}$  and the kinetic temperature (this analysis process is described in detail in 2.2.2), and these are plotted as a function of time in Figure 3-4. This shows the extracted drift velocity and cloud temperatures from the histograms in the  $x$ - (i.e., parallel with the axial electric field) and  $z$ - (i.e., transverse to the axial electric field) directions, plotted as a function of time for both the gravity (Figure 3-4a) and microgravity (Figure 3-4b and Figure 3-4c) experiments from camera 1, and gravity (Figure 3-5a) and microgravity (Figure 3-5b and Figure 3-5c) experiments for camera 2.

Starting with the ground-based observations in Figure 3-4(a), it is observed that within 1 - 2 video frames after polarity switching (i.e.,  $\Delta t \leq 0.028$  s), there is a decrease in the velocity of the particles from their drift speed from  $v_{drift} \sim 10\text{-}15$  mm/s to  $v_{drift} < 1$  mm/s (Figure 3-4(a) top and

Figure 3-4(b) top). The time scale for this reduction in the drift velocity is generally consistent with the slowing of the dust particles due to collisions with the neutral atoms, i.e., Epstein drag, where the dust-neutral collision frequency is estimated to be  $f_{epstein} = 88.3 \text{ s}^{-1}$ . From this, assuming that the particle drift will become damped by the neutral drag, this corresponds to a damping time  $1/f_{epstein} \sim 0.011 \text{ s}$ , equivalent to  $\sim 1$  video frame, which is consistent with the experimental observations.

For the gravity-based measurements, the presence of waves in the lower part of the cloud is observed. Because our goal is to investigate the conditions of the flowing particles, the PIV analysis configuration was optimized for the “flow” and “capture” phases of PK-4 and not for the waves, as illustrated in Figure 3-6 (which is also Figure 2-9, and repeated in this chapter for convenience). Therefore, when the waves are present, we have less accurate flow and temperature measurements before  $t = 0 \text{ s}$ , and this data is overlaid with gray boxes in Figure 3-4(a) and Figure 3-5(a). In the gravity-based  $x$ -direction temperature (Figure 3-4(a), bottom) there appears to be a momentary increase in temperature, and then the temperature quickly drops back to ambient dust cloud temperatures. These results are consistent with our preliminary ground-based PK-4 work and motivated our work to determine whether microgravity conditions would allow us to reveal additional details of the apparent change in the thermal properties of the dust cloud that may be occurring at the onset of polarity switching.



**Figure 3-6: A sample ground-based PIV resultant vector field, overlapping the original data frame, to illustrate that the waves in the bottom of the field-of-view do not yield (reasonable) vectors when using the “flow” settings. Note, the background image contrast is saturated to make the waves more visible in the lower part of the image. This is a repeated image of Figure 2-9.**

Figure 3-4(b,c) and Figure 3-5(b,c) show measurements of the x- (along the axial electric field) and z- (perpendicular to the axial electric field) components of the particle response to polarity switching for an experiment performed under microgravity conditions in the ISS, for camera 1 and camera 2, respectively. It is first noted in Figure 3-4(b) and Figure 3-5(b) that there is a decrease in the horizontal (x-component) drift velocity of the particles that nearly exactly matches the ground-based experiments in Figure 3-4 (a) and Figure 3-5(a). Since both experiments are performed under the same gas pressure and discharge current conditions, we can conclude that, with respect to the flow, in both the ground-based and microgravity experiments, the particle drift decays in 1 to 2 video frames on a time scale that is consistent with Epstein drag. However, in terms of the thermal response of the system, there is a substantial difference between the ground and microgravity systems.

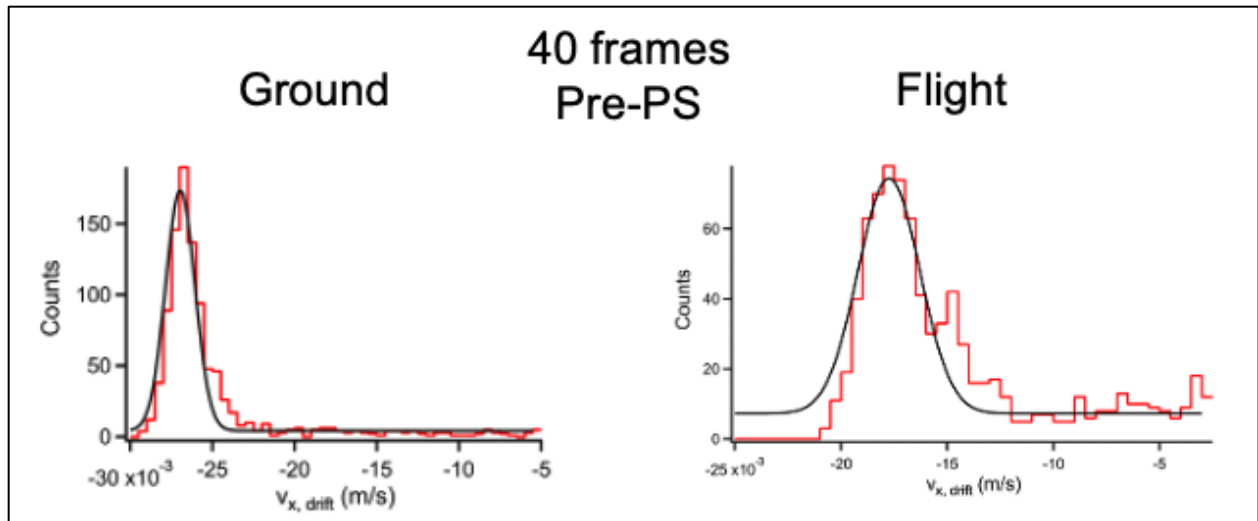
The microgravity-based x-direction temperature (Figure 3-4 (b) and 3-5(b), bottom) shows a large rise in magnitude and then an extended time for this apparent heating to dissipate. A careful examination of the PIV settings was performed to ensure that this was not an experimental or



analytical artifact (further discussed in the next section), and several analysis approaches are discussed below to confirm the validity of these observations. The microgravity z-direction temperature (Figure 3-4(c) and 3-5(c), bottom) also shows a smaller heating and dissipation event, which may indicate an additional dissipation of the flow kinetic energy into both the parallel and perpendicular directions (relative to the flow) at the application of polarity switching. These responses vary slightly from the two cameras, but the response trends are present throughout the dust cloud, indicating this is not an isolated occurrence.

### **3.2.1. Histogram Benchmarking**

In the initial results presented, there is an apparent difference in temperature observed between ground and flight-based results, and a difference from previous ground-based experiments as well. Our first step is to make sure those differences are not artificially induced from analysis techniques. This is investigated by evaluating the resultant histograms in detail, in many locations throughout the experimental data. In doing so, we found that some frames were returning inaccurate fits that will be elaborated on further in this section. Ultimately, this benchmarking has created a qualitative refining process to our analysis technique that will be presented in this section to ensure our analysis is accurate demonstration of the dust cloud characteristics.



**Figure 3-7: Sample histograms of the  $v_x$  components yielded from PIV from ground (left) and flight (right) experiments pre-polarity switching. Both histograms are for 40 frames prior to the application of polarity switching, for the same sample plasma conditions,  $p = 0.6$  mBar,  $I = 0.7$  mA. There is a large difference in the distributions, ground-based is symmetric and thinner width, but the flight distribution might show two peaks and has a larger drift velocity.**

First, we will compare histograms of approximately the same frame for the dust cloud in both experiment locations. Figure 3-7 shows the comparison of ground-based and microgravity experiments at approximately the same time, 40 frames prior to the application of polarity switching in each dataset, where we would expect the background plasma conditions to be comparable between ground and microgravity. The ground-based experiment has a symmetric histogram, with a thin width (temperature). The flight-based experiment has a larger width and might even have a second peak at  $-15$  mm/s, that is not incorporated into the MB fit of the distribution (black line). This led us to examine additional frames for the microgravity experiment to see if this was a trend that could be artificially yielding heating results.

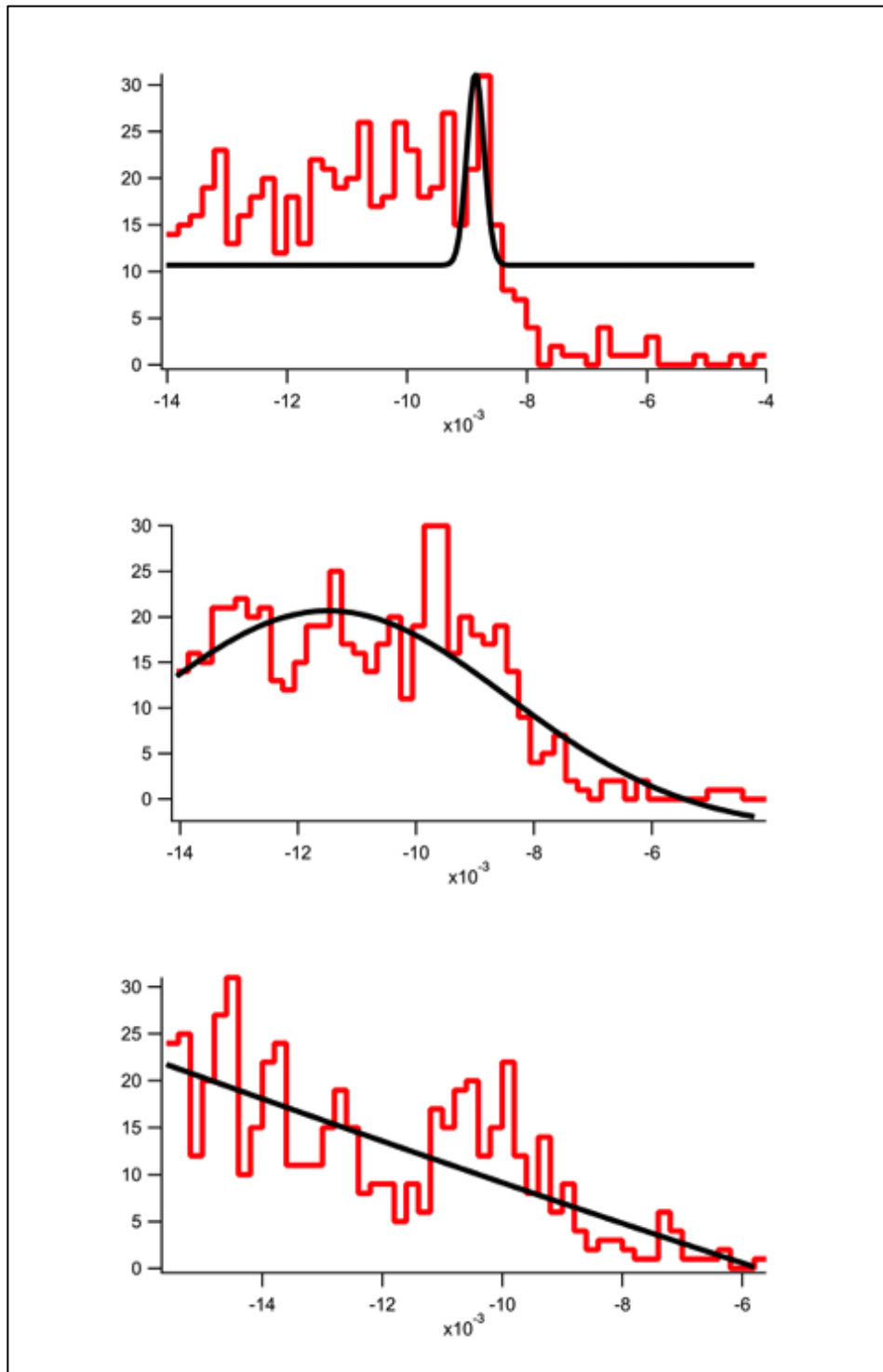
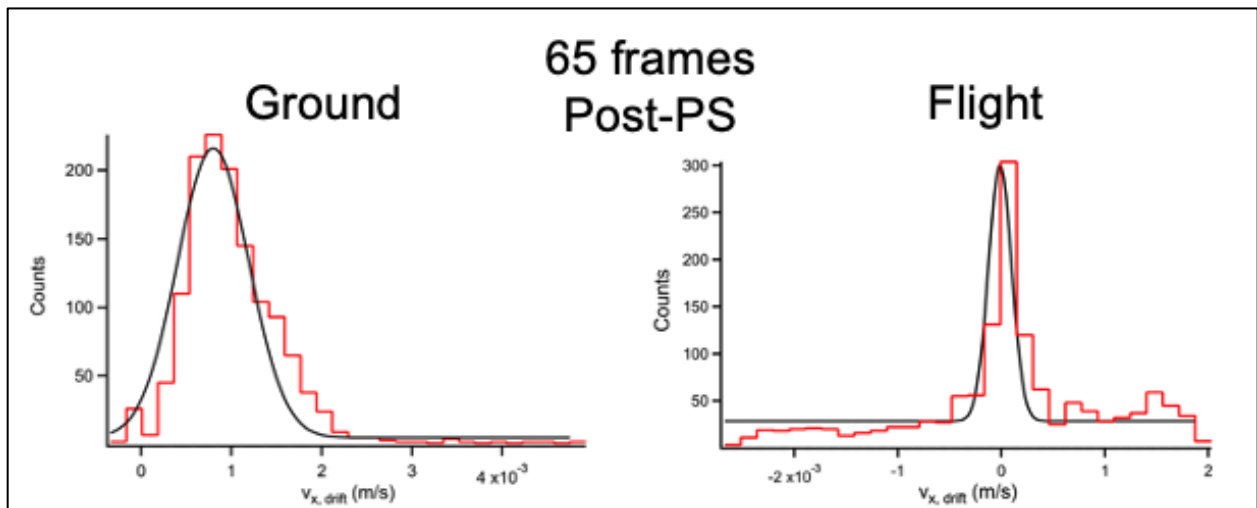


Figure 3-8: Sample injection segment frames' histograms (red) for the x-direction velocity components which yield "bad" MB fits (black).

Besides the second peak in Figure 3-7 not being included in the fit, we also found many other frames with unreasonably high or low reconstructed temperatures, seen in Figure 3-8. We were able to clean some frames up, by refining our PIV settings. However, there were other cases that were unsalvageable due to either large streaking particles or not enough vectors to reconstruct a velocity distribution arising from the split-nature of the cloud for the  $p = 0.6$  mBar,  $I = 0.7$  mA dataset. The frames that were not able to be cleaned and verified were systematically removed (i.e. values less than room temperature, or 10x larger than neighboring frames' values) in the data seen in Figure 3-4(b), hence the gap in data pre-polarity switching.

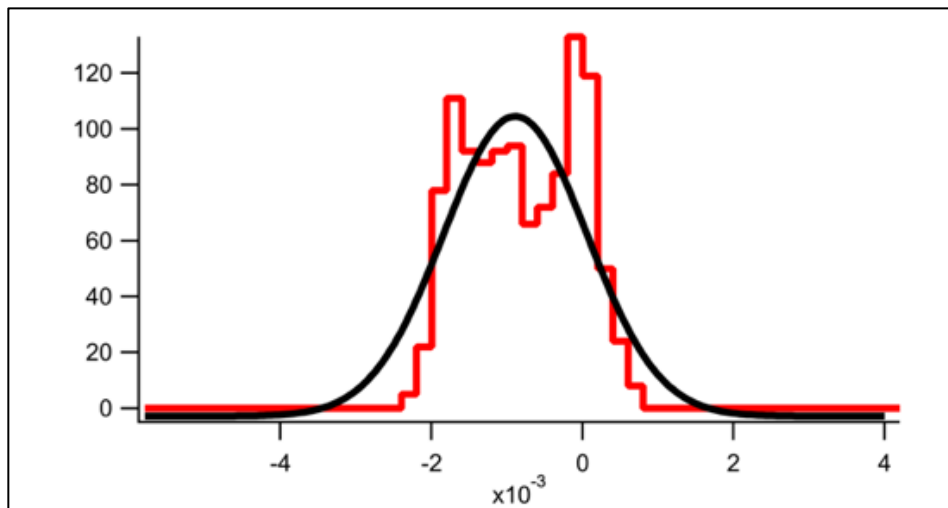


**Figure 3-9: Sample histograms of the  $v_x$  components yielded from PIV from ground (left) and flight (right) experiments post-polarity switching. Both histograms are for 65 frames after the application of polarity switching, for the same sample plasma conditions,  $p = 0.6$  mBar,  $I = 0.7$  mA. While the ground distribution has a slight drift velocity, they are in agreeable comparison of width (note the difference in counts on the vertical axis). This is after the heating has dissipated in the flight dataset (which lasts  $\sim 40$  frames).**

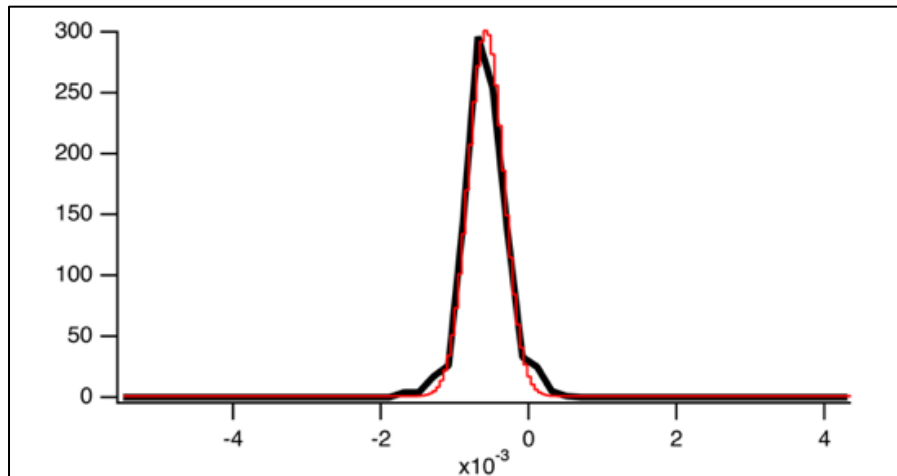
To investigate this difference in ground versus microgravity experiments' histograms further, consider a post-polarity frame. Since we know the heating occurs for about 40 frames in the microgravity dataset, based on Figure 3-4b, we will look at a histogram from a frame after that

decay has finished, arbitrarily set at 65 frames after polarity switching. Figure 3-9 shows the comparison between ground and flight experiment's post-polarity (and dissipation of temperature) distributions. Data for both cases appear to be in better agreement with the MB fit.

Figure 3-10 shows another frames' x-direction histogram (red) post-polarity switching. The fit (black) attempts to fit both peaks shown in the figure, leading to an apparently large width of the distribution, which would be interpreted as a "large" temperature. However, the large peak at 0 mm/s is artificial which can arise from "pixel locking" during the PIV processing. Pixel locking is a consequence of very slow moving particles whose frame-to-frame displacement is smaller than the PIV resolution limit ( $\sim 0.1 - 0.2$  pixels) and returns a displacement of 0 pixels. In fact, many frames with unreasonable distributions or split histograms showed a peak at  $v = 0$  mm/s. To resolve this, we modified our vector post-processing code to remove "locked" vectors that have a component magnitude of less than 0.1 pixels. This helped improve the results, as seen in Figure 3-11, which is a much cleaner fit of the same frame of data after refining the analysis techniques.

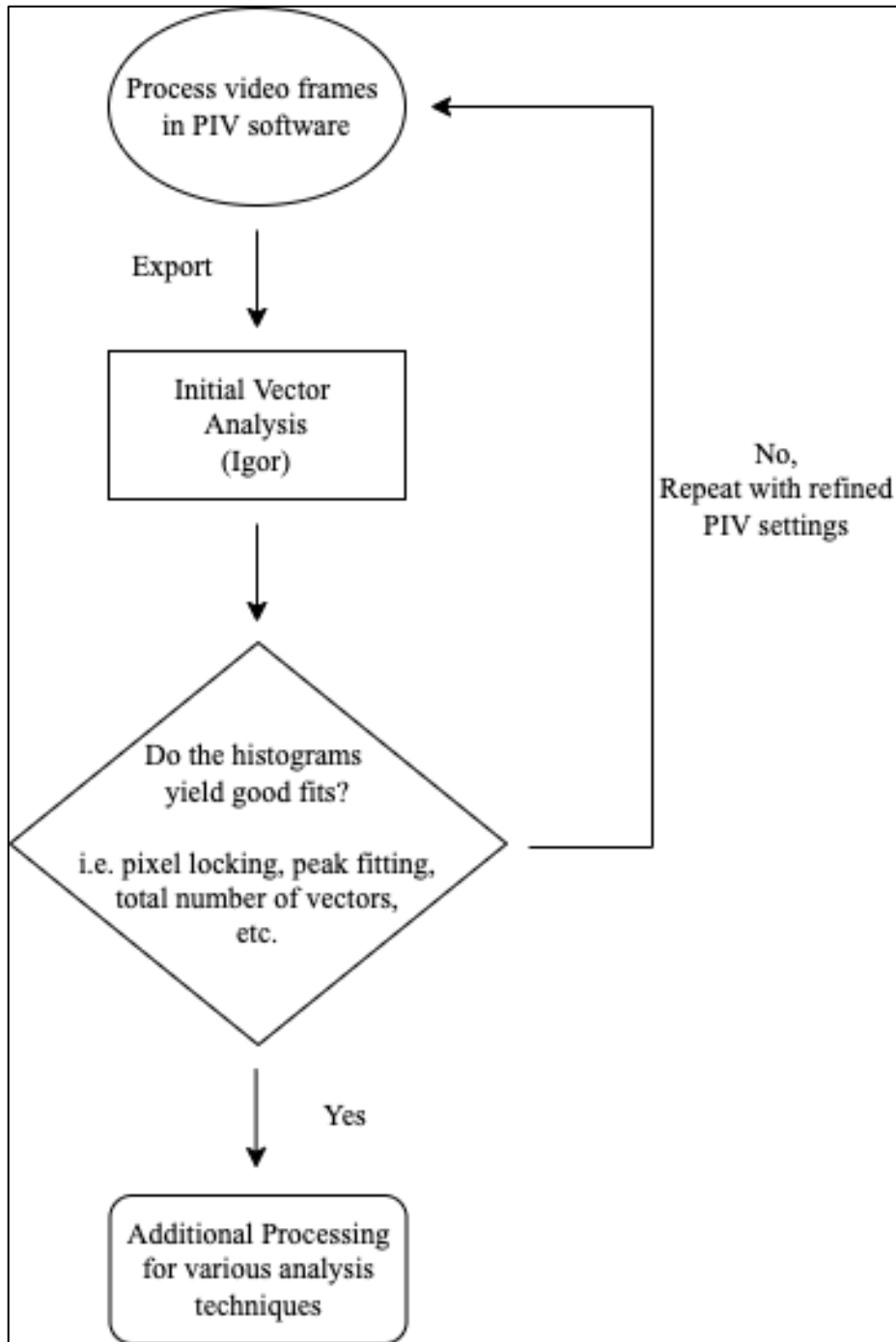


**Figure 3-10: Invalid fit of a histogram during the captured segment of the microgravity  $p = 0.6$  mBar,  $I = 0.7$  mA dataset. The artificial peak at 0 is due to pixel-locking of the PIV software. We can remove the vectors with  $|v| < 0.1$  pixels, which would mean the result was artificial due to PIV pixel locking.**



**Figure 3-11: Improved fit after a redo of the same frame for the histogram in Figure 3-10. The better results are after rerunning the PIV, removing more vectors due to pixel locking, and repeating our analysis techniques.**

The histogram benchmarking process presented in this section can be summarized in a qualitative flowchart, seen in Figure 3-12. To achieve reliable results, the PIV process may require multiple iterations to refine the settings. But once we are confident in our vector field and the subsequent histogram MB fits that are used for post-processing, we can therefore be confident in our drift velocity and temperature results as well. Upon further investigation at a multitude of individual frames' histograms for a variety of plasma conditions and times throughout the experiment, we can conclude that the heating is not artificial. Even with all of these corrections we have presented, the presence of fast-moving, “streaking” particles still cause histograms pre-polarity switching to occasionally yield invalid fits. Nonetheless, these procedures can yield consistent and reproducible measurements of the dust particle drift velocities and dust kinetic temperatures through a polarity switching event. This gives us a high degree of confidence in the experimental observation of a dust kinetic temperature rise and an extended decay after the application of polarity switching, shown initially (Figure 3-4 and Figure 3-5).

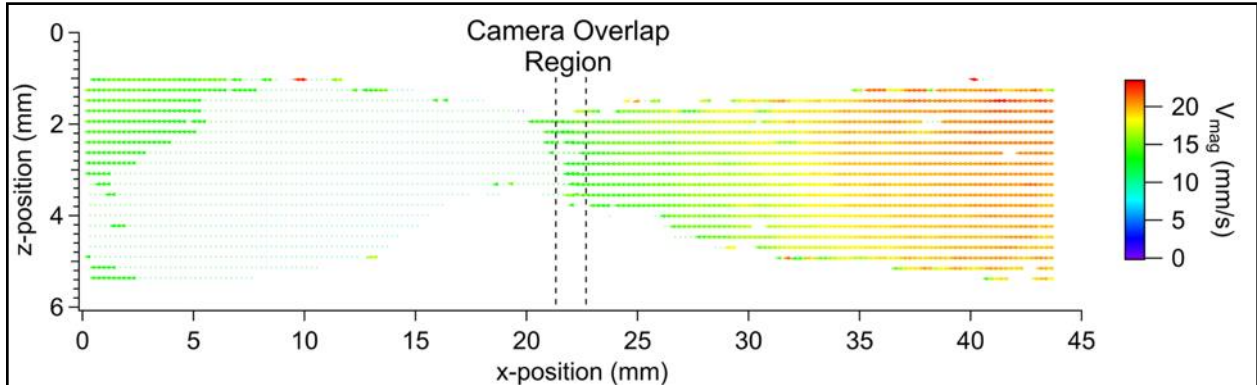


**Figure 3-12: Qualitative PIV analysis technique flowchart. We focus on confirmed the results of histograms are good fits before we do our further analysis.**

Furthermore, we are confident in these experimental heating and extended dissipation results since they occur after polarity switching, where the fits are more well-behaved. We also

reprocessed the PIV with different settings, and the post-polarity switching section would always yield similar results. Since the results are real and not an artifact of the histograms, we must now we now consider an alternative approach to confirm that the observed heating and extended dissipation of energy in the experiment is a valid interpretation of these results.

### 3.2.2. Subdividing into Regions

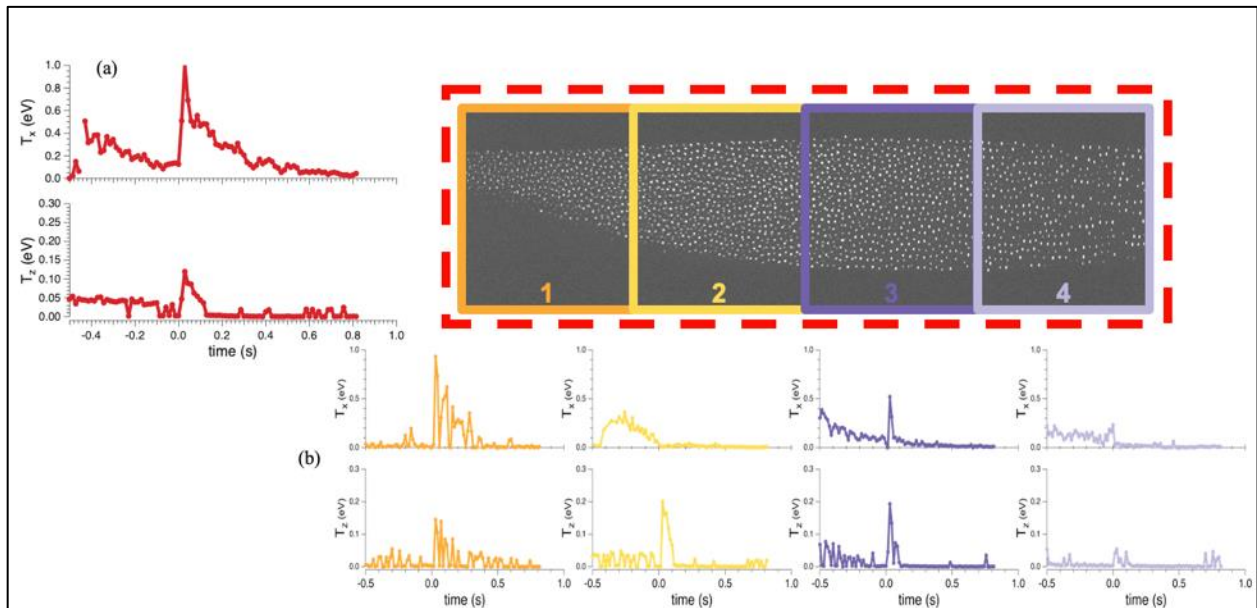


**Figure 3-13: A resultant velocity vector field from the  $p = 0.6$  mBar,  $I = 0.7$  mA dataset, combining PO1 and PO2 for a full Field of View. This dust cloud was split into two smaller clouds at capture, which is why there are a smaller number of vectors (and smaller magnitude) from  $x = 10 - 20$  mm. Even looking at the “second” cloud to the right, it is visible that there is a large range in velocities, and dividing the frame into sub-regions provides additional insights to the dust cloud dynamics than averaging the vector results over an entire frame.**

Figure 3-13 shows the resultant vector field when both cameras are processed through PIV and then combined appropriately. The region  $x = 0 - 22$  mm is from PO Camera 1 and  $x = 22 - 44$  mm is from PO Camera 2. This particular pair of frames had a split in the cloud, which is why there are not many vectors returned in the region  $x = 10 - 20$  mm. It is important to note the varying magnitude of velocity in the right half of the cloud. This magnitude variance indicates a spatially non-uniform response to the application of polarity switching. To better account for the variations in the dust cloud distribution functions, we repeated our initial analysis processes from



the MB fits using a sub-divided regions approach, to examine the energy dissipation throughout the dust cloud at the application of polarity switching.



**Figure 3-14:** a) The same dust cloud temperature data from camera 1 presented in Figure 3 b and c, reorganized to have  $T_x$  on the top, and  $T_z$  on the bottom. b) A sample frame divided into 4 regions, and the boxes marking the regions match the color to the region’s temperature data, directly below. All 4 regions show a rise in temperature at the application of polarity switching, but there is a non-uniform response in the magnitude of the temperature increase. This suggests a non-uniform response throughout the cloud to the application of polarity switching. Note the difference in the temperature axes scales between  $T_x$  and  $T_z$ .

A second approach to investigating the cause of the rise in temperature at the application of polarity switching is to identify the spatial non-uniformity by dividing the frames into four regions and repeating the analysis techniques, as seen described in Figure 3-4. The red data shown in Figure 3-14 is the original camera 1 microgravity data from Figure 3-4b, and the four other datasets correspond to a specific subregion as indicated by the dust cloud frame directly above. By comparing the individual regions and whole field datasets, the rise in temperature is coming from

only part of the cloud, in this case, most strongly from regions 1-3. This further indicates that there is a non-uniform response in the dust cloud at the application of polarity switching as well as confirming that the heating at the onset of polarity switching throughout the cloud is real in the experiment, and not an analysis artifact.

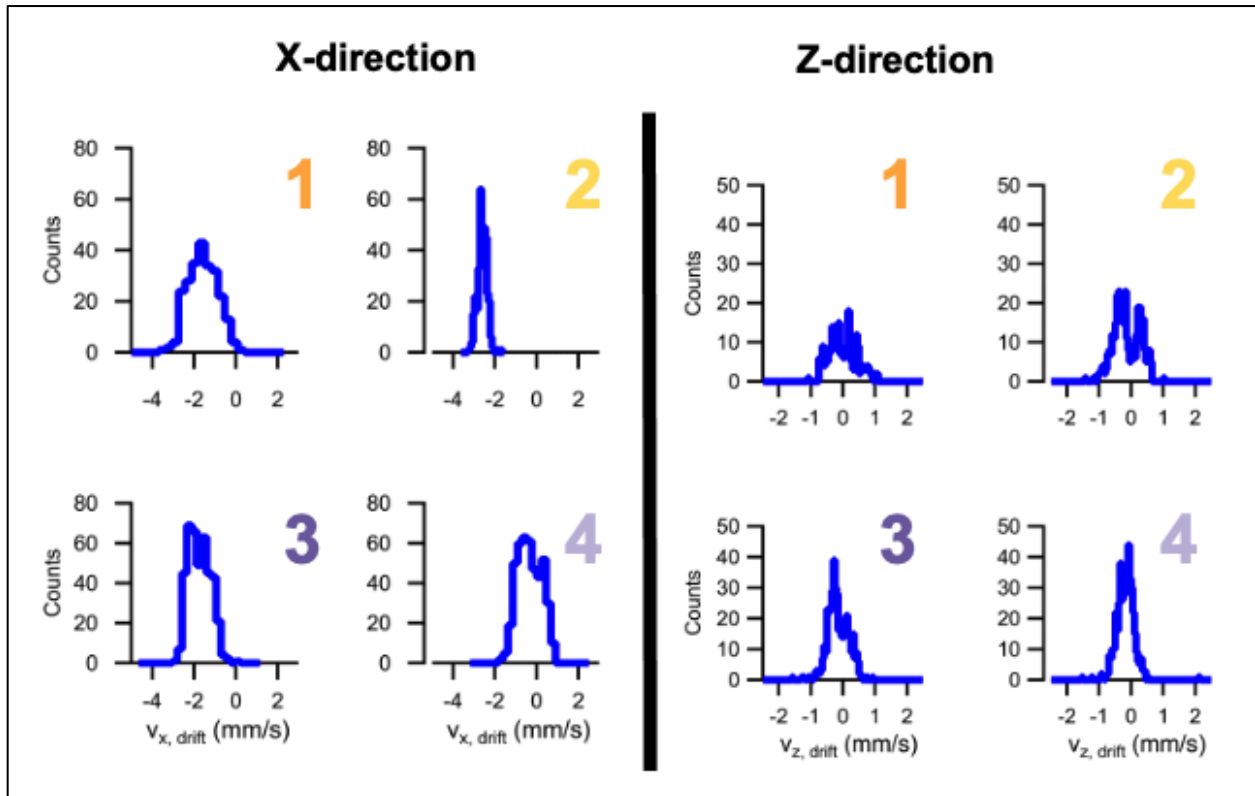


Figure 3-15: using the same regions shown in Figure 3-14, we can look at a single frame’s region histograms to further show the variation in regions’ results. In the x direction, the width of region 2 is much smaller than the other regions, and there are variations in all drift velocities. In the z-direction, the peak in section 4 is much larger, with a smaller width compared to the others, but all drift velocities are around 0 mm/s.

To further confirm the regions results of Figure 3-14, we can look at a single frame’s histograms for each region in more detail, as shown in Figure 3-15. The x-direction region 2 histogram is consistently thinner throughout the dataset, which agrees with the lower temperatures

in Figure 3-14. The z-direction regions 1 and 2 are created from fewer vectors than the other (consistent with Figure 3-13) and therefore have more variance and width in the distribution, which may artificially increase the temperature measurement. This is why we want to return as many vectors as possible so that the distributions are clean and return reliable results. Therefore, the sub-region analysis provides a useful confirmation that within distinct regions of the particle cloud, there is clear evidence that polarity switch is leading to an increase in the measured dust kinetic temperature. However, this technique does significantly reduce the number of available vectors for the velocity distributions, which - itself - could introduce additional errors in the analysis. Therefore, while this approach provides a useful test to confirm the apparent temperature rise, much of our subsequent analysis will focus on “whole cloud” distributions because they can provide a large number of vectors for the distribution function reconstruction.

### **3.3. All Datasets from Campaign 7**

Now that we have confirmed the heating and extended dissipation is not an artifact, but can arise from smaller regions within the cloud, we can begin to look at datasets from other plasma operating conditions in our campaign 7 experiment. First, we will compare the whole field results for all datasets with both cameras, similar to Figure 3-4.

Figure 3-16 shows only the  $T_x$  values from a whole field analysis, to help focus on finding when extended dissipation of energy occurs. The red data is from camera 1 and the blue data is from camera 2 for each dataset, where the plasma operating conditions are labeled on the top (current) and side (pressure). By using the same PIV analysis settings and IGOR macro processing codes, we discovered that sometimes the heating occurs and sometimes it does not. Table 3-2 summarize the results for which conditions there is heating and dissipation, with a “yes” quantified as taking at least 10 frames after polarity switching occurs for the heating to dissipate into the

system. Extended dissipation seen in both cameras is a green “yes”, seen in one camera is a yellow “yes”, and not seen in either camera is a red “no”. These colors will be used later in Section 3.5.

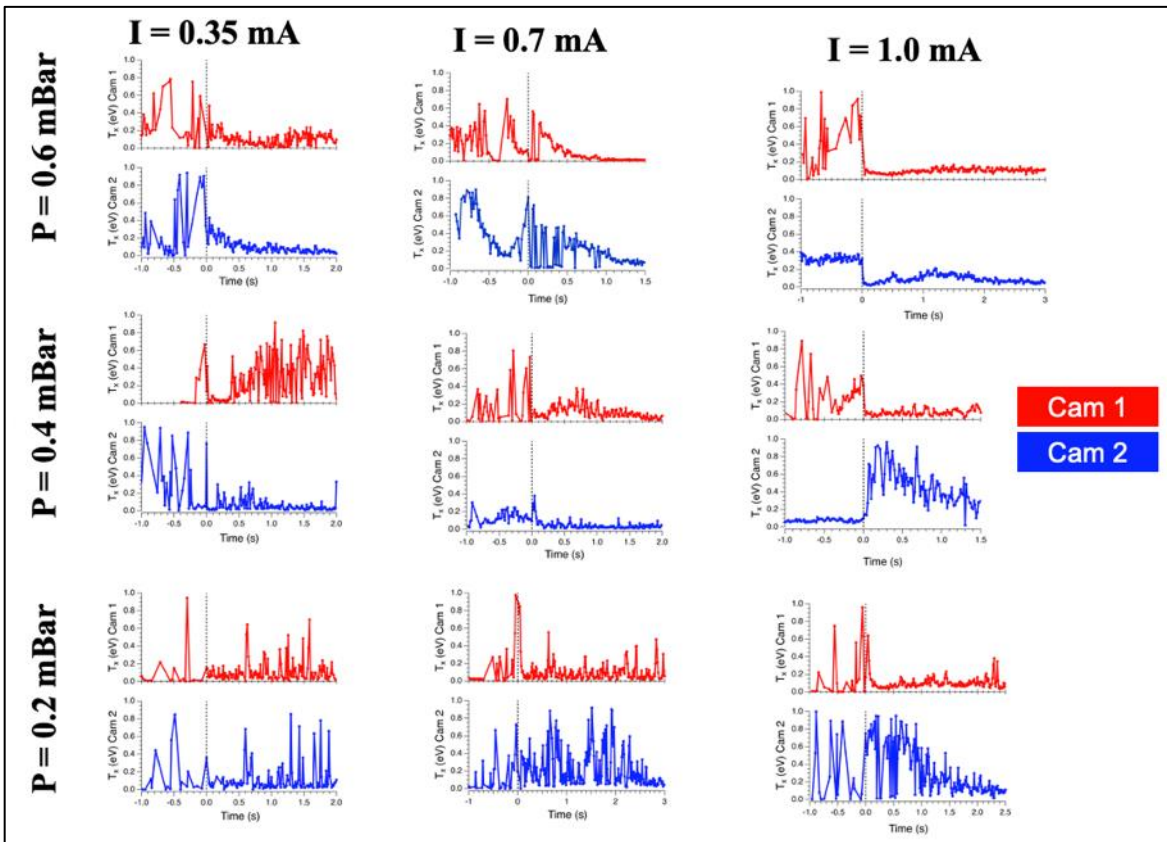


Figure 3-16: The whole-field x-direction temperatures for data from all 9 parameter spaces and both cameras. Red is camera 1, and blue is camera 2. In some parameter spaces, heating decay occurs in both cameras, in some spaces it only occurs in one camera (likely due to cloud location and number of vectors returned) and in other parameter spaces, the heating decay does not appear to occur at all. When decay is present in a parameter space appears to coincide with a small window of effective dust shielding lengths.

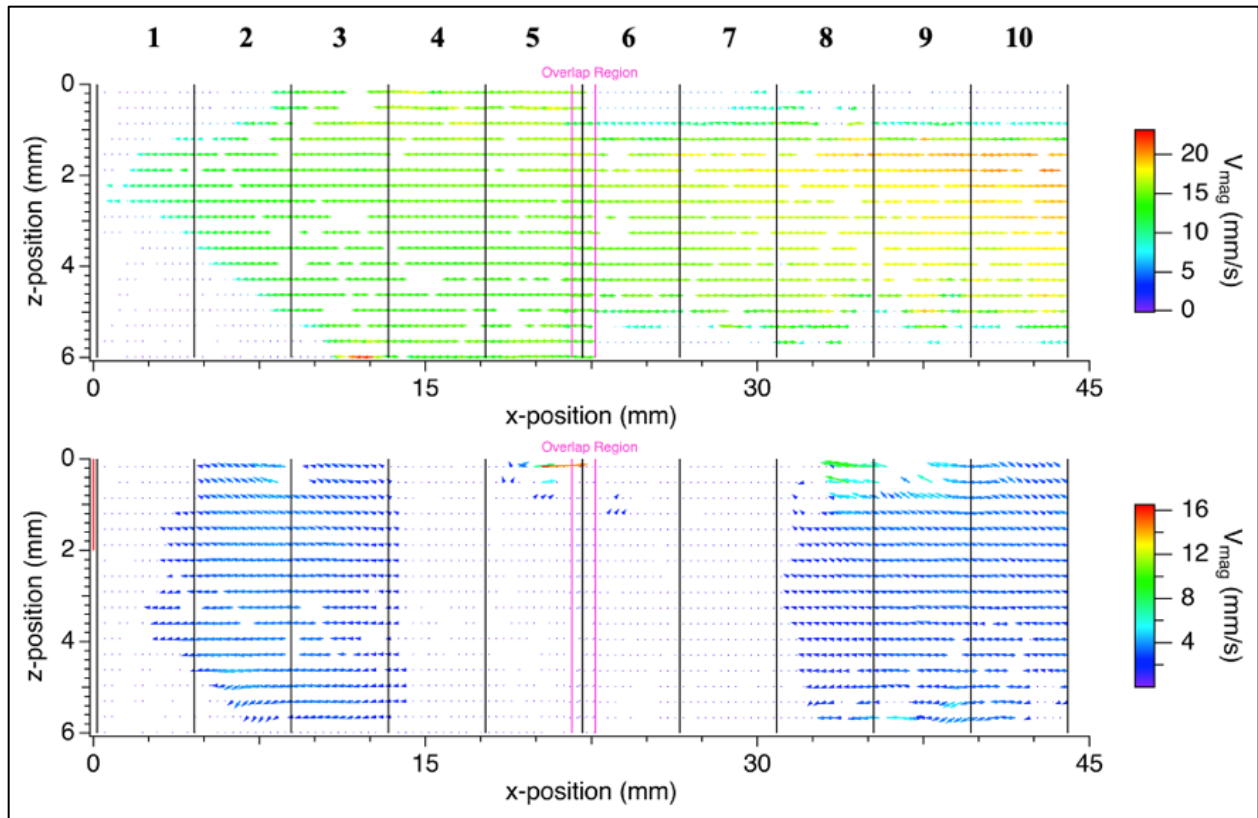
Extended Dissipation?	I = 0.35 mA	I = 0.7 mA	I = 1.0 mA
P = 0.6 mBar	One Cam	Yes, Both	No
P = 0.4 mBar	No	One Cam	One Cam
P = 0.2 mBar	No	No	Yes, Both

**Table 3-2: Summary of which plasma operating conditions show heating at the application of polarity switching in at least one camera in the microgravity campaign 7 experiment, shown in Figure 3-16. The yes conditions (either one or two) are quantified as such if the dissipation takes at least 10 frames.**

When we look at both cameras together, it also doesn't always show up in the entire cloud (both cameras), there are three instances where it is just one camera. We applied the regions analysis technique to one of these single camera results datasets to further determine why the cameras yield different energy dissipation.

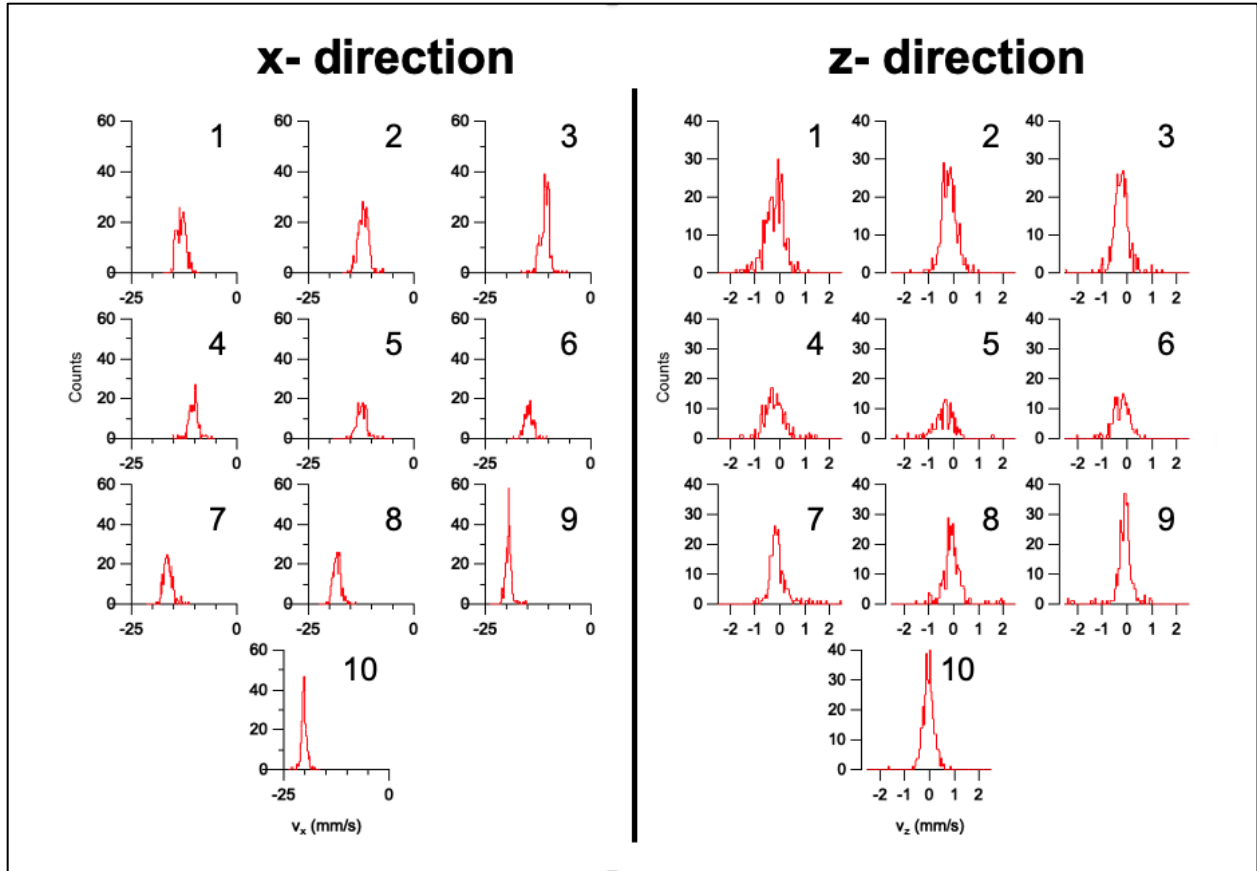
### 3.3.1. Regions Analysis for Additional Datasets

We want to further investigate a dataset where there is heating in one camera (PO2) but not the other (PO1), so this will look at dataset p = 0.6 mBar, I = 0.35 mA. First, we will look at select velocity fields for before and after polarity switching, seen in Figure 3-17. This cloud was a split cloud again, and it seems like the split occurred perfectly centered over the camera overlap region. The flowing frame (top) is the "left" cloud that has a majority outside of the FOV when the dust cloud is captured. The back edge of the flowing cloud is what appears in PO1 when captured (bottom). And then the front edge of the cloud that was seen fully in FOV for other datasets is on the right side with a focus for PO2. This split in the cloud can also be seen in the overview PO3 camera shown in Figure 2-3.



**Figure 3-17: Sample vector fields for before polarity switching (top) and after capture (bottom). This helps show that the dust cloud is split, pretty significantly, yielding heating in camera 2 (right of overlap) but not camera 1 (left of overlap). This is the  $p = 0.6$  mBar,  $I = 0.35$  mA, the top left “whole frame” comparison in Figure 3-16. Note the labels of regions 1-10 at the top, this coordinates with the next figure.**

The gap between clouds being in the FOV for capture also leads to a wide variation in the histograms for each of the 10 regions, shown in Figure 3-18. Note, each region is numbered above in Figure 3-17 and labeled individual in Figure 3-18. It can be seen that regions 2 and 3 are the only regions of camera 1 with enough vectors to create reliable histograms in the x-direction, but their results are still different. And the drift velocities for all 10 regions are slightly different. For the z-direction, the histograms are more similar in drift velocity, but the number of vectors still plays a significant role in the temperature values returned by the fits.



**Figure 3-18: Histograms in the captured frames' (bottom) 10 regions in Figure 3-17 for the x-direction (left) and z-direction (right). The numbers 1-10 for the regions in this figure correspond to the numbers at the top of Figure 3-17.**

Ultimately, this shows that the cloud is not necessarily in the camera 1 FOV for the  $p = 0.6$  mBar,  $I = 0.35$  mA dataset. This accounts for the biggest discrepancies between cameras in the plasma conditions where at least one camera has extended dissipation of heating. The small difference in having two full regions with vectors (regions 2-3 for PO1) and three full regions with vectors (regions 8-10 for PO2) can be the difference to have enough vectors to highlight the heating and dissipation in the system at the application of polarity switching.

### **3.4. Other Experimental Insights**

The campaign 7 experiment capturing data is a great first step at determining when heating and dissipation occur in the dust cloud. We also have data for a select few other limited experimental excerpts from other campaigns, such as the downlinked data from campaign 12, performed from Auburn's campus in 2021, and other sections of the campaign 7 data. We will now look at these limited portions to further investigate when this phenomenon might occur in the PK-4 microgravity experiment.

#### **3.4.1. Campaign 12 Reinjections**

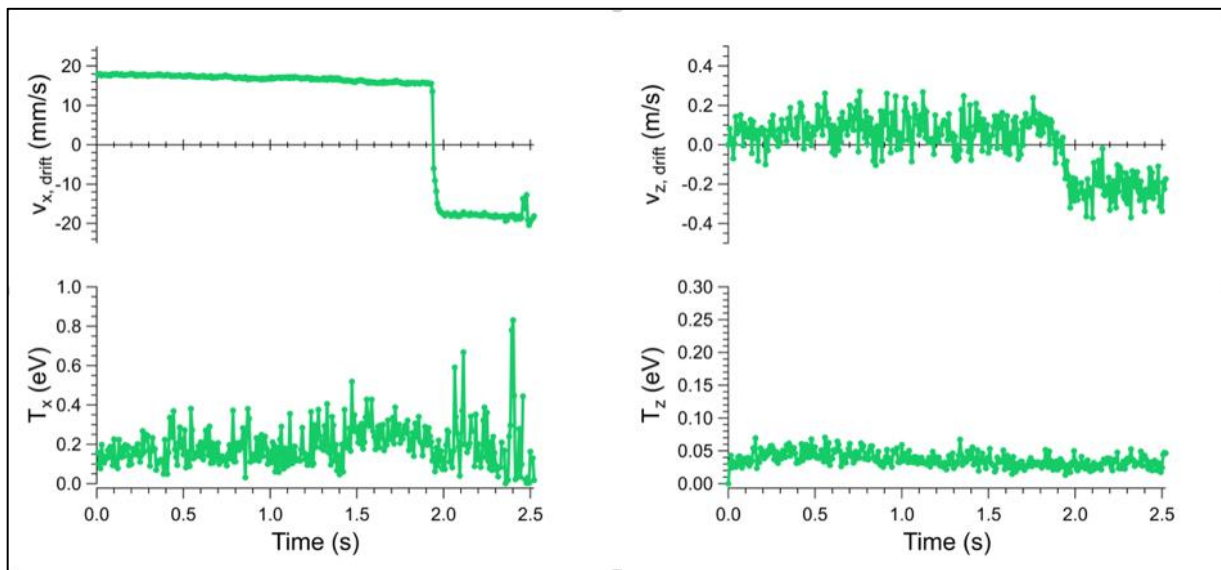
Since the diagnostics of the on-board experiment are limited, we proposed a follow-up experiment to further investigate this heating event that occurs at the application of polarity switching. This time, we requested a reduction in field-of-view in exchange for a higher framerate data collection, as previously mentioned in Section 2.1.3. This increase in temporal resolution was to give us more datapoints during the heating process at the application of polarity switching, and subsequent decay, and to hopefully be able to determine what happens in these transitions with more detail. The proposal and script flowchart for Campaign 12's experiment can be found in Appendix A.2.1 and A.2.2, respectively.

This experiment was run in June 2021, this time from the Physics Department in Auburn. Since we were not able to travel to CADMOS due to the COVID-19 pandemic, NASA MSFC (Marshall Space Flight Center) helped set up a datalink for us to perform the experiment from our offices. The initial data we thought we requested for downlink at the end of the campaign week was a replica  $p = 0.6$  mBar,  $I = 0.7$  mA injection and capture, to be a direct reference to our campaign 7 main dataset. But we actually received the reinjection turn around segment and the



dust cloud capture was not included. But as not to waste our limited amount of data, we still wanted to see if we gained any insights for this energy transfer phenomenon from this dataset.

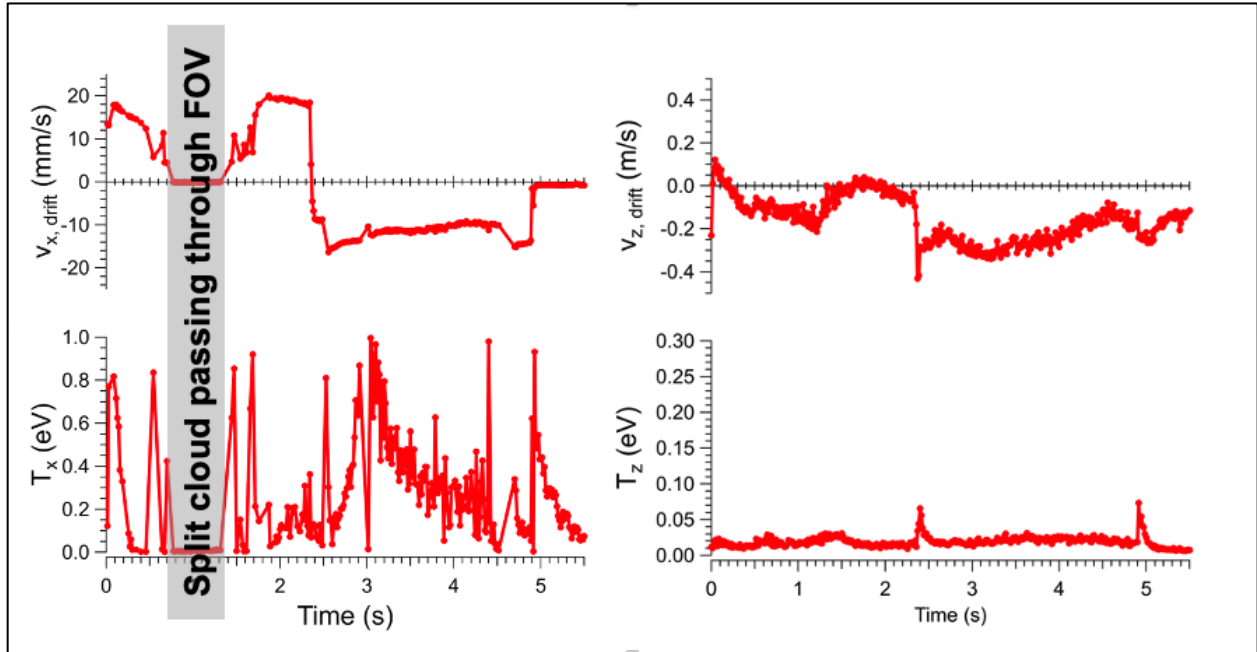
Reinjection minus is defined as such by setting the electric field to a negative (minus) value, therefore sending the negatively charged dust cloud to the right of the chamber. This results in a drift velocity of a positive value ( $v > 0$  mm/s). And subsequently, reinjection plus is when the electric field is set to a positive (plus) value, sending the dust cloud back into the left ( $v < 0$  mm/s) like we see with normal injection segments in the experiment.



**Figure 3-19: Campaign 12 whole field analysis for the reinjection turn around (flow from right to left) segment in microgravity. This is the  $p = 0.6$  mBar,  $I = 0.7$  mA dataset, and the entirety of our downlinked dataset.**

Figure 3-19 is the drift and temperature plot (same as Figure 3-4) for the entirety of our downlinked data (and the only campaign 12 data we have to date). This is a reinjection segment, indicated by the drift velocity magnitude switching from positive (left-moving cloud) to negative (right moving, typical injection direction). However, once we looked at this data, there still might be evidence of heating at the application of polarity reversal (when the drift velocity changes

magnitude,  $t \sim 1.8$  s). there is also a small change in the z-direction drift velocity, but we no longer see the spike in temperature that we saw in the capture datasets from campaign 7. This was an intriguing discovery, and we immediately went to look at our campaign 7 reinjection as well.



**Figure 3-20: Campaign 7 reinjection segment whole field. This shows both the reinjection minus and the reinjection plus segments to help put the data into perspective, as well as shows polarity switching capture around the  $t = 5$  s mark. The drop to 0 for  $t = 0.75 - 1.25$  s and then resuming normal magnitudes is due to the split cloud.**

By looking at the entirety of the campaign 7 reinjection segment (both minus and plus) as seen in Figure 3-20, we see there is a change in the temperature at the application of polarity reversal,  $t \sim 2.4$  s. we can also seen when the split in the cloud passes through the FOV, as the drift velocity and temperature both drop to 0, so this has been blurred out with the gray box in the figure. The heating and dissipation at  $t \sim 5$  s is the same data shown in Figure 3-4. We also see another spike in the z-direction at the polarity reversal segment, as well as at the time of capture. All of

this combined has guided us to an updated hypothesis: heating and extended dissipation can occur at any change in the electric field, not just when the dust cloud is captured with polarity switching.

### 3.4.2. Campaign 7 Polarity Stepdown

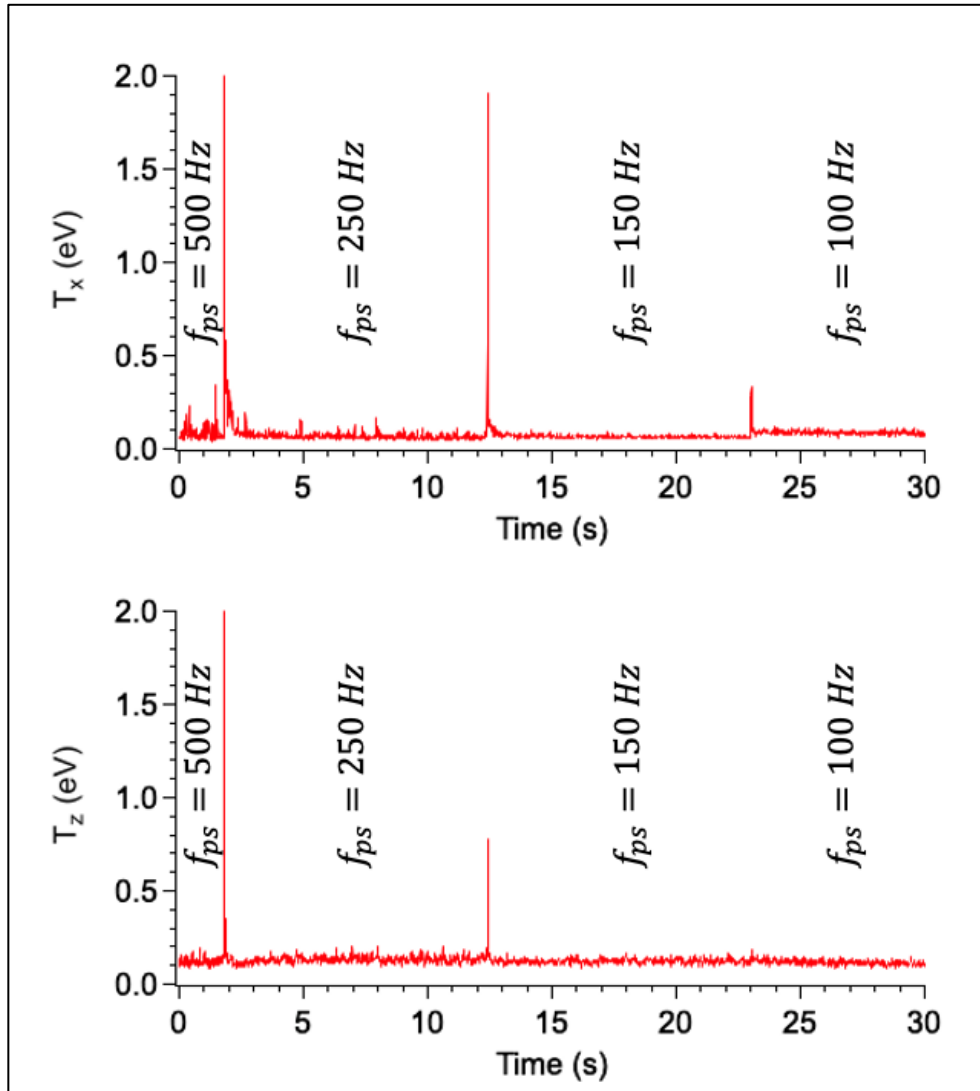
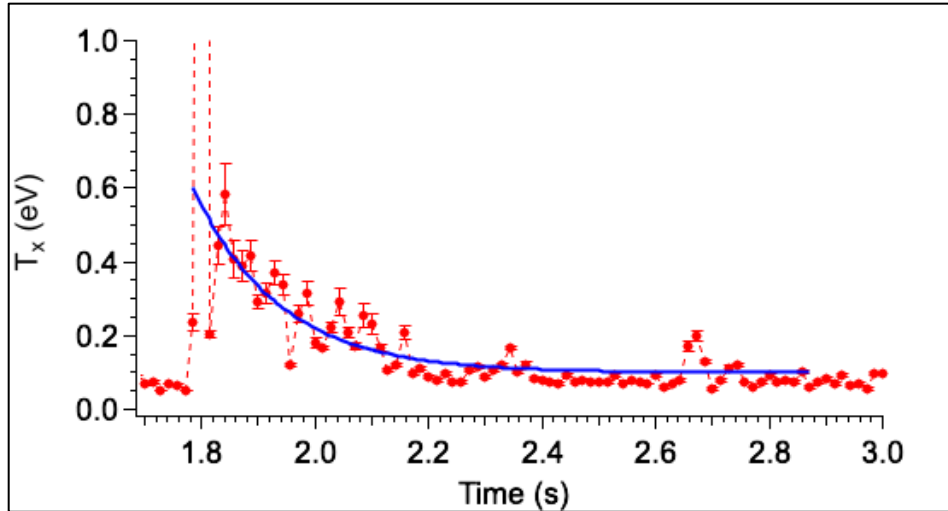


Figure 3-21:  $T_x$  and  $T_z$  as a function of time for the PS stepdown section of the Campaign 7 microgravity experiment. This dataset is the  $p = 0.6$  mBar,  $I = 1.0$  mA dataset. There is heating in both directions for the two first changes in PS, and heating in the x-direction for the third transition as well.

Another excerpt of data we have is the polarity switching stepdown segments at the end of a dataset. The goal of the “stepdown” of the polarity switching frequency was originally to

determine if dust-dust collisions could be induced with lower frequencies which could then be used as possible diagnostic measurements for the particles' charges. However, as will be described in this section, it was possible to use these changes in the polarity switching as a technique to obtain further information on the possible dust particle heating. This work was led by Dr. Jeremiah Williams and his students at Wittenberg University but helps support the work shown here. As we began to see this evidence of heating and dissipation, our two teams worked together to begin looking for additional evidence of dust particle heating throughout our datasets.

During the polarity switching frequency stepdown process, there are 5 changes of polarity switching. The polarity switching frequency is adjusted every 10 s in the following steps,  $f_{ps} = (500, 250, 150, 100, 50, 25)$  Hz. The first three changes, steps from 500 to 100 Hz, are seen in Figure 3-21 for the x- (top) and z- (bottom) direction temperatures. The last two changes in frequency caused the dust to shake very rapidly and the particle motion is difficult to analyze (and even look at, honestly), so we have not included this data. The particle motion is analyzed using PTV (Particle tracking velocimetry), and the dust particles' motion jumps from 500Hz to 250 Hz, so the magnitude of the temperature in the first frame after the polarity switching frequency change (highest magnitude point) likely has a greater error, but after that the PTV is able to track all particles, so the results are reliable. There is heating at each of the transitions in the x-direction and there is also heating at the first two transitions for the z-direction. So, we have heating at a change of an electric field, but what about the extended dissipation we are trying to characterize?



**Figure 3-22: A closer look at the PS 500 Hz to 250 Hz stepdown heating and dissipation in the x-direction from Figure 3-21. The heating magnitude matches that of the PS capture segment, with extended dissipation as well.**

To look at the temperature dissipation, we look closer at the data starting 2 frames after the large spike for the 500 to 250 Hz transition in Figure 3-21. This result is shown in Figure 3-22, and shows consistent decay in comparison to the dissipation of the heating from the capture data in Figure 3-4b (the microgravity x-direction data from camera 1), both lasting about 0.4s.

This work is ongoing by Dr. Jeremiah Williams and his group at Wittenberg University, so we have only shown one sample result. However, they have looked at all of the polarity switching stepdown datasets in campaign 7 and the summary can be found in Table 3-3. (We are also working on reproducing the heating in dissipation at all segments of the PS stepdown section with YOAK $\mu$ M). These results indicate that a change in the polarity switching appears to occur in most plasma conditions.

<b>Heating and Dissipation?</b>	<b>I = 0.35 mA</b>	<b>I = 0.7 mA</b>	<b>I = 1.0 mA</b>
<b>p = 0.6 mBar</b>	Yes	Yes	Yes
<b>p = 0.4 mBar</b>	No	Yes	Yes
<b>p = 0.2 mBar</b>	PS Stepdown Not Performed		

**Table 3-3: Summary of which plasma operating conditions show heating during a decrease of polarity switching frequency in at least one camera in the microgravity campaign 7 experiment.**

### **3.5. Experiment Discussion**

The presence of heating was expected, based on our initial ground-based experiments, and this was the basis of our proposal for our first experiment in campaign 7. When we began to process our microgravity results, we found the heating event as hoped, and dissipation into the perpendicular direction, and the occasional occurrence of the large extended time for this heating to dissipate back into the system.

Based on the analysis of other portions of the campaign 7 datasets and our downlink excerpt of campaign 12, this heating and extended dissipation can occur at any change in the electric field, if the plasma conditions are correct, not just the application of polarity switching. Since the plasma can react to the changes in the electric field quicker due to their inertia, the interactions of the changing plasma with the dust particles can be described by the dusts' effective screening length. If we look further at the plasma conditions for each dataset, we find that there is a possible trend between these extended dissipation occurrences and the calculated electron Debye lengths for the plasma conditions.

$\lambda_{\text{screening}}$ Value	I = 0.35 mA		I = 0.7 mA		I = 1.0 mA	
p = 0.6 mBar	■	2.188	■	1.481	■	1.235
	●		●		●	
p = 0.4 mBar	■	2.364	■	1.617	■	1.36
	●		●		●	
p = 0.2 mBar	■	3.060	■	2.060	■	1.733
	●		●		●	

**Table 3-4: Electron Debye length calculated based on plasma operating conditions. The boxes represent the results from Table 3-2 for the capture data, and the circles represent the results from Table 3-3 for the PS stepdown data. Green means there was heating and extended dissipation in both cameras (or present for PS stepdown), yellow means there was heating and dissipation in one camera, and red means there was no extended dissipation present. Furthermore, the conditions where the extended dissipation occur all are within the range of values 1.36 - 2.19 for the dust’s effective screening length.**

The summary of results for capturing datasets in Table 3-2 (squares) and PS stepdown segments in Table 3-3 (circles) can be compared against the calculated electron Debye length, an upper limit of our dust effective screening length. These comparisons are all compiled into Table 3-4. To remind the reader of the previous tables’ results we have added the color-coded shapes to this table as well. A green shape represents both cameras indicate heating and dissipation for the capturing data or a yes for the PS stepdown, a yellow square is one camera saw extended dissipation in the capture data, and red means no extended dissipation was present in the data. The gray circles are for p = 0.2 mBar, because the PS stepdown segment was not performed in that pressure operating conditions. The heating and extended dissipation effect appears to coincide with the dust particles’ effective screening length due to the changing plasma conditions, all occurrences happen if the dusts’ effective screening length is less than 2.2  $\mu\text{m}$ . Note, we did not see it in the p = 0.2 mBar, I = 0.7 mA, even though that screening length falls in the range, but that is likely due to the fact that there were waves in the capture segment, and PS stepdown was not

performed for  $p = 0.2$  mBar for total slot timing purposes. So, the conditions could be correct for heating and extended dissipation and we were just unable to see the results with our analysis.

The various datasets all corroborating the same initial experimental answer is further reassuring that this phenomenon is real. There is a change in the plasma when the electric field changes that creates heating in the dust cloud. And if the plasma operating conditions fall within a certain range of for the electron Debye length, the dust cloud will also exhibit extended dissipation of said heating in microgravity. This suggests the extended dissipation arises from interactions between the plasma and dust cloud that are on a timescale not typically seen in a complex plasma. Typically, we would expect the dust cloud's dissipation to be on the order of Epstein drag, or other interaction characteristics. However, the limits of the experimental data collecting capabilities of a "black-box" experiment (no probes, other benchmarking of plasma conditions without dust, etc.) do not allow us to experimentally investigate the reason for this extended timescale dissipation at the change of an electric field further, so we must now turn to simulations.



## Chapter 4: Numeric Model Simulations

In order to understand the experimental observations, our group has developed a Molecular Dynamics (MD) simulation code named YOAK $\mu$ M. The code framework was introduced in Section 2.3. Through the use of these MD simulations, we show that there are subtle interactions between the dust particles that give rise to the observed extended timescale for the dissipation of energy after a change in the electric field. The electrostatic potential energy arising from the spatial configuration of the charged dust grains in the dust cloud is proposed as a source to keep the net effective dust cloud temperature larger for longer than expected when compared against Epstein drag. This also suggests there is a modification of the effective dust cloud shielding length throughout the dissipation process. This chapter will discuss how we reproduce PK-4 in our YOAK $\mu$ M environment in Section 4.1 and 4.2, validate the code results in Section 4.3, and present the simulation results and discussion in Section 4.4.

In particular, we invite the reader to focus on Section 4.2. This is where the bulk of the physics for the simulation is located, arising from our core understanding of the experimental results. This is intended to describe our understanding of the underlying physical principles that are leading to the observed heating and subsequent dissipation in the dust cloud's temperature when there is a change in the electric field in the PK-4 system.

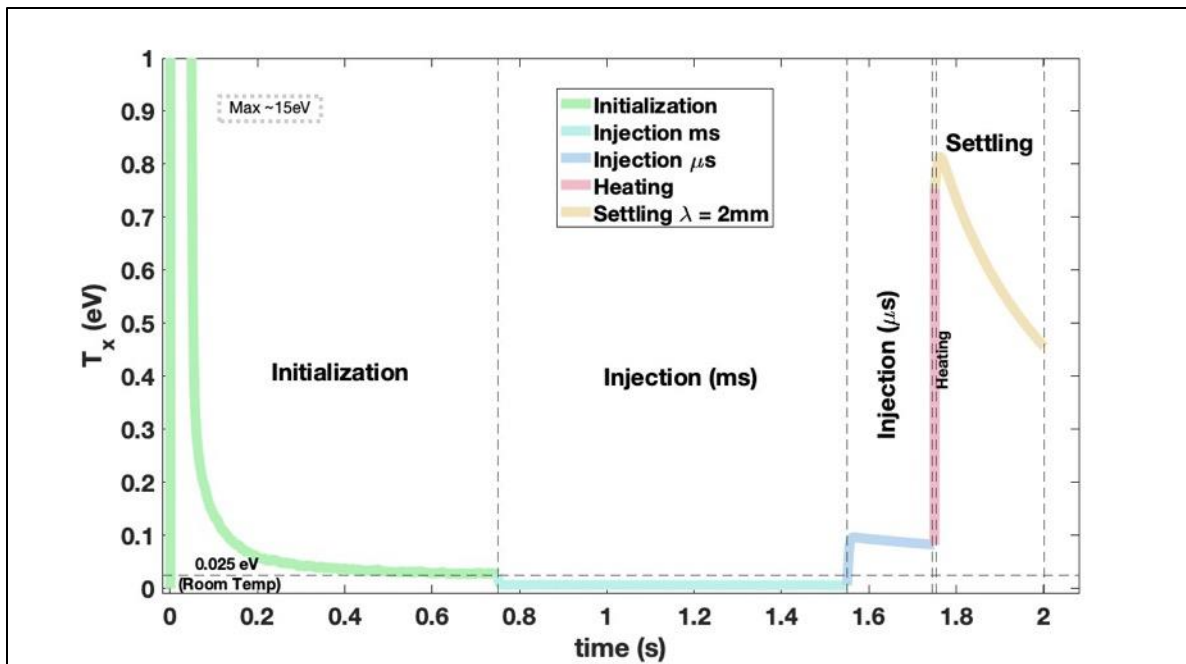
#### 4.1. YOAK $\mu$ M Step-by-step Replication of PK-4

To accurately represent PK-4 in YOAK $\mu$ M, first we need to create a dust cloud with qualitatively the same statistical and thermodynamic characteristics of the experimental dust clouds. This is a piecewise process, varying the fundamental forces involved in each time segment based on the experimental segments. The data at the final timestep of one segment serves as the initialization data for the next segment. First, we will look at the overview of the segmented results, and then in more detail will be presented in Section 4.1.1 – 4.2 below. We will describe how we replicate the experimental process and the applicable forces involved for a particular segment. The descriptions and equations for each of these forces can be found in full detail above in Section 1.4, *Select Forces in a Dusty Plasma*.

The first step of any code is initialization, in this case determining the dust particle positions and charges that will be used for our system. For this example dataset, we will use a dust cloud with 1000 particles in 2D, immersed in the same plasma conditions for the PK-4 operating settings,  $p = 0.6$  mBar, and  $I = 0.7$  mA, just like the example experimental dataset in Section 3.2, with charges on the order of the estimated OML charge values for the dust particles' size ( $\sim 10000e$ ). We have run simulations with 3D clouds and 2D simulation clouds with more particles to validate that this selected cloud simulation is representative of a larger cloud, and this comparison will be presented in Section 4.3.3. Ultimately, we chose to use 2D clouds with 1000 particles for optimal computational efficiency. For this MD simulation, we use traditional cartesian coordinates. So, the vertical direction throughout this chapter on simulations is “y” even though in the PK-4 experiment it was defined as “z”. This coordinate system choice was made so YOAK $\mu$ M could be used to replicate a variety of dusty plasma systems, not just the PK-4 experiment.

After initialization, we then recreate the “flowing” injection of PK-4, by applying an axial electric field. These first segments are important to create the proper drift velocity and dust cloud

temperature in the simulated cloud so that we can look at the effects of the polarity switching on the cloud. After the particles are injected, we then change the electric field to an oscillating electric field to stop the dust cloud and evaluate the physics behind the heating and extended dissipation of the experimental dust system. The overview dust temperature and drift velocity results of these three (three and a half) initial YOAK $\mu$ M steps can be found in Figure 4-1 and Figure 4-2, respectively. The reason we have “injection  $\mu$ s” (blue) lines will be described further in 4.1.2. These initial segments reproduce the experimental injection drift velocity and dust cloud temperature well and therefore we can use the end of these segments as the initial parameters for our heating and dissipation simulation investigation.



**Figure 4-1: Evolution of the dust cloud temperature during the preparation segments for the simulation, and one example “settling” segment. The dashed lines show the end of one segment and the beginning of the next segment.**

Figure 4-1 shows the evolution of the dust cloud temperature during the three (three and a half) initial condition runs in preparation for investigating the evolution of the dissipation of the

heating event in the dust cloud. The maximum value for the “initial” file is irrelevant to the final results simulation and is not shown for scaling purposes. The large maximum is simply a numeric artifact of the code essentially having coulomb explosions while creating a uniform dust cloud with random particle initializations. The temperature during injection is around room temperature, 0.025 eV. The heating file is purposely run and we determine where to cut it off as the input for the settling files based on the experimental results, which will be discussed further in Section 4.2.

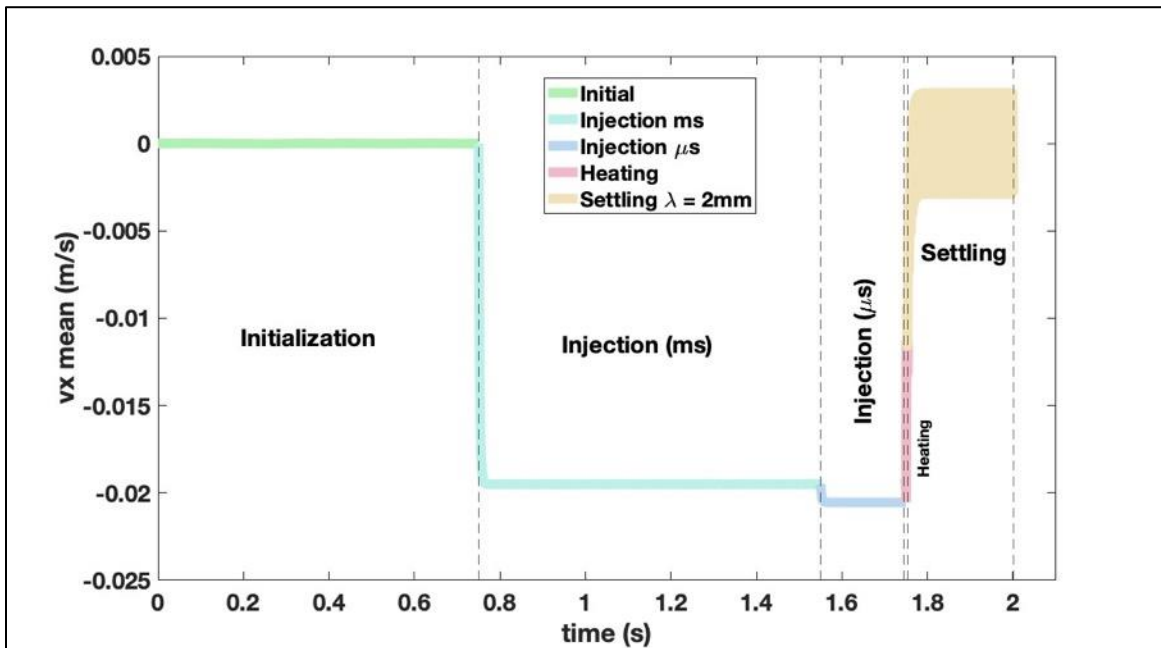


Figure 4-2: The evolution of the average (mean) drift velocity during the preparation steps for the simulation, and one example “settling” segment. The dashed lines show the end of one segment and the beginning of the next segment.

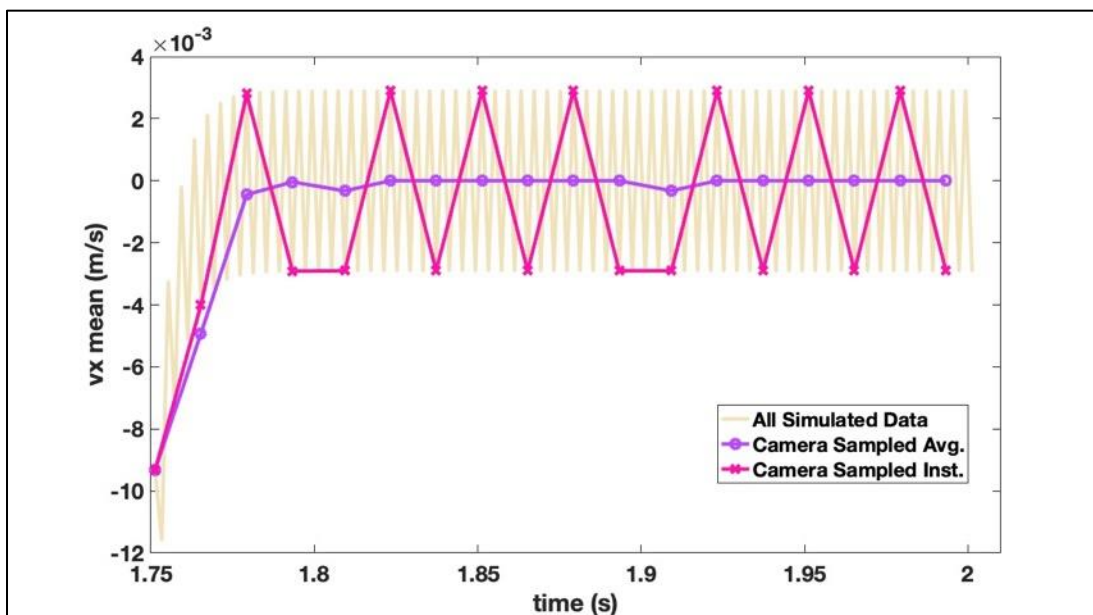


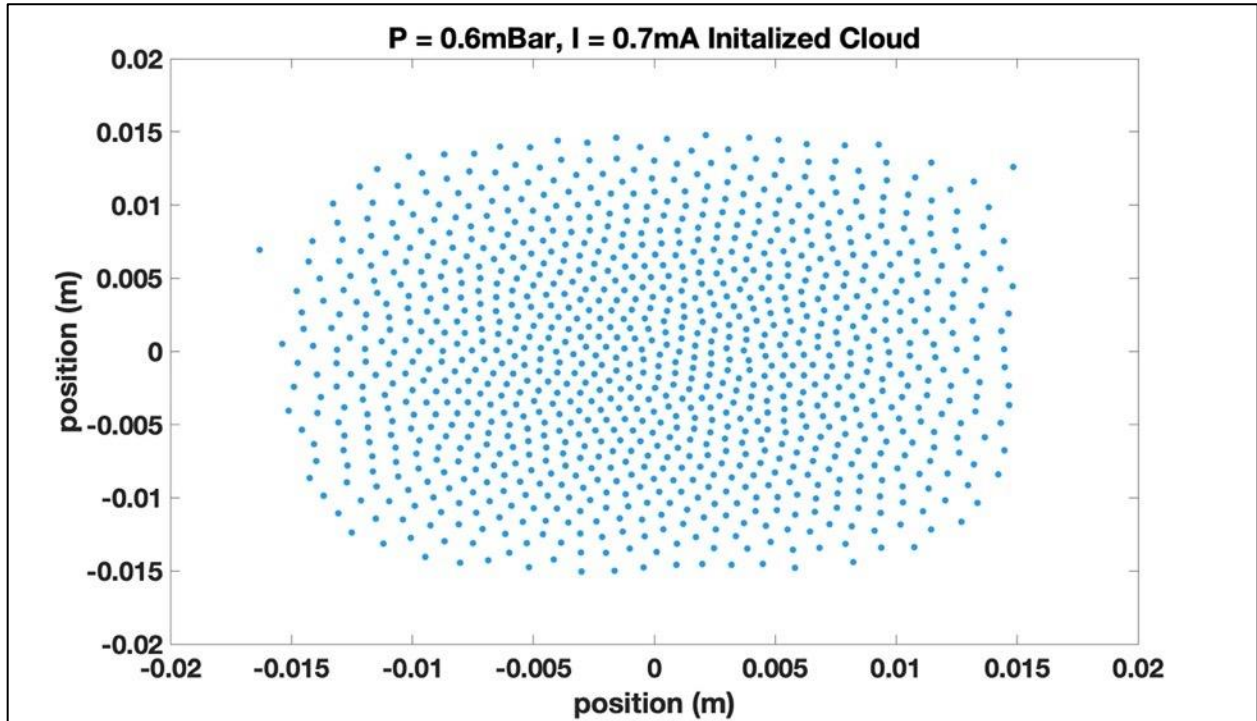
Figure 4-3: The same settling data as shown in Figure 4-2 (yellow), zoomed in to show the oscillation of the drift velocity in the dust cloud. When we camera-sample this data with averages over the 0.014s (purple), the mean drift velocity appears to be near 0 mm/s, with some minor deviations due to manual camera sampling biases. When we camera-sample this data with instantaneous values (pink), the drift velocity still oscillates. We will use the averaging approach for drift velocity, and the instantaneous approach for dust cloud temperatures.

Figure 4-2 shows the mean drift velocity of the dust cloud throughout the entire simulation of the dust in PK-4. When the dust particles are injected into the simulation space and a constant electric field is applied, they drop to experimental drift velocity values ( $\sim 20$  mm/s), due to the neutral drag. There is a small change in magnitude when we change the simulation to  $\mu$ s timesteps, but that is expected when changing the conditions in an MD simulation, as detailed in Section 2.3. When the oscillating electric field is applied, the dust cloud shows oscillation in the drift velocity, which appears as a big block of a line due to the large frequency. When we zoom in to just the “settling” segment, as seen in Figure 4-3, the particle motion is oscillating appropriately at the polarity switching frequency. To perform a comparison between the simulation and experimental

results, the simulation data can be presented as “camera sampled”. As reported in Section 2.1.3, the cameras on PK-4 operate at 70 frames/second (fps) for our Campaign 7 experiment, or over intervals of 1/70th of a second. In order to “camera sample” that simulation data, output quantities from the MD code, such as the position and velocity, will be averaged over  $\Delta t = 0.014$  sec and used to generate a single data point; equivalent to what is recorded by the PO-cameras. The camera sampled (purple) drift velocity results are illustrated in Figure 4-3 and compared to the full output of average drift velocity of the dust cloud. We also can take the instantaneous values (pink) of the output quantities. This will be used for the temperature data. Remember, we can see the oscillation of the dust cloud in the experiments if the polarity switching frequency is low enough ( $< 100$  Hz), as shown in Figure 3-1, so seeing the oscillation in the simulations is reassuring, especially at a much higher temporal resolution than cameras can achieve.

The following subsections will describe each of these initial segments shown in the overview for replicating PK-4 in YOAK $\mu$ M in much greater detail. Each segment has subtle nuances that are essential for the code to operate as close to PK-4 experimental conditions as possible and to create self-consistent results.

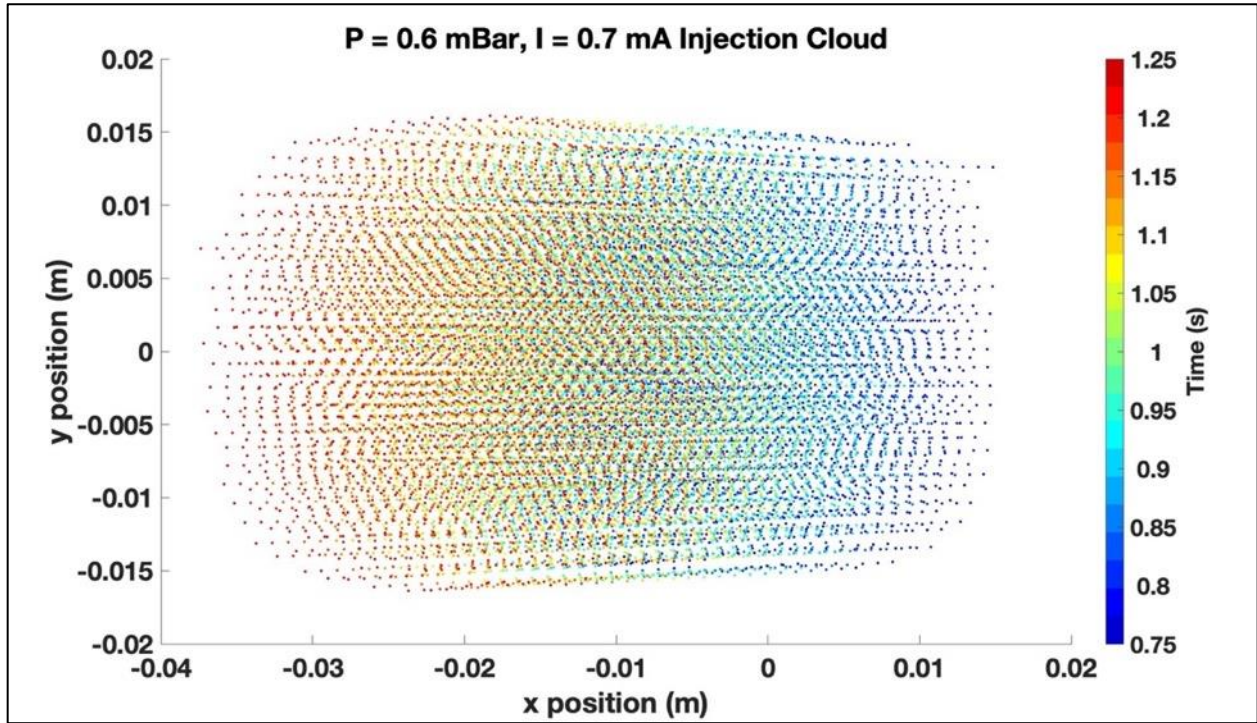
### 4.1.1. Initialize Cloud



**Figure 4-4: Sample dust cloud structure of 1000 particles after initialization and allowing the particles to interact for 0.75s.**

Figure 4-4 shows a sample initialized dust cloud of 1000 particles for the simulation conditions equivalent of experimental conditions  $p = 0.6$  mBar and  $I = 0.7$  mA, the initial set of experimental plasma conditions as shown in Section 3.2. The “initialization” segment of code is run using Yukawa dust-dust interactions and Epstein (neutral) drag, which is how we simulate the plasma environment. Thermal kicks from the Langevin heater are used to equilibrate the dust kinetic temperature to approximately room temperature. The code is set to run for 0.75s using a dust characteristic timestep ( $\Delta t = 1$  ms) to allow the cloud to interact and create a steady-state cloud structure. The last timestep in this initialization file is then read-in for the “time 0” positions and velocities of each dust particle for the next segment of the PK-4 experiment simulation, injection.

### 4.1.2. Injecting the Dust Particles



**Figure 4-5: The simulated dust cloud’s motion during the injection segment, equivalent to the injection of the dust cloud in PK-4. The dust cloud moves from right to left, with color representing time evolution. The cloud slowly grows larger in the y-direction.**

Figure 4-5 illustrates the simulation dust cloud’s motion during the injection segment. The code is run using Yukawa interactions, neutral drag, thermal kicks, and now there is also an electric field equivalent in magnitude to that of the experiment, which accurately reproduces the net drift velocity of the dust cloud we see experimentally. The code is set to run for one second using a dust characteristic timescale ( $\Delta t = 1$  ms) and is “injected” for 1s. Note that as the cloud is injected, there is a small growth in the y-direction because there is no externally imposed confining boundary structure on the dust cloud.

However, to properly resolve the dust particle interactions with the background plasma environment, and among themselves for the subsequent physical processes, we determined that a



finer time resolution was needed (i.e., resolved on a timescale approximately comparable to ion-dust interactions). Therefore, for the final 0.2 s of this injection flow period, the simulation timestep is reduced from 1 ms to 1  $\mu$ s. This is important so that we are utilizing the most accurate dust state values, to self-consistently and accurately reproduce the dust cloud in the PK-4 system, since changing the timestep artificially creates a larger initial acceleration for the dust particles. As seen in Figure 4-1 and Figure 4-2, by changing this timestep (“injection  $\mu$ s”, blue line), we increase the temperature of the dust cloud and magnitude of the velocity slightly, but these values actually align closer to the experimental results ( $T_{x,exp} = 0.11$  eV,  $T_{x,sim} = 0.09$  eV). The last  $\mu$ s timestep data is the initial values for the next segment, heating.

#### **4.2. Physics of Heating and Dissipation in the Dust Cloud**

The next segment of the experiment is to apply polarity switching to the system to capture the dust particles. To recreate polarity switching, we change the electric field from unidirectional (- x -direction) to an oscillating electric field ( $\pm$  x -direction) at a polarity switching frequency of 250 Hz. The electric field is still at the same magnitude of the experimental conditions.

However, the early simulations showed that simply changing the electric field from one direction to oscillating did not lead to particle heating in the simulated dust cloud system as was observed in the experimental results at the equivalent operating condition. Numerous simulation tests were originally performed to make changes to the simulation time steps, plasma conditions, and particles’ charge in order to replicate the experimental observations. It was through these various attempts that led us to development the approaches described below, that led to reproducible, self-consistent dust particle heating that were consistent with the experiment.

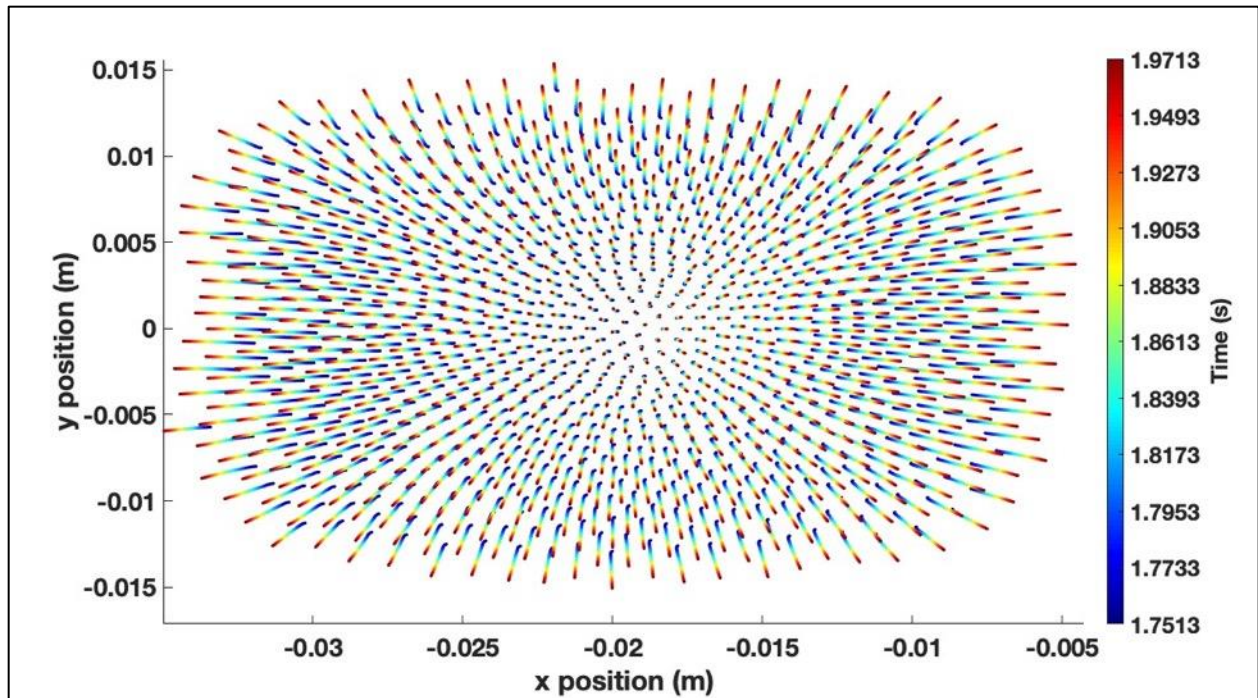
The first key change was first resetting the simulation time steps to  $\Delta t \sim 1$   $\mu$ s for the “heating” and polarity switching portions of the simulation in order to resolve changes in the dust-

particle spatial ordering that provide a reservoir of electrostatic energy that can be converted into thermal energy.

The second key change was motivated by complementary studies performed by the Baylor University dusty plasma group [88]. In those lab-based studies using PK4-BU (a ground-based model of the PK4 setup) and high speed imaging (~5000 frames per second), their work showed that at the application of polarity switching there was a momentary ( $\leq 0.1$  ms) collapse and reforming of the plasma. This was also supported by PIC simulations that were reported in the same paper. This effect is critical because it means that there with a lowering of the plasma density there is a reduction in the dust-dust screening, while the dust particles still remain charged.

We have incorporated this important physical phenomenon into the simulation in three ways: (1) for a short time interval of  $\Delta t = 1.5$  ms, we allow the particles to interact with an unscreened Coulomb potential; allowing the dust grains to self-heat through limited Coulomb explosion (described further below); (2) the effective screening length of the plasma become an important parameter in determining the subsequent dynamics of the particles and is used in the simulations as a “control knob”; and, (3) to ensure that all variations in the dust kinetic temperature are arising self-consistently from dust-dust interactions, the Langevin heater is no longer applied one the code time scale is operating on the  $\Delta t \sim 1$   $\mu$ s time step. We also incorporate these changes ( $\mu$ s timestep and no heater) before the heating segment begins, at the end of the injection segment, so that there are no discontinuities during the transition of segments.

To induce heating into the simulated dust cloud, we change the dust particle interaction from Yukawa to Coulombic. When the dust particles can now interact with all other particles in the cloud instead of just their nearest neighbors due to shielding, this creates a Coulomb explosion [89] and the dust cloud begins to expand. A sample Coulomb explosion for a simulated dust cloud can be seen in Figure 4-6.



**Figure 4-6: A sample dust cloud experiencing a Coulomb explosion. When the dust-dust-interaction changes from Yukawa (shielded) to Coulomb, the dust particles now interact with each other long-range and this net repulsion creates an “explosion” of the dust cloud. Note, this is a different cloud than shown throughout this example section, we run the cloud shown here for a longer “heating” segment ( $t = 0.25$  s) to fully illustrate the explosion.**

We purposefully choose an arbitrarily large total runtime for this heating-induced segment (~40 ms). We then retroactively cut this segment short, based on the experimental results for the same effective plasma conditions. For this example case, we cut the heating segment off when the dust cloud temperature reaches  $T_x = 0.75$  eV, the first “heated” value of the experiment results after the application of polarity switching (shown in Figure 3-4b). This example segment yields a total “heating” segment runtime of 1.3 ms for our sample simulation of the  $p = 0.6$  mBar,  $I = 0.7$  mA dataset in campaign 7.

The final step(s) in this simulation process are to model the dissipation of energy after polarity switching. We do so by using different effective screening lengths in order to find the

value that most accurately represent the experimental results, which will be shown in Section 4.4. This section is run with a  $\Delta t = 1 \mu\text{s}$  timescale, and a total runtime of  $t = 0.25 \text{ s}$ . The only change between the “heating” segment and this section is going back to Yukawa interactions, with a variable Debye length for the code, or the effective dust cloud screening length of the system, to attempt to replication the experimental results. Again, there is no thermal heater in this section, so that there is not a background thermal bath for the dust clouds to keep a net energy artificially higher.

### 4.3. Validation of YOAKUM results

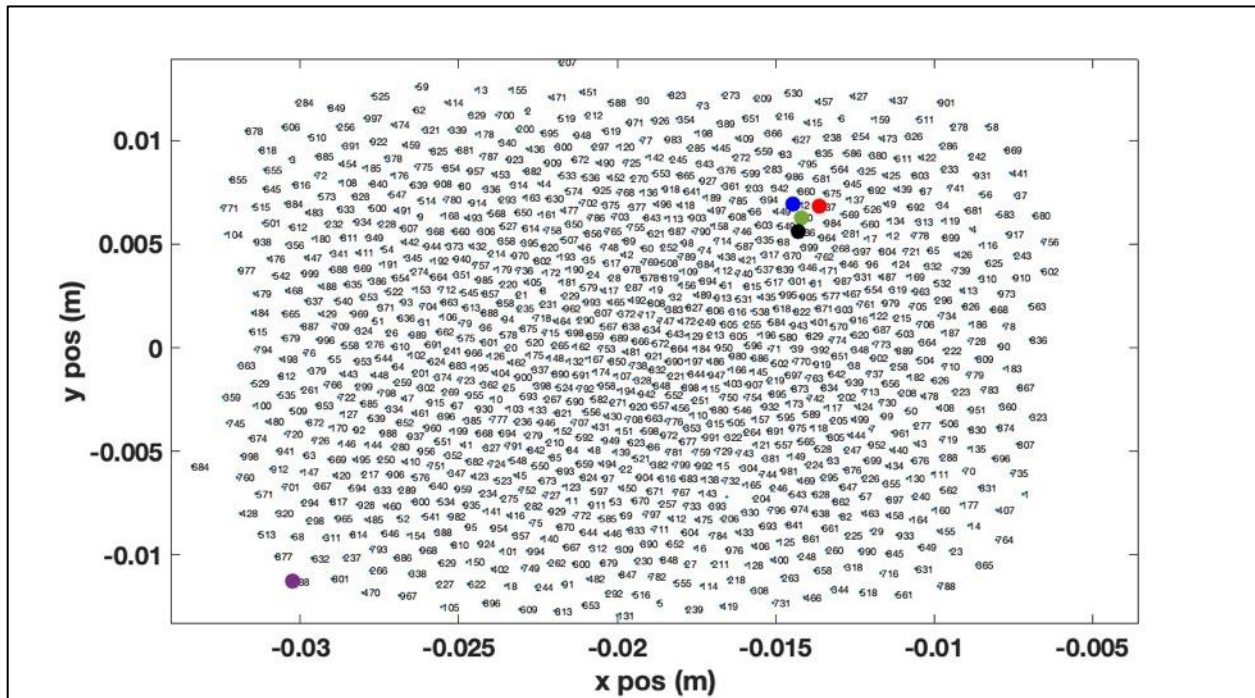
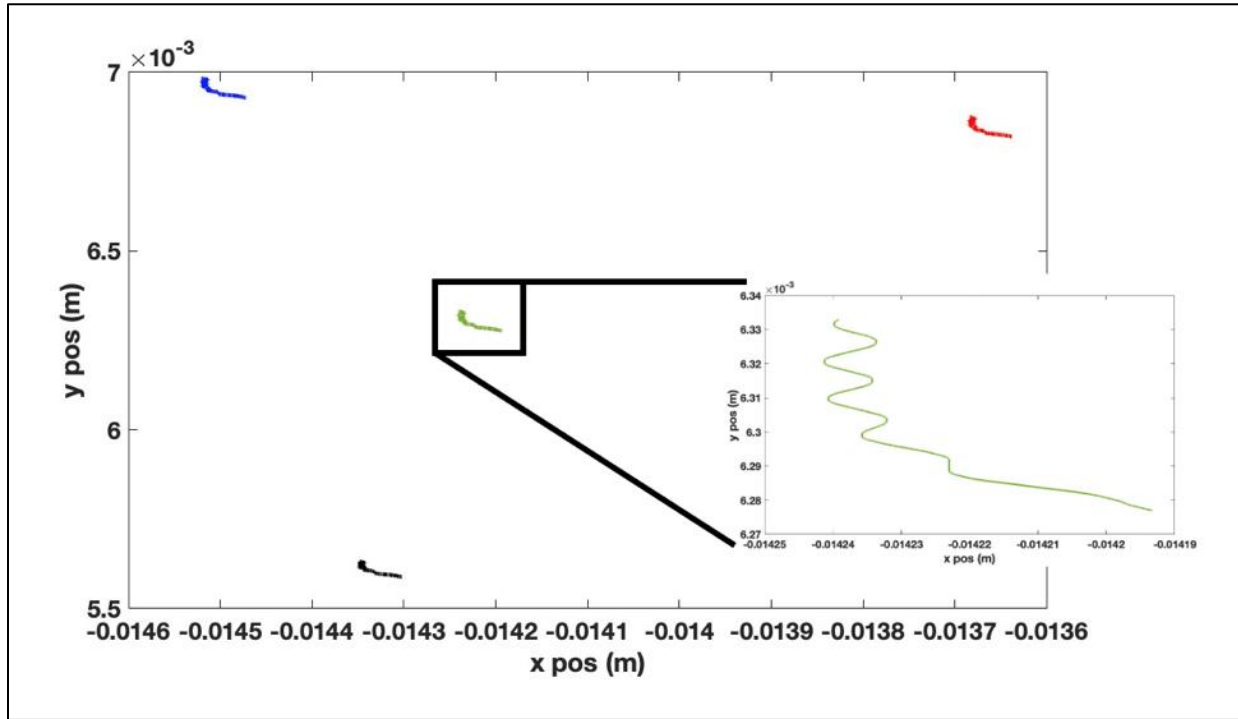


Figure 4-7: A YOAKUM cloud with the particle numbers (indices) labeled. The green dot in the upper right quadrant is particle #40, the focus of our arbitrarily picked in depth analysis along with its nearest neighbors in black, blue, and red, particles #936, #512, and #837, respectively. The purple dot in the bottom left quadrant is the furthest neighbor from our selected particle.

To confirm these simulated temperature measurements are not an artifact of an MD simulation, we will pick several particles to analyze further. To gain all information about the particles, I have rerun the YOAK $\mu$ M simulation with the same input and forces, but now saving every  $\mu$ s timestep's full set of data for a shorter total run time. I arbitrarily picked particle #40 out of 1000 for this closer analysis approach. A layout of the initial cloud is shown in Figure 4-7, where the index number of each particle is labeled. This initial creation of the cloud uses randomized positions, so particles next to each other are not in numeric order for indexing. Particle #40 is the larger green dot, and its nearest neighbors are also shown with larger markers of black, blue and red, all in the upper right quadrant. We will also look at long-range interactions in the dust cloud, so the furthest neighbor is shown with the large purple marker at the bottom left of the cloud. While the positions of the particles within the cloud change with time in the simulation, the particles stay in the same geometric order.

### 4.3.1. Particle Positions and Velocities



**Figure 4-8: A closer look at the 4 reference particles selected. Green is #40, our focus particle. The box shows the trajectory of the particle more closely. Black is particle #936, the nearest neighbor of #40. Blue is particle #512, and Red is particle #837.**

Looking specifically at the selected particle and its 3 nearest neighbors can be seen in Figure 4-8. Particle #40 is in the middle and a more detailed view of the particle's trajectory in time can be seen on the zoom in box to the right. The particle starts at the bottom right of the trajectory, and follows the path to the upper left with time, and the capturing due to the oscillating electric field. The 3 closest neighbors in the dust cloud trajectories are plotted as well, which are represented by the black, blue and red paths, coordinating with their color and initial positions in Figure 4-7. To verify that particles are independent in the simulation, we then wanted to confirm that there are differences in the responses to the forces in the code.

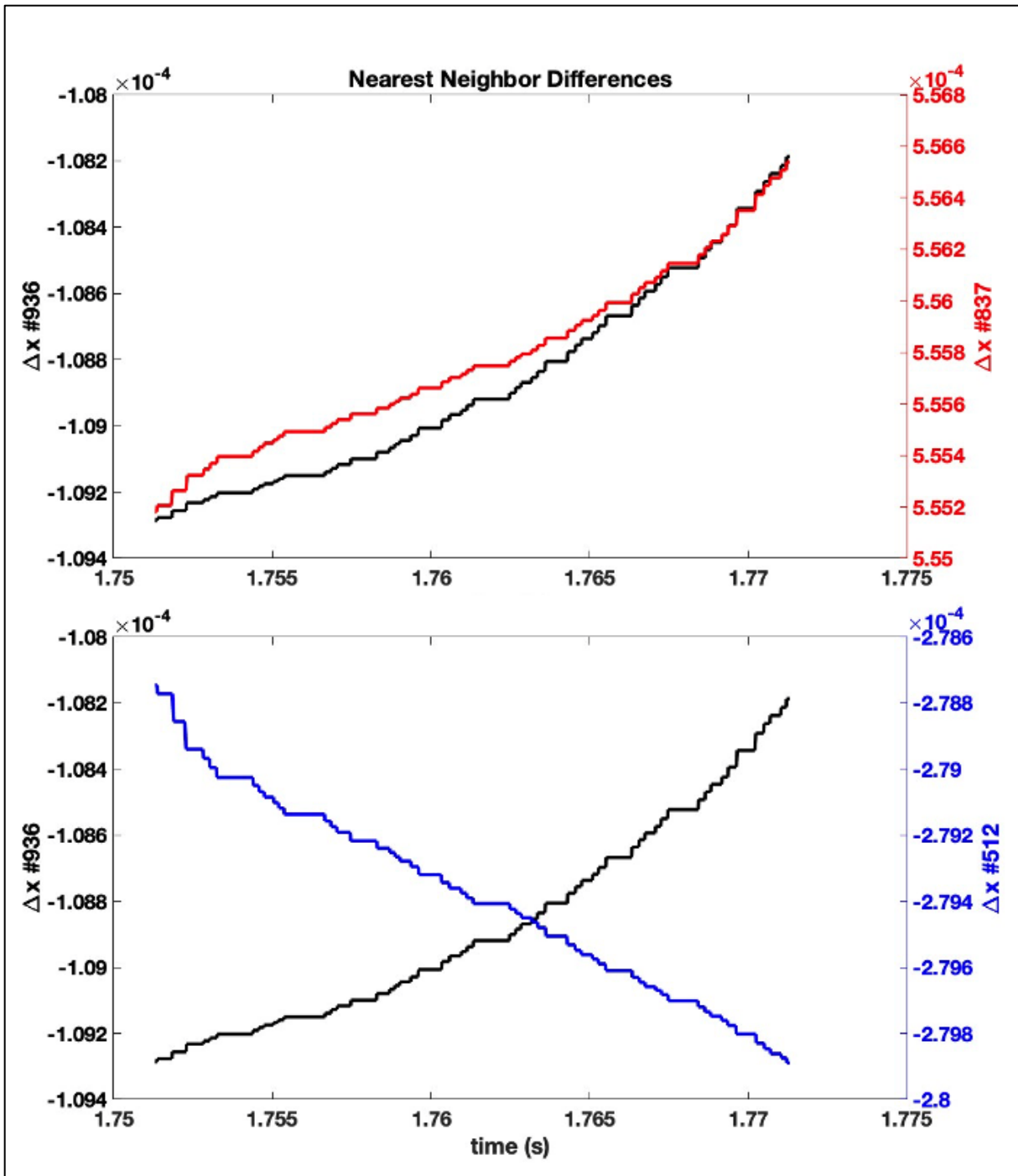
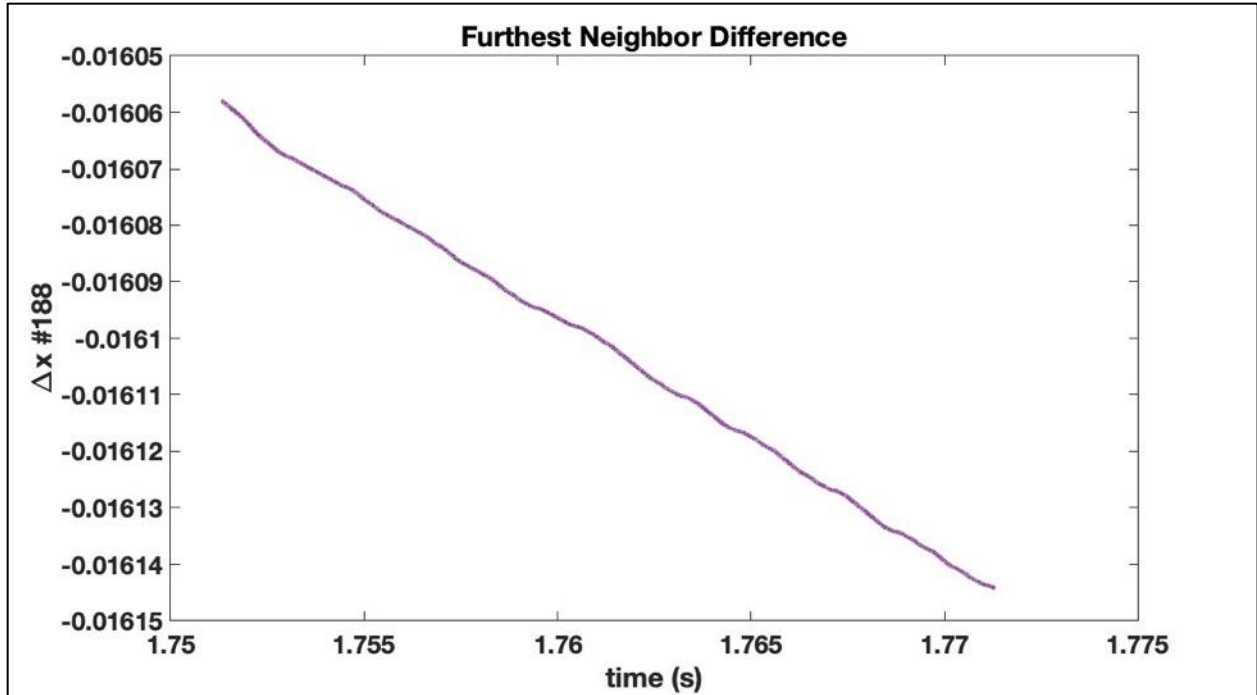


Figure 4-9: Nearest neighbor differences relative to particle #40. The left, black axes in both graphs are the difference for the closest neighbor, #936. The red axis in the difference in path for particle #837, the blue axis is for particle #512, and the colors coordinate with the previous figures. The red and black differences have slightly different trendlines, and the blue and black differences go in opposite directions. These differences in curves help verify the particles respond independently to the forces in the simulation.

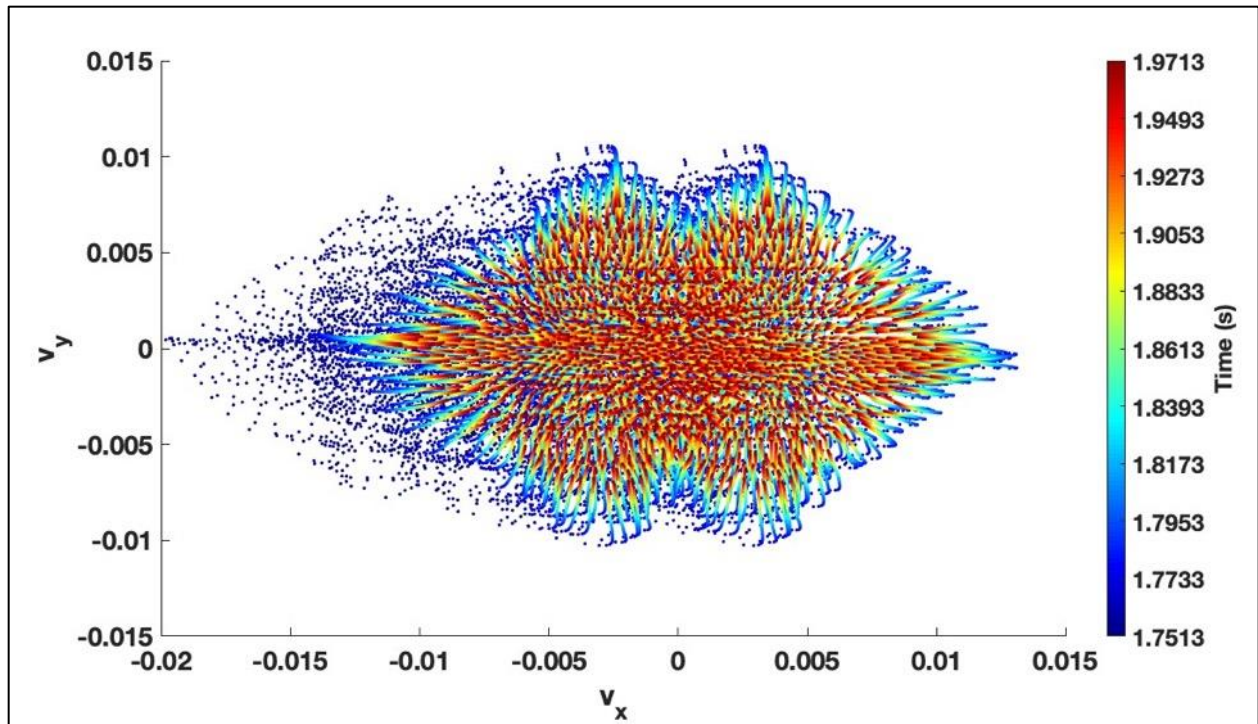
Figure 4-9 shows the difference in x positions for 3 different particles, all using the same reference particle, #40. Note that the black axes in both plots are the same, and the use of two vertical axes (left and right) is simply to be able to highlight the features of the data without the axis's limits blending it together. The stepwise feature that all 3 particles show is important to see that there is difference in how the particles respond to the oscillating electric field and other particles. It is also interesting to note that there are occasionally "longer" steps in these results, and these coincide with when the sinusoidal function of the electric field being at a maximum or minimum. In the upper figure, the differences in position having slightly different trendline slopes, while still trending in the same direction, helps suggest that particles are responding to the forces at different rates. This is further confirmed by the differences in the lower figure having different directions. The variation in these three results suggests that the particles are independent in the simulation which helps support the concept that microscopic responses may be the underlying the basis of the macroscopic heating effects.





**Figure 4-10: Difference between #40 and the furthest particle, #188. This long-range interaction in the dust cloud has a much smoother difference in positions (not stepwise like the nearest neighbors), which indicates that the particles' individual motions are not affected by each other, consistent with Yukawa-shielded interactions.**

To further check that our particles respond to the forces in the simulation independently, we now look at the particle the furthest away from our selected particle, which is particle #188. While there is some variation in the differences, it is not as distinctly stepwise like the nearest neighbors. This shows that the long-range interactions are not as significant on a particle's trajectory as the nearest neighbors, which is to be expected when using Yukawa interactions in the simulation.



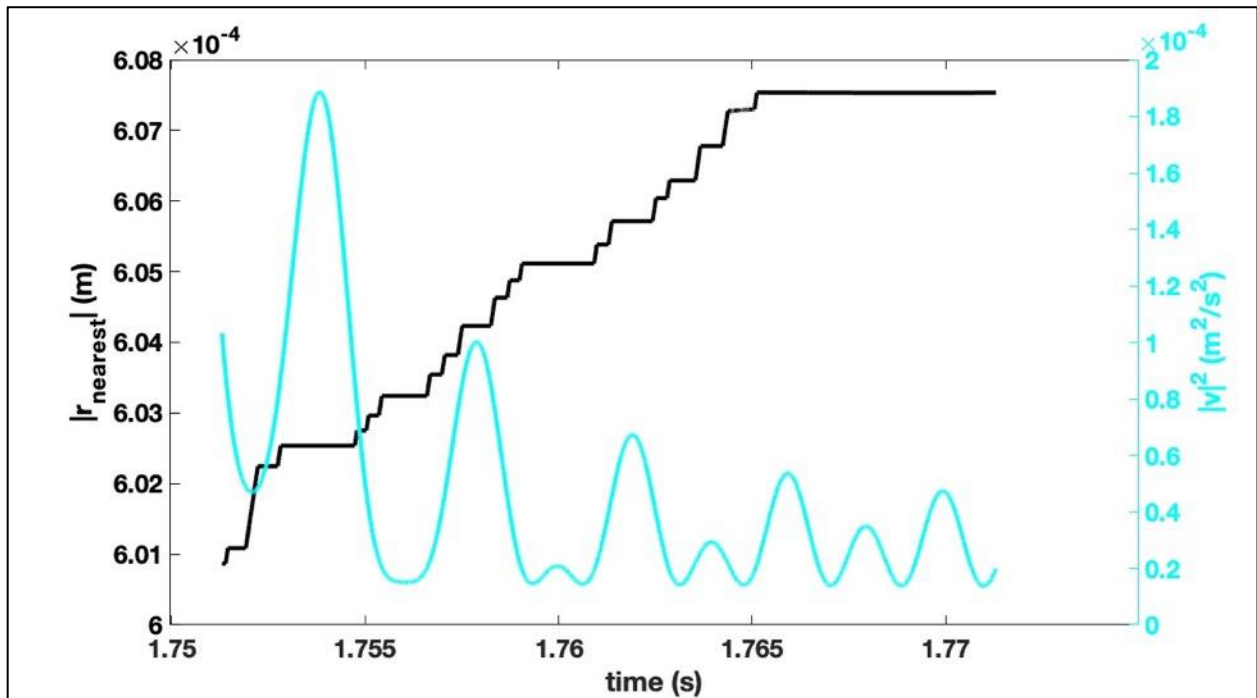
**Figure 4-11: Velocity phase-space evolution during the dissipation segment of the simulation. Note there are two islands, representing the oscillatory velocity after polarity switching. The distribution of velocities all trend towards these local islands as a function of time (color), indicating the dissipation of velocity (kinetic energy).**

We can also look at the dust cloud velocities during this dissipation. Figure 4-11 shows the velocity phase space evolution during the dissipation segment. The velocity of all particles are shown, and color represents time during the simulation. There are two local islands at  $\pm 5$  mm/s in the phase space diagram, which is from the oscillatory nature of the velocity due to polarity switching. All particles' velocities are dropping in magnitude in the y-direction, as seen by the time paths curving inwards, which suggests the growth of the cloud is slowing.

#### **4.3.2. Particle Energies**

Next, we can look at the various energy-dependent values for this sample dust cloud. To get a representative view of the cloud, we will look at the average nearest interparticle distance,  $r$ ,

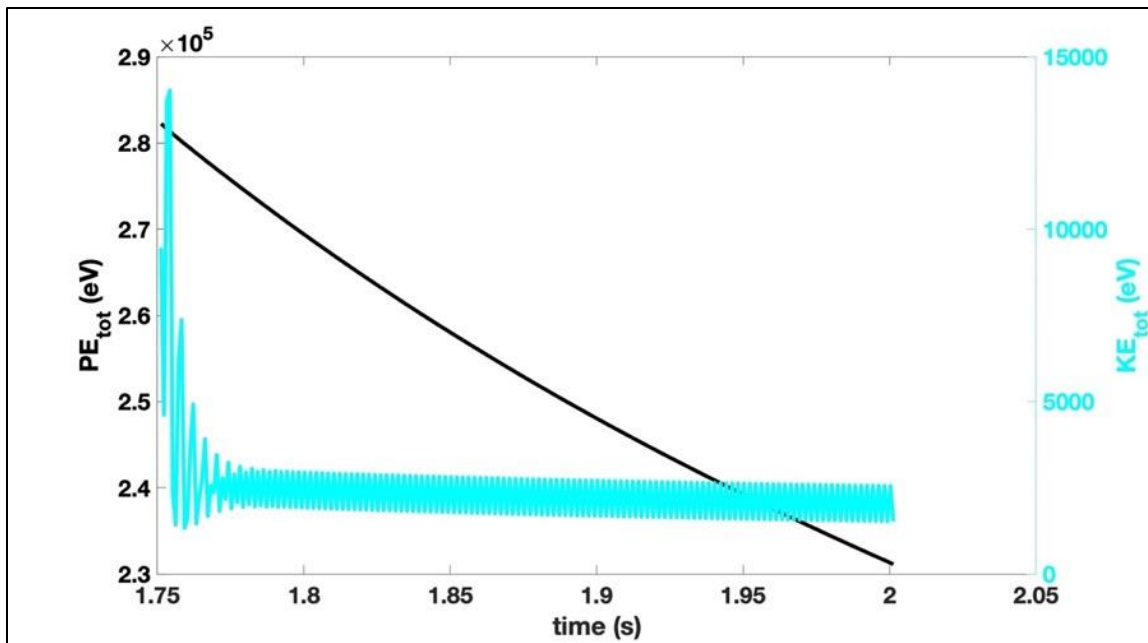
and the values of the average squared velocity of the particles to compare cloud size and speed. Figure 4-12 shows these values on their own vertical axes, for scaling purposes. The interparticle distance is slowly increasing, and the squared velocity is decreasing. This helps show that the increase of the cloud size ( $r$ ), is slowing as the magnitude of velocity squared is decreasing, suggesting the cloud is reaching a steady state, further suggested by the distance reaching a plateau around  $t \sim 1.765$  s.



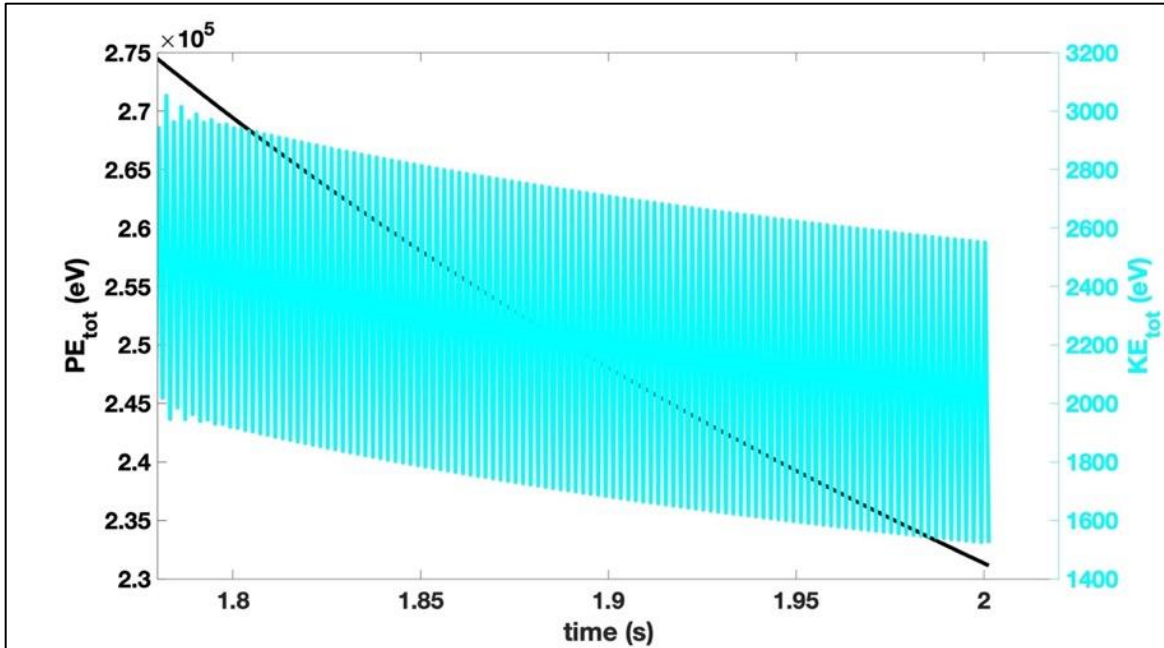
**Figure 4-12: Average dust cloud interparticle distance (left) and average dust particle's velocities (right) vs. time. The particles' distance is slowly increasing (on a small scale) and the velocities are decreasing.**

For some physical meaning of these values, we can estimate how much these small changes in position (between a pair of particles) and velocity (of a single particle) effect the cloud dynamics. Using a dust charge  $Z_d = 10,000e$ , and an effective shielding length of  $\lambda = 1.5 \mu\text{m}$ , we can calculate the potential energy between two particles. A pair of particles with  $600 \mu\text{m}$  for an interparticle distance (smallest value distance in Figure 4-12) would have an interparticle potential

energy (Yukawa-shielded, using Equation 1-13) of  $\sim 161$  eV. When the particles are then moved to an interparticle distance of  $608 \mu\text{m}$  (the largest value distance in Figure 4-12), the potential energy is then decreased to  $\sim 158$  eV. This  $8 \mu\text{m}$  change in spacing releases 3 eV, for a single pair of particles. Next, the kinetic energy of a particle with  $v^2$  magnitude of  $2 \times 10^{-4} \text{ m}^2/\text{s}^2$  (largest value) would be  $\sim 18\text{eV}$ , and the kinetic energy of a particle with magnitude  $0.2 \times 10^{-4} \text{ m}^2/\text{s}^2$  (smallest value) would be  $\sim 1.8\text{eV}$ . A single particle loses an order of magnitude of kinetic energy when transitioning from a flowing to a captured (due to polarity switching) state. The combination of these changes in microscopic particle energies, when scaled to the large dust cloud sizes of PK-4, can significantly impact the macroscopic effect of the dust cloud system.

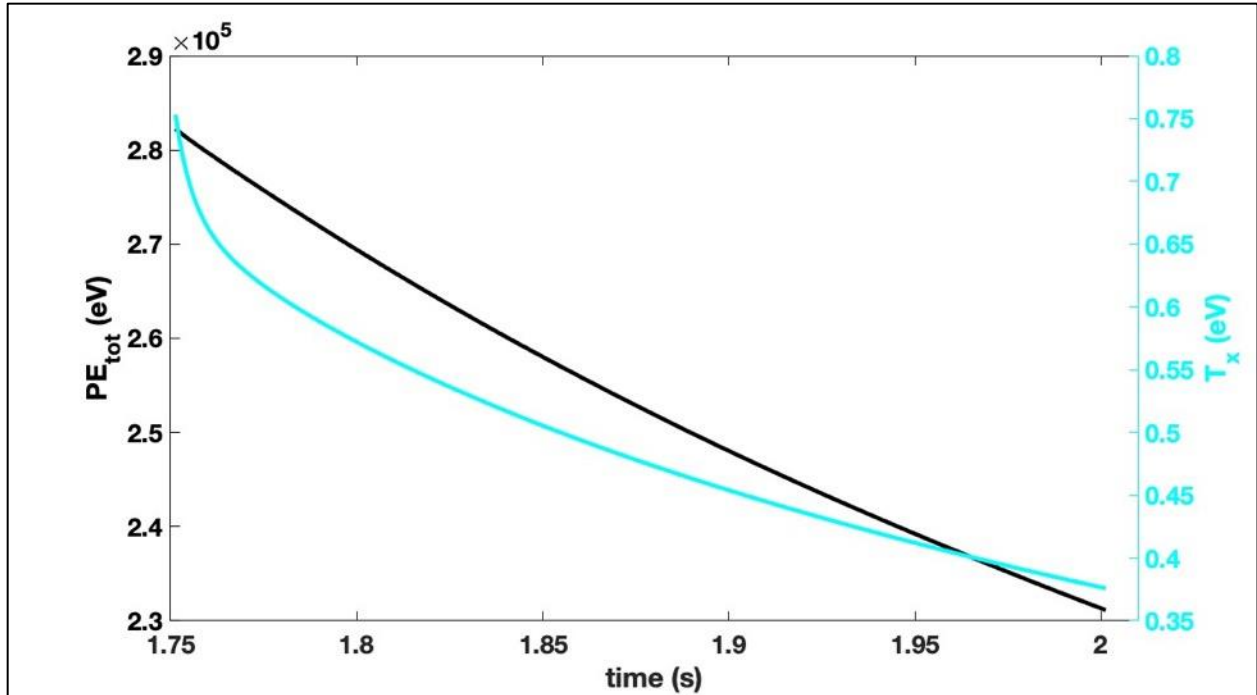


**Figure 4-13: Total potential energy and kinetic energy vs. time for the dust cloud in the YOAK $\mu\text{M}$  settling segment,  $\lambda = 1.75 \mu\text{m}$ . As the potential energy decreases, so does the dust cloud temperature, which can be seen in more detail in Figure 4-14.**



**Figure 4-14: Same energy data as Figure 4-13, but focusing on  $t = 1.775 - 2$  s, with modified kinetic energy vertical axes limits to highlight the decay in both types of energy. Both values decay with time.**

Next, we look at the macroscopic energies of the dust cloud. The graphs of total potential energy and total kinetic energy in the dust cloud are compared in Figure 4-13. Both energy values are decreasing with time, which can be seen further by Figure 4-14, where we have rescaled the kinetic energy axis to show the decrease in magnitude as the potential energy decreases. The potential energy is  $\sim 100$  times larger in magnitude than the total kinetic energy of the system. But, they decay at similar rates on their respective scales. The kinetic energy dissipation is likely into neutral drag with the background plasma environment. The interparticle potential energy decay is suggesting that the dust particles are able to tap into this as a energy reservoir, which is possibly why the temperature stays higher for longer than compared to other forms of dissipation (Epstein drag) in the dust cloud system.



**Figure 4-15: Comparison of a representative potential energy (from the average nearest neighbor distance) and the dust cloud temperature vs. time.**

To investigate the potential energy “bath” further, we can also look at the energy compared to the dust cloud kinetic temperature from the MB fits, shown in Figure 4-15. While these scales are very different, on their own respective axes, they decay at similar rates. There is no direct way to compare between the aggregate (dust kinetic temperature) and individual (potential energy) values in this system. However, the potential energy in the dust-dust interactions appears to provide a “reservoir” of energy that can be tapped to counteract the dissipation due to neutral drag. This is our proposed source of energy that contributes to the slower than expected (Epstein drag) cooling of the dust cloud.

Ultimately, since the trend in energies presented are consistent with our predictions for the experimental interactions at the application of polarity switching, this further leaves us confident

that our simulations are representative and a well-rounded approach to determining the heating and dissipation of the dust cloud temperature at the application of polarity switching.

### 4.3.3. Cloud Size Validation

To optimize computational efficiency, we want to use the least number of particles possible without creating a large numeric error when compared to the experimental results. The dust clouds seen in the PK-4 experiment have a numeric density of  $n_d \sim 10^5 \text{ cm}^{-3}$ , and based on cloud dimensions, that means there are  $\sim 8 \times 10^5$  particles in the system (based on 5 shakes of the  $3.34\mu\text{m}$  diameter dust particles, used in Campaign 7). However, for computational efficiency, we could never accurately simulate that many particles in a system. We can keep a similar interparticle distance, and use a smaller total cloud size, to yield similar interactions of the PK-4 system. Figure 4-16 shows the dust temperature evolution for three different sized simulation clouds- 1000 particles, 2000 particles, and 5000 particles. The data for 1000 particles is the same data used to illustrate the segments in Figure 4-1. The “initial” segment requires different total run times to reach an equilibrium in the dust cloud, and is shown by the short, dashed lines. These all occur before  $t = 3$  s. The injection segment run with ms timesteps is shown as longer dashed lines, from  $t = 3-4$  s, and the  $\mu\text{s}$  timestep injections are shown as solid lines from  $t = 4 - 4.25$  s. The injection  $\mu\text{s}$  timesteps segments are used to compare the dust cloud sizes directly. There is currently a 1.1% difference between the 1000 particle cloud and 2000 particle cloud temperatures. And the difference between the 1000 particle cloud and the 5000 particle cloud temperatures is 1.8%.

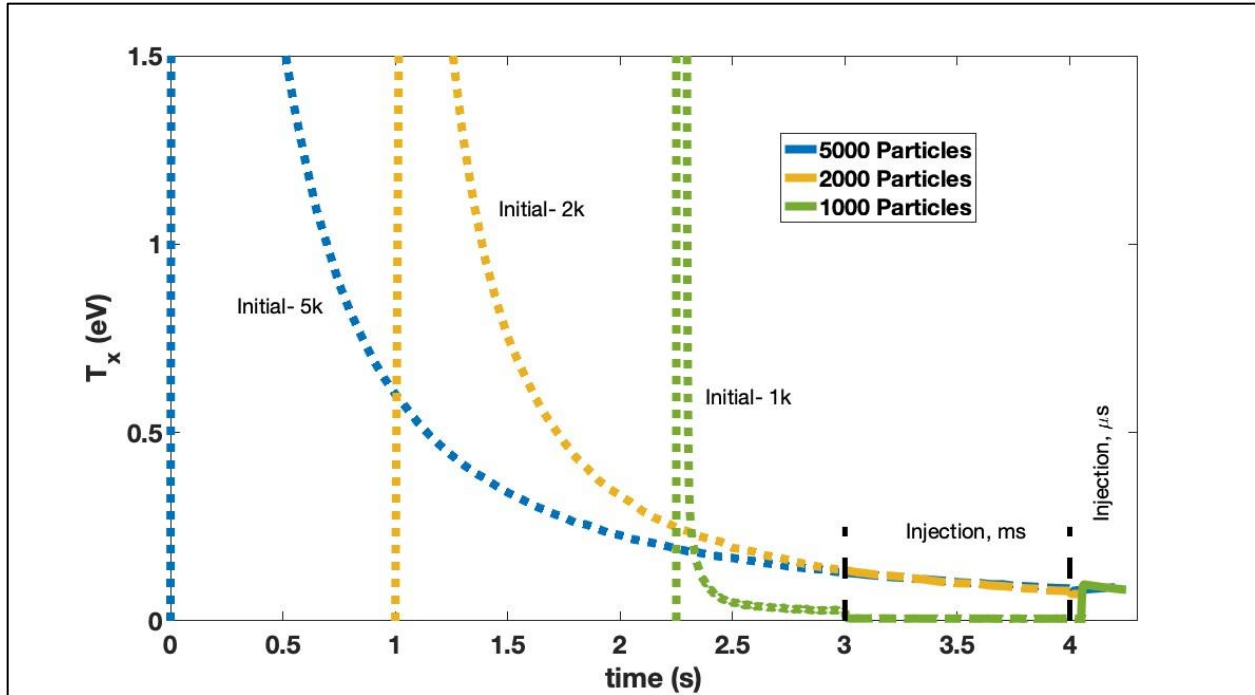
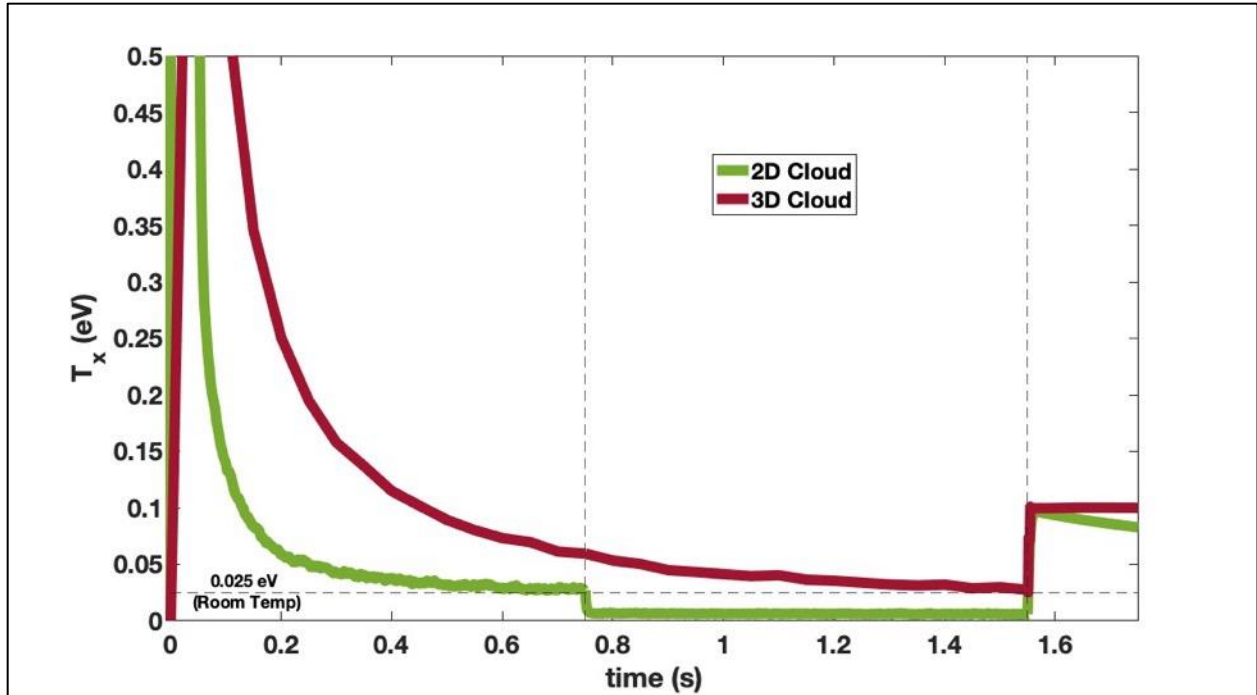


Figure 4-16: Dust cloud temperature evolution, for 1000, 2000, and 5000 particles during the three initialization segments. The quick dash line indicates initialization, the longer dashed lines indicate injection ms, and the solid lines are injection  $\mu$ s segments. The black dashed lines are to help show transitions between segments for each of the dust clouds. The difference in 1000 and 2000 particles at the end of the injection  $\mu$ s segment is 1.1% and the difference in 1000 and 2000 particles is 1.8%.

The difference in total calculations (and runtime) scales as  $N^2$  since the simulation calculates forces for every particle, as well as the (Yukawa) interactions between a particle and all other particles in the system. Even though these values are effectively screened for long-range interactions, the simulation doesn't skip calculating their contributions. Between 1000 and 2000, there are 4 times more calculations, and between 1000 and 5000 there are 25 times more calculations. Since all three clouds yield similar results, we choose to use 1000 particles for optimal simulation time.





**Figure 4-17: Comparison between 1000 particle 2D clouds and 1000 particle 3D clouds during the initialization process. The green, 2D cloud is the same temperature evolution shown in Figure 4-1, and the vertical axes is a smaller range to illustrate the subtle difference between the two clouds in the simulation.**

We can also validate our 1000 particle cloud 2D cloud against a 1000 particle 3D cloud. Figure 4-17 compares the dust cloud kinetic temperature evolution during the initialization processes, through the injection using  $\mu$ s timesteps segment. The 1000 particle temperature line is the same as in Figure 4-1, now presented as all one color for easier comparison, and the vertical axis limits have been reduced to better highlight the subtle differences between simulated dust clouds. While there are differences in the initialization of the 2D and 3D dust clouds, the injection segment using  $\mu$ s timesteps has a similar result for the dust cloud, yielding a 4.4% difference between the two curves at the end of the segment,  $t = 1.75$  s.

#### 4.4. MD Simulation Results

Based upon the wide range of simulation tests presented above and supporting experimental results from the Baylor group [88], we have developed the following hypothesis for the observed dust heating. As the plasma momentarily changes/collapses, the dust particles lose a small amount of charge that allows for the dust structure's interparticle spacing to decrease. As the plasma returns to the new, oscillating conditions, the dust cloud then has a coulomb explosion response as they return to their calculated charges, which causes the heating event. This heating then takes an extended time to dissipate back into the potential energy within the dust cloud structure.

We have incorporated this phenomenon into the YOAK $\mu$ M simulation in the following manner: at the onset of polarity switching, a 1.3 ms suppression of the plasma is modeled by allowing the screening length of the particles to become disturbed, i.e.,  $\lambda_{\text{screening}}$  is allowed to become large and the particles are allowed to have a more Coulomb-like particle interaction during this period. This 1.3 ms is empirically chosen to best reproduce the heating event numerically. This leads to a Coulomb expansion of the cloud as electrostatic potential energy between the charged dust particles is converted into kinetic energy. At the end of this 1.3 ms period, the plasma is restored with a new effective screening length. If the screening length is too large (i.e., Coulomb-like), the cloud continues to Coulomb expand, and if it is too small, it rapidly decays into an ordered dust cloud. This is illustrated in the results summarized in Figure 4-18.

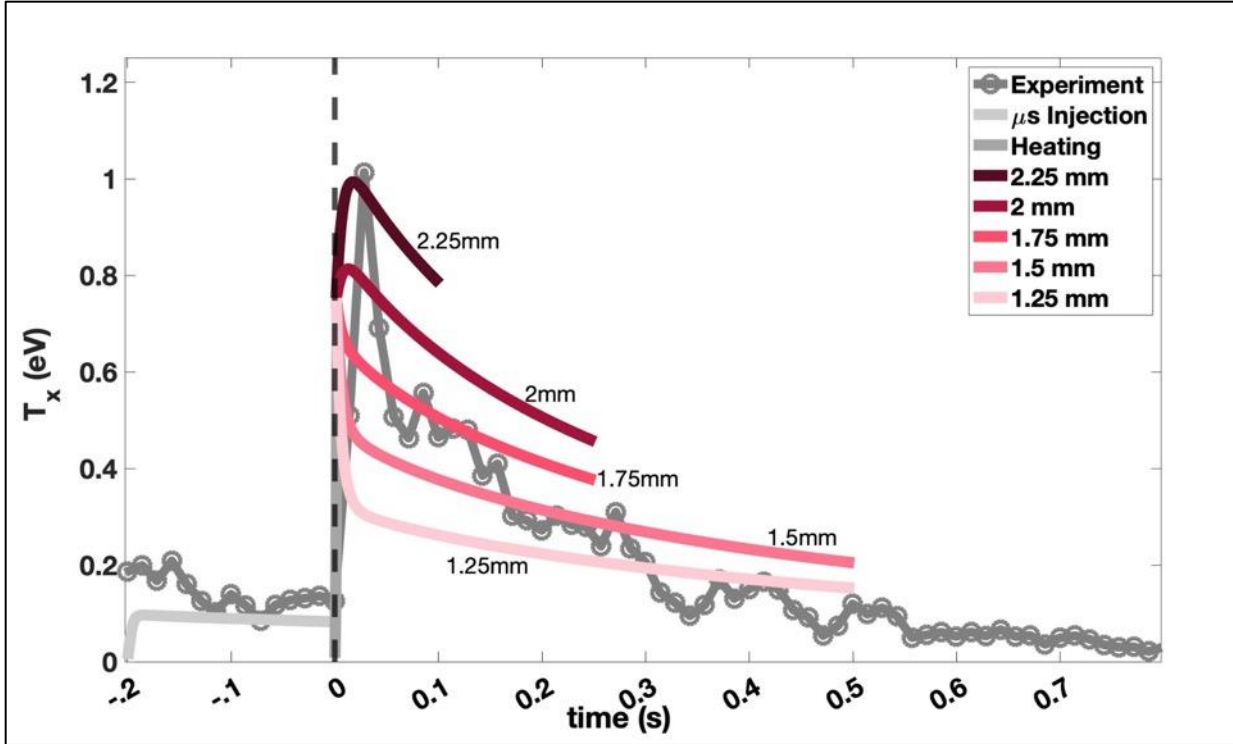
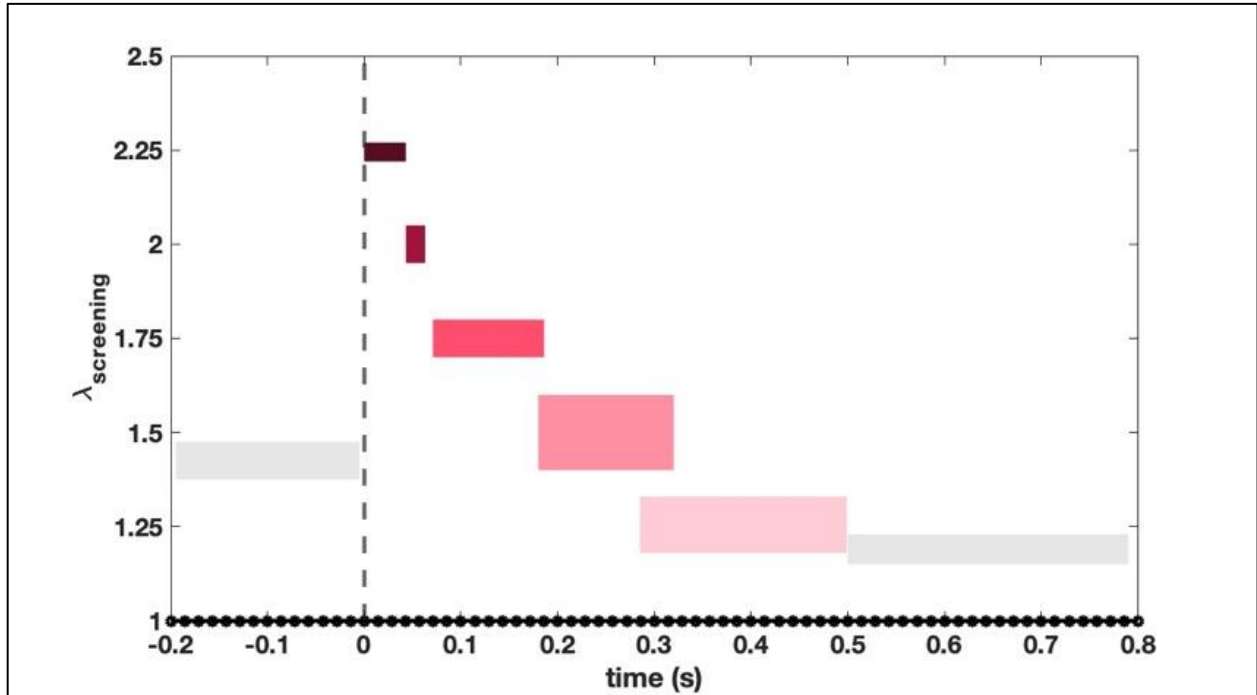


Figure 4-18: The change in dust cloud temperature after the application of polarity switching (normalized to  $t = 0$  s) for varying screening lengths. The code is paused at  $t = 1.3$  ms to change the dust-dust interaction's screening length from Coulomb-like to a screening length indicated on the legend. The dots are the microgravity experimental x-direction temperature data points from Figure 3-4. The 1.75 mm screening length is the best-fit data for the first half of the dataset but does not follow the decay of the experimental data exactly, indicating smaller screening lengths are a better fit as the experimental temperature decays.

Figure 4-18 shows the results of several simulations. The simulation results are shown in the varying pink-colored curves and the experimental measurements from PK-4 are shown as the dark gray, circular data points. All of the simulations start by allowing Coulomb-like particle interaction for 1.3 ms, the light gray curve, to model the heating event for the dust cloud temperature. The results of five subsequent simulation runs are shown for effective screening

lengths,  $\lambda_{\text{screening}} = (1.25, 1.5, 1.75, 2, 2.25)$  mm. For reference, the electron Debye length for these conditions based on the parameters in *Pustylnik, et al.* [62] is  $\lambda_{\text{De}} = 1.45$  mm.



**Figure 4-19:** The estimated characterization of dust screening length as a function of time at the application of polarity switching. This is based on when the simulation curves pass through the experimental curve in Figure 4-18. To help reference between the two figures, the x-ticks in this figure are set at the framerate of the experimental data.

The simulation results suggest that immediately after heating event, an effective screening length  $\lambda_{\text{screening}} \sim 2.25$  mm, which is larger than the pre-polarity switching electron Debye length, provides an effective fit to the experimental observations. However, this agreement only persists from  $0.0 < t < 0.05$  seconds. There is  $\sim 1$  frame for which  $\lambda_{\text{screening}} \sim 2.0$  mm agrees with the experimental data. For  $0.07 < t < 0.2$  sec, an effective screening length of  $\lambda_{\text{screening}} \sim 1.75$  mm appears to be in reasonable agreement. For  $0.2 < t < 0.32$  sec, an effective screening length of  $\lambda_{\text{screening}} \sim 1.5$  mm appears to agree. Then, for  $t > 0.32$  sec,  $\lambda_{\text{screening}} \sim 1.25$  mm provides reasonable agreement with the experimental data. This change in dust screening length versus time is

visualized in Figure 4-19, with the colors corresponding to the simulation result lines in Figure 4-18. To further help coordinate between the two figures, the x-axis ticks in the screening length time-variation figure are at the framerate of the experiment to help match when the simulations and the experimental temperature values intersect to the exact datapoint.

In the equation of motion, Equation 1-26, this extended decay passing through the various screening lengths would suggest that the dominating term influencing this extended decay timescale is the interaction force, the term that is expanded in Equation 1-14. This arises from the change in plasma conditions, either density or temperature (or a combination of both), for the plasma species, with the single species Debye lengths defined in Equation 1-18, and subsequently effects the dust particles' effective screening length, which is dependent on both the ion and electron Debye lengths by Equation 1-20.

#### **4.5. Simulation Discussion**

The results shown here suggest that the computational model determined via YOAK $\mu$ M are providing a reasonable fit to the experimental observations. We have validated the use of this code using segments to reproduce each part of the PK-4 dust cloud, and the initial injection results align with the experimental results. The dust particles in the simulation respond independently to the forces of the simulation, which ultimately means the structural energy of the dust cloud can serve as an energy source for keeping the dissipation of energy at a slower rate than we would expect experimentally.

These results also suggest there is temporarily an effective larger screening length at the application of polarity switching than calculated based on the experimental conditions. On the assumption that the screening length is representative of an effective Debye length and that its variation is dominated by a change in the plasma density, then decreasing screening length

observed in the simulations would suggest an increasing plasma density for a short period of time after the application of the polarity switching; this would be consistent with the modification in the plasma shown by Baylor [90].

## **Chapter 5: Conclusions and Future Work**

At the beginning of this dissertation, our goal was to investigate the transition of kinetic energy into thermal energy in a dusty plasma at the application of an oscillating electric field in the PK-4 microgravity experiment. This work was established by previous ground-based experiments and validations, and this was our group's first ISS experiment. We successfully ran several experiments on the International Space Station and in this concluding chapter, a discussion on the how this goal was met will be presented.

In Chapter 1, an introduction to the relevant dusty plasma principles and forces within the scope of this dissertation was presented. This included deriving and outlining the method of dust charging, OML, relevant forces in a dusty plasma that would be used for our MD code, as well as a detailed presentation of the derivation of Debye length in a plasma. We knew the dust-dust interaction would be important since in a microgravity plasma, the particles can fill the entire plasma volume, whereas on ground-based experiments the cloud is compressed into sheets within the plasma sheath, and the role of dust-dust interactions (even though the particles may be closer together), can be effectively suppressed due to the roles of gravity and the ion drag forces. The charge of a dust particle also plays an important role when interacting with the other charged species (electrons and ions) in a plasma. The Debye length is a fundamental characteristic of a plasma, but due to our experimental limitations, we have to view this as an effective screening length. These all combine to create an entangled set of interactions between the plasma system and the dust particles.

Chapter 2 presented the experimental setup and hardware, PK-4 (schematic in Figure 2-1), which is a unique experiment setup designed to look at differences in “fluid like” (when flowing) and “solid like” (captured) dust clouds. We seek to investigate the transition between these phases. We also introduced the primary analysis technique, PIV, using the 2D-2V (2 dimensions, 2 velocity components) high framerate video cross-correlation method, and discussed the various settings we can use in this software to optimize our results without creating artificial vectors. We also presented past work that validates the use of PIV on the PK-4 experiment, for varying PIV settings and PK-4 framerates. And finally, we presented an introduction to molecular dynamics (MD) simulation codes. MD codes use a balance of forces to calculate the state of every particle in a system. This can be computationally costly, and there is a fine balance between computationally efficient timesteps and making sure you are able to not overlook any physical effects.

Our experimental results were presented in Chapter 3. This was our first experiment on the International Space Station, and we collaborated with the researchers of CASPER at Baylor University to maximize an experiment slot. Based on our initial ground experiments, we wanted to investigate the dissipation of kinetic energy at the application of polarity switching in both directions. We found that there is dissipation of energy into the perpendicular ( $z$ ) direction, but the main focus of our work became the long-extended dissipation that arises in the parallel ( $x$ ) direction, shown in Figure 3-4.

The results of the experiments are limited due to time allotments and the inability to gain other data (probes, running the experiment without dust to get plasma information, etc.). We are confident that the results we found from our Campaign 7 experiment are physically relevant and not due to an analysis error, as presented with our histogram validations. This extended dissipation



in microgravity is further confirmed by it appearing in 6 out of 9 datasets in at least one camera for the injection segments, shown in Figure 3-16.

We also see evidence of this heating and dissipation in the data taken during our campaign 12 experiment, during the reinjection minus/plus transition between captures (Figure 3-19). This is when we set the electric field back to unidirectional to send the particles out of the FOV to the right and then set the electric field back to bring the cloud back in like an injection. We also see heating during the polarity switching stepdown segments (Figure 3-21), which was originally to investigate the change in the chain-like structures when the polarity switching is changes. The property all of these instances of heating and extended dissipation have in common is a change in the electric field, whether it be direction at a slow rate (single change) or a frequent rate (oscillation).

Since the electric field directly interacts with the charged particle species of the plasma environment, we examined the calculated electron Debye length for all experimental conditions. We found that there is a range of values (1.36 – 2.19 mm) for which this extended dissipation can occur experimentally, summarized in Table 3-4. The electron Debye length also represents an effective dust particle screening length, and therefore the plasma-dust interactions must also be modified when there is a change in the electric field.

And finally, simulation results were shown in Chapter 4. We discussed how to reproduce PK-4 in the simulations, including which forces are turned on in the simulation for which segments, and that the initial simulation segments produce dust clouds that replicate the drift velocity and temperature of the experimental dust cloud well. However, when it came to reproducing the heating and dissipation in the simulation at the application of polarity switching, we could not reproduce this heating with just changing the electric field in the code. By adding a segment which induced heating by allowing a short Coulomb explosion, and then returning to

Yukawa interactions, the modified screening length results reproduce the experimental results well (seen in Figure 4-18). By examining the experimental results align with the various screening length simulated results, we can show the time-dependence of the screening length, in Figure 4-19.

## **5.1. Conclusions**

The microgravity-based PK-4 experiments replicated the heating and extended dissipation seen in our initial ground-based experiments (although our campaign 7 ground-based campaign experiments did not exhibit these characteristics). These characteristics appear in many plasma operating conditions, and at a variety of changes in the electric field, not just the application of polarity switching. Upon further analysis, it appears that there is a correlation between when this heating occurs and the dust characteristic screening length, akin to the dust Debye length.

Based on our initial experiments, we are able to reproduce the PK-4 experiment in our MD simulation. If we change the interaction type from Yukawa-like to coulombic for a short period of time, and then return to Yukawa interactions, we can induce heating and the extended dissipation is similar to the experiment. However, the experimental decay passes through several different simulated screening lengths, indicating that there is a change in the screening length during this dissipation event. A change in the screening length would be due to changes in the electron or ion temperatures and densities of the plasma environment, which has been shown in previous works [88,91], so this may be a plausible explanation for the experimental observations.

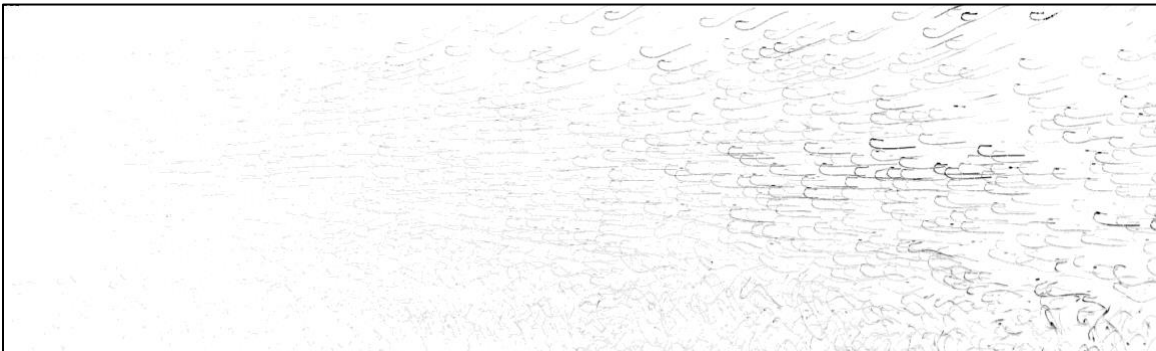
The subtle variations in the simulation suggest that the dust cloud is able to tap into the structural energy as an energy source to keep the dust cloud's temperature higher for longer than expected, which in turn looks like an extended dissipation timescale. As the effective screening length is changing, the dust particles are interacting with each other differently, and this means the potential energy is effectively changing with the plasma screening length as well.

Thermal heating is physical, not artificial, and can only be resolved by simulations due to limitations of the experiment. Further investigation of the simulations show that it must be run on  $\mu\text{s}$  timescales in order for the extended heating to occur. This suggests that at the  $\mu\text{s}$  timescale, the dust particles are able to tap into the cloud's system potential energy to keep energy and cause this extended decay.

## 5.2. Future Work

There have been many other interesting results from the PK-4 experiments I have run throughout my graduate work. While I have not been able to delve deeper into these results, I will list them here for future studies. Each was determined in direct support of this work, but ultimately was not investigated fully, due to various limitations at the time.

### 5.2.1. “Hook” Paths



**Figure 5-1: Hook-like features that occur at the application of polarity switching in the ground experiment. This image is 4 consecutive frames overlapped from the  $p = 0.6 \text{ mBar}$ ,  $I = 0.7 \text{ mA}$  experimental conditions. Note: the color is inverted (black/white) for visibility.**

During the analysis of the first campaign 7 data, we discovered “hook” paths that occur in multiple experimental conditions in ground-based experiments. These are a special case, in part due to what is happening right before these images, the polarity switching stepdown segments mentioned in Section 3.4.2. At the lower frequencies, the dust particles are not captured strongly

and therefore can oscillate along with the electric field horizontally, which was shown in the benchmarking data from Figure 3-1. This lowest polarity switching frequency is occurring right before the frames shown in Figure 5-1, at a frequency of 25 Hz. Figure 5-1 is a sequence of 5 consecutive video frames that are overlaid to show the motion of the particles. This was done to see the effects of the particles at the application of polarity switching, and then we found more than we expected in this unique particle trace sequence. There are hook-like paths that form throughout the dust cloud, and this occurs in multiple ground-based experimental conditions. We do not see these traces at the end of the microgravity polarity switching stepdown segments. As you can see at the bottom of the frame, the particles have some waves and the hooks also follow the wave motion. In the middle of the cloud, the hooks tend to be more horizontal oriented, and towards the top of the frame the particles swing upwards in the cloud at a small angle.

In these instances, we can see the individual exposures of each frame when overlapped. During the first frame, or the top of the hook, the particles are moving towards the right. In the second frame the particles turn around and come back towards the left and curve downwards a few pixels. Then in frame three they continue following the curved path, and for frame four and five they are ejected on the bottom of the trajectory and continue in their path of reinjection minus, sending them to the right. Some preliminary modeling of this work has been done by an Auburn undergrad student, Neve Smith and Dr. Thomas, and Dr. Williams and students at Wittenberg University.

### **5.2.2. Campaign 12 Full Analysis**

The next opportunity for further work is a full analysis on all of the campaign 12 data. This experiment was designed to have as many injections as possible using a variety of dust particle diameters and dusty densities/cloud sizes. This work was conducted in June 2021, and the full

dataset has not been received yet, but we anticipate delivery of this data from the European Space Agency control center in France (CNES), shortly. The work discussed in chapter 3 was the only portion of our experiment that was downlinked to us at the end of the day.

This experiment ultimately had 6 injections, and used a higher framerate of 140 fps, which will give us higher temporal resolution to the heating event evolutions at the application of polarity switching. This should help resolve how quickly the dissipation occurs, since the current experimental results are jumpy from frame-to-frame datapoints. We used 2 dust particle sizes and two different total dust cloud sizes (number of shakes) to also be able to investigate the dependence of this heating and dissipation on the dust particles size and total dust cloud numeric density. This entire experiment was run using the plasma conditions  $p = 0.6$  mBar, and  $I = 0.7$  mA for direct comparison to our Campaign 7 primary dataset. The list of injections can be found below in Table 5-1, since the exact values were determined on the day of the experiment and not explicitly stated in the flowchart of the experiment in A.2.2.

Dust Diameter ( $\mu\text{m}$ )	Number of Shakes
3.34	5
3.34	7
6.86	2
6.86	4
6.86	4
3.34	5

**Table 5-1: Injection dust cloud parameters for Campaign 12. The fourth injection was not captured well, and we repeated the conditions a second time for a better result.**

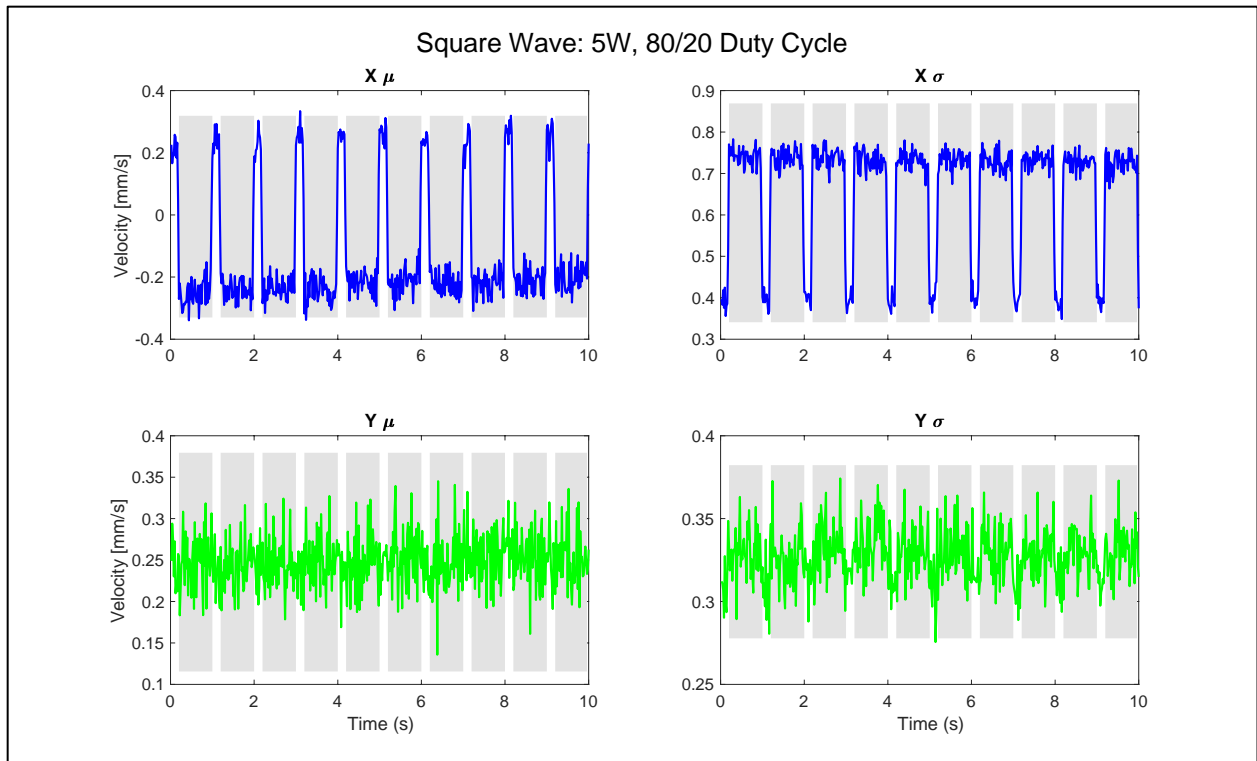
### 5.2.3. Campaign 9- “Capture” to “Flowing” Reverse Experiment

Another experiment we ran was designed to be a reverse effect of the work in campaign 7- “What happens to the dust cloud energy when going from a captured state to a flowing state?” This experiment was performed in February 2020 (3 days before Auburn shut down international

travel due to the COVID-19 pandemic), and this full dataset has not been received, but is also being shipped from France at this time.

In our campaign 9 experiment, we used a dc neon plasma, and injected and captured the particles with polarity switching (this was meant to be additional capture instances for reproducibility). After the dust cloud is captured, we let the cloud sit and form chains for our Baylor collaborators. Then, we used the optical manipulation laser to create a flow in the middle of the dust cloud at various intensities for a set interval. By changing the laser power multiple times, we would be able to see if there were multiple changes in the dust cloud temperature throughout the laser segment of the experiment.

Initial development for this project was done on the BU-PK-4 experiment, in Waco, Texas. This allowed us to use higher framerate cameras to see if there was a change in the dust cloud temperature when the manipulation laser was applied. Figure 5-2 shows that there is a change in the drift velocity ( $\mu$ ) and temperature ( $\sigma$ ) when a square wave with a duty cycle of 80% on / 20% off is applied to the gravity-based dust cloud.



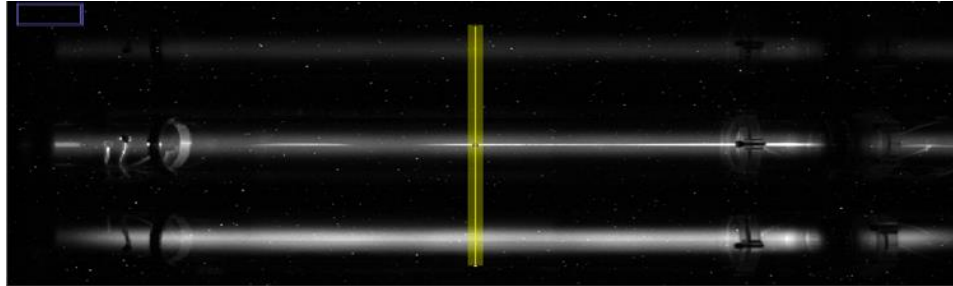
**Figure 5-2: Using a square wave modulation on the manipulation laser, we can induce a change in the dust temperature in the X direction. Data taken on BU-PK-4 in January 2020. The grey indicates the “on” portions of the duty cycle.**

This experiment at Baylor was the basis of our proposal for campaign 9, in which we applied a stepwise function to the laser’s power generator to see the changes in temperature at varying manipulation conditions. Due to the long delays in shipping from the ISS and throughout the European continent because of the COVID-19 pandemic, we have still not received our entire set of data from this experiment, so we have not been able to analyze the laser segments of this experiment. We also downlinked an injection to serve as replicability support for our campaign 7 results, so we do not have any laser measurements to analyze at this time.

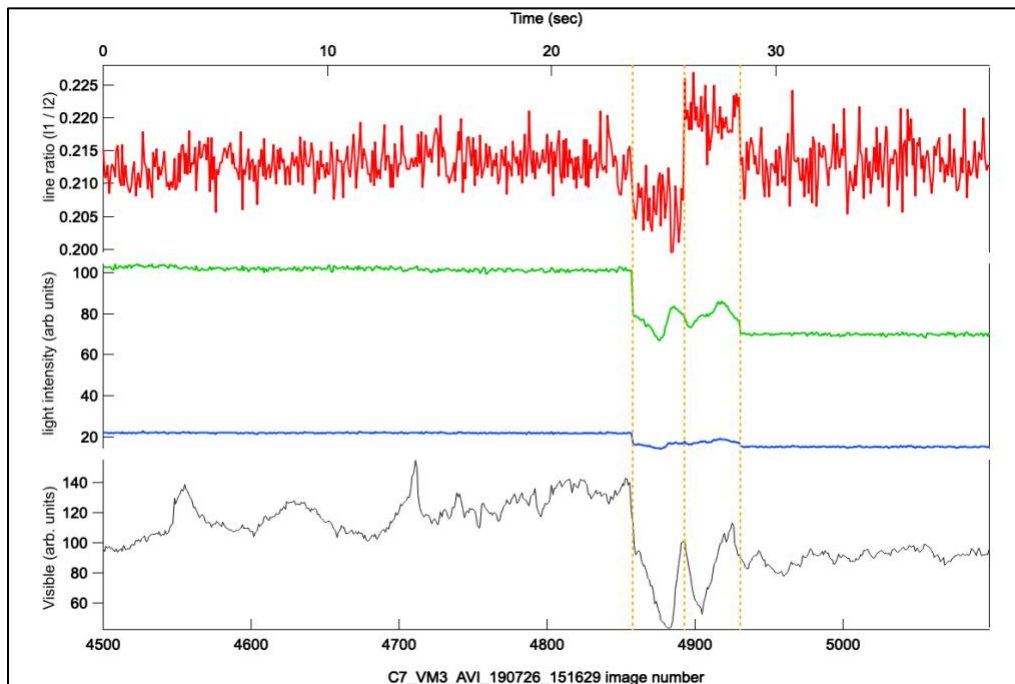
#### 5.2.4. All PO3 Data

Some other data that was taken with each experiment is the overview, PO3 camera, described in detail in section 2.1.3, camera settings. If we take a line-integrated segment on the

filtered camera views and then take a line ratio, we can determine if there are changes in the plasma. This has been done initially for  $p = 0.6$  mBar,  $I = 0.7$  mA and can be seen by the line in Figure 5-3 with the results in Figure 5-4.



**Figure 5-3: A line segment through the three views of PO3, in which we take pixel intensity values which correlate to the plasma glow strength.**



**Figure 5-4: Three line-integrated intensity traces and a line ratio of the filter views from VM3. The red line at the top is the line ratios of green/blue, where green is the view with the 585.2 nm filter, and blue is the view with the 703.2 nm filter. The black line is the intensity of the visible light view. The 3 yellow dashed lines indicate the time that Reinjection-, Reinjection+, and Polarity switching occur, respectively.**



The green trace is for the lower view of the plasma chamber, 585.2 nm line, the blue trace is for the upper view of the plasma chamber, 703.2 nm line, and the red is the resulting line ratio. This also shows the trace for the intensity of the visible light camera view in the middle of the frame as a reference. The yellow dashed lines are when we send the particle out with reinjection -, send them back in with reinjection +, and turn on polarity switching, respectively. This shows that there is a change in the plasma at the application of any of these electric field changes.

While we can see that the ratio changes with the change of the electric field, we are unable to extract further information without further benchmarking. This trend helps support that there are changes in the background plasma but we cannot quantify that change from this data at this time.

### **5.2.5. YOAK $\mu$ M**

Our results from YOAK $\mu$ M are a great first step for using simulation to give insight into our experimental work. However, there is much more that can be done with the simulation to better reproduce our results.

The first opportunity is to run this code in 3D and compare gravity and microgravity clouds. We have validated 2D and 3D clouds without gravity, and the results are consistent, so we chose to go with 2D for this work for simulation optimization. But to further investigate the differences between ground-based and microgravity clouds, the 3D cloud needs to be further utilized.

The second opportunity is to create time-varying variables for the system, specifically the Debye length used for the dust-dust interactions. Dr. Jeremiah Williams has converted the code to have time-varying variables for his electron beam research, so the basics are present and ready to be implemented. But we ultimately chose not to approach this at the time for my work because it was kind of a guessing game/ research time sink- how to model the Debye length, when to change

it based on experiments (like Figure 4-19), validations, etc. Our hypothesis of time-varying dependent screening length simulations would be that the dissipation results would be confirmed even further that the heating and dissipation are real and arise from interparticle structural energy.

## References

- [1] N. C. Adhikary, H. Bailung, A. R. Pal, J. Chutia, and Y. Nakamura, *Observation of Sheath Modification in Laboratory Dusty Plasma*, Physics of Plasmas **14**, 103705 (2007).
- [2] A. Douglass, V. Land, K. Qiao, L. Matthews, and T. Hyde, *Determination of the Levitation Limits of Dust Particles within the Sheath in Complex Plasma Experiments*, Physics of Plasmas (1994-Present) **19**, 013707 (2012).
- [3] J.-Y. Liu, Z.-X. Wang, X. Wang, Q. Zhang, X. Zou, and Y. Zhang, *The Bohm Criterion for the Dusty Plasma Sheath*, Physics of Plasmas **10**, 3507 (2003).
- [4] J.-X. Ma and M. Y. Yu, *Electrostatic Sheath at the Boundary of a Dusty Plasma*, 7 (n.d.).
- [5] S. Jaiswal, M. Menati, L. Couédel, V. H. Holloman, V. Rangari, and E. Thomas, *Effect of Growing Nanoparticle on the Magnetic Field Induced Filaments in a Radio-Frequency Ar/C<sub>2</sub>H<sub>2</sub> Discharge Plasma*, Jpn. J. Appl. Phys. **59**, SHHC07 (2020).
- [6] K. Ouaras, G. Lombardi, L. Couédel, C. Arnas, and K. Hassouni, *Microarcing-Enhanced Tungsten Nano and Micro-Particles Formation in Low Pressure High-Density Plasma*, Physics of Plasmas **26**, 023705 (2019).
- [7] L. Couédel, M. Mikikian, A. A. Samarian, and L. Boufendi, *Self-Excited Void Instability during Dust Particle Growth in a Dusty Plasma*, Physics of Plasmas **17**, 083705 (2010).
- [8] P. K. Shukla, *New Collective Processes in Dusty Plasmas: Applications to Space and Laboratories*, Plasma Phys. Control. Fusion **42**, B213 (2000).
- [9] C. K. Goertz, *Dusty Plasmas in the Solar System*, Reviews of Geophysics **27**, 271 (1989).

- [10] F. Verheest, *Dusty Plasmas in Application to Astrophysics*, Plasma Phys. Control. Fusion **41**, A445 (1999).
- [11] L. Boufendi, M. C. Jouanny, E. Kovacevic, J. Berndt, and M. Mikikian, *Dusty Plasma for Nanotechnology*, J. Phys. D: Appl. Phys. **44**, 174035 (2011).
- [12] M. Cavarroc, M. Mikikian, Y. Tessier, and L. Boufendi, *Nanostructured Silicon Thin Films Deposited Under Dusty Plasma Conditions*, IEEE Transactions on Plasma Science **36**, 1016 (2008).
- [13] S. I. Krasheninnikov et al., *Recent Progress in Understanding the Behavior of Dust in Fusion Devices*, Plasma Phys. Control. Fusion **50**, 124054 (2008).
- [14] J. Winter, *Dust in Fusion Devices—a Multi-Faceted Problem Connecting High- and Low-Temperature Plasma Physics*, Plasma Phys. Control. Fusion **46**, B583 (2004).
- [15] S. I. Krasheninnikov, R. D. Smirnov, and D. L. Rudakov, *Dust in Magnetic Fusion Devices*, Plasma Phys. Control. Fusion **53**, 083001 (2011).
- [16] B. A. Klumov, M. Rubin-Zuzic, and G. E. Morfill, *Crystallization Waves in a Dusty Plasma*, JETP Lett. **84**, 542 (2007).
- [17] T. Deka, A. Boruah, S. K. Sharma, and H. Bailung, *Observation of Self-Excited Dust Acoustic Wave in Dusty Plasma with Nanometer Size Dust Grains*, Physics of Plasmas **24**, 093706 (2017).
- [18] E. Thomas, B. Lynch, U. Konopka, M. Menati, S. Williams, R. L. Merlino, and M. Rosenberg, *Pattern Formation in Strongly Magnetized Plasmas: Observations from the Magnetized Dusty Plasma Experiment (MDPX) Device*, Plasma Phys. Control. Fusion **62**, 014006 (2019).

- [19] M. Menati, U. Konopka, and E. Thomas Jr., *Variation of Filamentation Phenomenon in Strongly Magnetized Plasma with Various Discharge Parameters*, Contributions to Plasma Physics **n/a**, e202100083 (n.d.).
- [20] A. M. Lipaev, V. I. Molotkov, A. P. Nefedov, O. F. Petrov, V. M. Torchinskii, V. E. Fortov, A. G. Khrapak, and S. A. Khrapak, *Ordered Structures in a Nonideal Dusty Glow-Discharge Plasma*, J. Exp. Theor. Phys. **85**, 1110 (1997).
- [21] A. Bouchoule and L. Boufendi, *Particulate Formation and Dusty Plasma Behaviour in Argon-Silane RF Discharge*, Plasma Sources Sci. Technol. **2**, 204 (1993).
- [22] M. Chaudhuri, A. V. Ivlev, S. A. Khrapak, H. M. Thomas, and G. E. Morfill, *Complex Plasma—the Plasma State of Soft Matter*, Soft Matter **7**, 1287 (2011).
- [23] B. Liu and J. Goree, *Superdiffusion and Non-Gaussian Statistics in a Driven-Dissipative 2D Dusty Plasma*, Phys. Rev. Lett. **100**, 055003 (2008).
- [24] Y. Feng, J. Goree, and B. Liu, *Solid Superheating Observed in Two-Dimensional Strongly Coupled Dusty Plasma*, Phys. Rev. Lett. **100**, 205007 (2008).
- [25] S. Ratynskaia, G. Regnoli, B. Klumov, and K. Rypdal, *Grain Transport in Three-Dimensional Soft Dusty Plasma States*, Physics of Plasmas **17**, 034502 (2010).
- [26] H. M. Thomas and G. E. Morfill, *Melting Dynamics of a Plasma Crystal*, Nature **379**, 6568 (1996).
- [27] A. Melzer, A. Homann, and A. Piel, *Experimental Investigation of the Melting Transition of the Plasma Crystal*, Phys. Rev. E **53**, 2757 (1996).
- [28] B. A. Klumov, *On Melting Criteria for Complex Plasma*, 14 (n.d.).
- [29] S. Jaiswal, T. Hall, S. LeBlanc, R. Mukherjee, and E. Thomas, *Effect of Magnetic Field on the Phase Transition in a Dusty Plasma*, Physics of Plasmas **24**, 113703 (2017).

- [30] M. Begum and N. Das, *Self-Diffusion as a Criterion for Melting of Dust Crystal in the Presence of Magnetic Field*, Eur. Phys. J. Plus **131**, 46 (2016).
- [31] L. Couedel, V. Nosenko, A. V. Ivlev, S. K. Zhdanov, H. M. Thomas, and G. E. Morfill, *Direct Observation of Mode-Coupling Instability in Two-Dimensional Plasma Crystals*, PHYSICAL REVIEW LETTERS **4** (2010).
- [32] L. Couédel, V. Nosenko, M. Rubin-Zuzic, S. Zhdanov, Y. Elskens, T. Hall, and A. V. Ivlev, *Full Melting of a Two-Dimensional Complex Plasma Crystal Triggered by Localized Pulsed Laser Heating*, Phys. Rev. E **97**, 043206 (2018).
- [33] D. Samsonov, S. K. Zhdanov, R. A. Quinn, S. I. Popel, and G. E. Morfill, *Shock Melting of a Two-Dimensional Complex (Dusty) Plasma*, Phys. Rev. Lett. **92**, 255004 (2004).
- [34] M. D. Acciarri, S. D. Baalrud, C. H. Moore, and N. Kot, *Strong Correlation Effects in Atmospheric Pressure Plasmas*, in *2022 IEEE International Conference on Plasma Science (ICOPS)* (IEEE, 2022), pp. 1–1.
- [35] M. K. Warrens, G. M. Gorman, S. J. Bradshaw, and T. C. Killian, *Expansion of Ultracold Neutral Plasmas with Exponentially Decaying Density Distributions*, Physics of Plasmas **28**, 022110 (2021).
- [36] V. F. Kovalev, K. I. Popov, V. Yu. Bychenkov, and W. Rozmus, *Laser Triggered Coulomb Explosion of Nanoscale Symmetric Targets*, Physics of Plasmas **14**, 053103 (2007).
- [37] V. Yu. Bychenkov and V. F. Kovalev, *Coulomb Explosion in a Cluster Plasma*, Plasma Phys. Rep. **31**, 178 (2005).
- [38] M. Kardar, *Statistical Physics of Particles* (Cambridge University Press, 2007).
- [39] K. F. Ness and R. E. Robson, *Velocity Distribution Function and Transport Coefficients of Electron Swarms in Gases. II. Moment Equations and Applications*, Phys. Rev. A **34**, 2185 (1986).

- [40] G. A. Bird, *The Velocity Distribution Function within a Shock Wave*, J. Fluid Mech. **30**, 479 (1967).
- [41] W. G. Pilipp, H. Miggenrieder, M. D. Montgomery, K.-H. Mühlhäuser, H. Rosenbauer, and R. Schwenn, *Characteristics of Electron Velocity Distribution Functions in the Solar Wind Derived from the Helios Plasma Experiment*, Journal of Geophysical Research: Space Physics **92**, 1075 (1987).
- [42] Z. Ehsan, N. L. Tsintsadze, and S. Poedts, *A Modified Orbital Motion Limited (OML) Theory*, ArXiv:1110.6304 [Physics] (2011).
- [43] A. Barkan, N. D'Angelo, and R. L. Merlino, *Charging of Dust Grains in a Plasma*, Phys. Rev. Lett. **73**, 3093 (1994).
- [44] J. E. Allen, B. M. Annaratone, and U. de Angelis, *On the Orbital Motion Limited Theory for a Small Body at Floating Potential in a Maxwellian Plasma*, Journal of Plasma Physics **63**, 299 (2000).
- [45] J. E. Allen, *Probe Theory - the Orbital Motion Approach*, Phys. Scr. **45**, 497 (1992).
- [46] F. Chen, *Introduction to Plasma Physics and Controlled Fusion* (n.d.).
- [47] P. K. Shukla and A. A. Mamun, *Introduction to Dusty Plasma Physics* (CRC Press, 2015).
- [48] A. Melzer, *Physics of Dusty Plasmas* (n.d.).
- [49] R. E. Kidder and H. E. deWitt, *Application of a Modified Debye-Hückel Theory to Fully Ionized Gases*, J. Nucl. Energy, Part C Plasma Phys. **2**, 218 (1961).
- [50] T. Kennedy, *Debye-Hückel Theory for Charge Symmetric Coulomb Systems*, Commun.Math. Phys. **92**, 269 (1983).
- [51] H. Totsuji, T. Kishimoto, and C. Totsuji, *Structure of Confined Yukawa System (Dusty Plasma)*, Phys. Rev. Lett. **78**, 3113 (1997).

- [52] G. Livadiotis and D. J. McComas, *Electrostatic Shielding in Plasmas and the Physical Meaning of the Debye Length*, Journal of Plasma Physics **80**, 341 (2014).
- [53] P. S. Epstein, *On the Resistance Experienced by Spheres in Their Motion through Gases*, Phys. Rev. **23**, 710 (1924).
- [54] D. Samsonov and J. Goree, *Instabilities in a Dusty Plasma with Ion Drag and Ionization*, Physical Review E **59**, 1047 (1999).
- [55] E. S. Dzlieva, M. A. Ermolenko, V. Yu. Karasev, S. I. Pavlov, L. A. Novikov, and S. A. Maiorov, *Control of Ion Drag in a Dusty Plasma*, Jetp Lett. **100**, 703 (2015).
- [56] B. Liu, J. Goree, V. Nosenko, and L. Boufendi, *Radiation Pressure and Gas Drag Forces on a Melamine-Formaldehyde Microsphere in a Dusty Plasma*, Physics of Plasmas **10**, 9 (2003).
- [57] T. Hatano and S. Sasa, *Steady-State Thermodynamics of Langevin Systems*, Phys. Rev. Lett. **86**, 3463 (2001).
- [58] R. Gopalakrishnan and C. J. Hogan, *Coulomb-Influenced Collisions in Aerosols and Dusty Plasmas*, Phys. Rev. E **85**, 026410 (2012).
- [59] G. E. Morfill, M. Rubin-Zuzic, H. Rothermel, A. V. Ivlev, B. A. Klumov, H. M. Thomas, U. Konopka, and V. Steinberg, *Highly Resolved Fluid Flows: "Liquid Plasmas" at the Kinetic Level*, Phys. Rev. Lett. **92**, 175004 (2004).
- [60] A. Ivlev, G. Morfill, H. Lowen, and C. P. Royall, *Complex Plasmas And Colloidal Dispersions: Particle-Resolved Studies Of Classical Liquids And Solids* (World Scientific Publishing Company, 2012).
- [61] P. Hofmann et al., *Complex Plasma Research on ISS: PK-3 Plus, PK-4 and Impact/Plasmalab*, Acta Astronautica **63**, 53 (2008).



- [62] M. Y. Pustyl'nik et al., *Plasmakristall-4: New Complex (Dusty) Plasma Laboratory on Board the International Space Station*, Review of Scientific Instruments **87**, 093505 (2016).
- [63] J. E. Allen, *Probe Theory - the Orbital Motion Approach*, Phys. Scr. **45**, 497 (1992).
- [64] A. V. Ivlev et al., *First Observation of Electrorheological Plasmas*, Phys. Rev. Lett. **100**, 095003 (2008).
- [65] H. M. Thomas et al., *Complex Plasma Laboratory PK-3 Plus on the International Space Station*, New J. Phys. **10**, 033036 (2008).
- [66] M. Y. Pustyl'nik et al., *Three-Dimensional Structure of a String-Fluid Complex Plasma*, Phys. Rev. Research **2**, 033314 (2020).
- [67] A. V. Ivlev, M. H. Thoma, C. R  th, G. Joyce, and G. E. Morfill, *Complex Plasmas in External Fields: The Role of Non-Hamiltonian Interactions*, Phys. Rev. Lett. **106**, 155001 (2011).
- [68] *DaVis Software for Intelligent Imaging* (LaVision, GmbH, 2013).
- [69] R. J. Adrian, *Twenty Years of Particle Image Velocimetry*, Exp Fluids **39**, 159 (2005).
- [70] A. Melling, *Tracer Particles and Seeding for Particle Image Velocimetry*, Meas. Sci. Technol. **8**, 1406 (1997).
- [71] W. Humphreys, S. Bartram, and J. Blackshire, *A Survey of Particle Image Velocimetry Applications in Langley Aerospace Facilities*, in *31st Aerospace Sciences Meeting* (American Institute of Aeronautics and Astronautics, Reno,NV,U.S.A., 1993).
- [72] C. Brossard, J. C. Monnier, P. Barricau, F. X. Vandernoot, y. Le Sant, F. Champagnat, and G. Le Besnerais, *Principles and Applications of Particle Image Velocimetry*, Aerospace Lab 1 (2009).
- [73] J. D. Williams and E. Thomas, *Measurement of the Kinetic Dust Temperature of a Weakly Coupled Dusty Plasma*, Physics of Plasmas **14**, 063702 (2007).

- [74] R. Fisher and E. Thomas, *Thermal Properties of a Dusty Plasma in the Presence of Driven Dust Acoustic Waves*, IEEE Transactions on Plasma Science **38**, 833 (2010).
- [75] E. Thomas Jr. and J. D. Williams, *Private Communication to DLR; PK-4 PIV Calibration Report, Part 1 and Part 2*, (2015).
- [76] E. Thomas, J. Williams, and C. Rath, *Benchmarking Particle Image Velocimetry Measurements Applied to Dusty Plasmas*, IEEE Transactions on Plasma Science **38**, 892 (2010).
- [77] J. D. Williams and E. Thomas, *Initial Measurement of the Kinetic Dust Temperature of a Weakly Coupled Dusty Plasma*, Physics of Plasmas **13**, 063509 (2006).
- [78] J. D. Williams, *Application of Tomographic Particle Image Velocimetry to Studies of Transport in Complex (Dusty) Plasma*, Physics of Plasmas **18**, 050702 (2011).
- [79] E. Thomas, *Observations of High Speed Particle Streams in Dc Glow Discharge Dusty Plasmas*, Physics of Plasmas **8**, 329 (2001).
- [80] M. Schwabe, M. Rubin-Zuzic, C. R ath, and M. Pustylnik, *Image Registration with Particles, Exemplified with the Complex Plasma Laboratory PK-4 on Board the International Space Station*, Journal of Imaging **5**, 3 (2019).
- [81] R. Komanduri, N. Chandrasekaran, and L. M. Raff, *MD Simulation of Indentation and Scratching of Single Crystal Aluminum*, Wear **240**, 113 (2000).
- [82] S. M. Fatemi and M. Foroutan, *Recent Developments Concerning the Dispersion of Carbon Nanotubes in Surfactant/Polymer Systems by MD Simulation*, J Nanostruct Chem **6**, 29 (2016).
- [83] S. O. Nielsen, R. E. Bulow, P. B. Moore, and B. Ensing, *Recent Progress in Adaptive Multiscale Molecular Dynamics Simulations of Soft Matter*, Physical Chemistry Chemical Physics **12**, 12401 (2010).

- [84] R. A. Jefferson, M. Cianciosa, and E. Thomas, *Simulations of One- and Two-Dimensional Complex Plasmas Using a Modular, Object-Oriented Code*, *Physics of Plasmas* **17**, 113704 (2010).
- [85] D. Ticoş, A. Scurtu, J. D. Williams, L. Scott, E. Thomas, D. Sanford, and C. M. Ticoş, *Rotation of a Strongly Coupled Dust Cluster in Plasma by the Torque of an Electron Beam*, *Phys. Rev. E* **103**, 023210 (2021).
- [86] C. Runge, *Ueber die numerische Auflösung von Differentialgleichungen*, *Math. Ann.* **46**, 167 (1895).
- [87] E. Thomas Jr. and J. D. Williams, *Private Communication to DLR*, (2018).
- [88] P. Hartmann, M. Rosenberg, Z. Juhasz, L. S. Matthews, D. L. Sanford, K. Vermillion, J. C. Reyes, and T. W. Hyde, *Ionization Waves in the PK-4 Direct Current Neon Discharge*, *Plasma Sources Sci. Technol.* (2020).
- [89] G. Wang et al., *Review of Stopping Power and Coulomb Explosion for Molecular Ion in Plasmas*, *Matter and Radiation at Extremes* **3**, 67 (2018).
- [90] L. S. Matthews, D. L. Sanford, E. G. Kostadinova, K. S. Ashrafi, E. Guay, and T. W. Hyde, *Dust Charging in Dynamic Ion Wakes*, *Physics of Plasmas* **27**, 023703 (2020).
- [91] K. Vermillion et al., *Self-Organization of Dust Chains in the Presence of Ionization Waves*, *ArXiv:2111.00374 [Physics]* (2021).

## **Appendix A: PK-4 Documentation**

### **A.1. Campaign 7**

Campaign 7 served as our first microgravity experiment, and we combined with Baylor University researchers to utilize the proposal timeslot to fullest. We were specifically looking at the injections of a dust cloud, and they were investigating the string formation after the dust cloud is captured. They also wanted to look at the effect of polarity switching on the string formations, so we had a “polarity ramp down” section for two of the pressures. This actually turned out to be beneficial for us as well, because we could see that there is a change in the dust kinetic temperature when this frequency changes, as discussed in Section 3.4.2. The files begin on the next page.

## A.1.1. Proposal

The proposal document we submitted to the PK-4 operating team in February 2019.

### Proposal for a flight experiment on PK-4

Title: Measurements of charging, energy, dust chains and wave breaking in a flowing dusty plasma

Submitted: 4 February 2019

Team Members:

Auburn Univ. – Lori Scott, Surabhi Jaiswal, Uwe Konopka, Edward Thomas, Jr., Brandon Doyle,  
Dylan Funk  
Baylor Univ. – Truell Hyde, Lorin Matthews, Eva Kostadinova  
UCSD – Marlene Rosenberg  
Wittenberg Univ. – Jeremiah Williams

Scientific goals:

The PK-4 microgravity laboratory has unique capabilities for studying fluid-like behavior in complex plasmas. In particular, through the use of the polarity switching we can control both the symmetric formation of ion wakes and the net flow of microparticles in the plasma. Our four university teams have planned a coordinated series of experiments that make use of similar operational parameters to explore four phases of behavior in PK-4.

a) *Redistribution of flow kinetic energy at the onset of polarity switching.* Using particle image velocimetry (PIV) techniques developed by the Auburn team, we will measure the drift and random components of the particle velocities during the transition from a high-speed flow through the application of polarity switching (Table 1). Based upon an analysis of ground data and some preliminary microgravity data, we have identified a systematic, spatial redistribution of the particle velocities that persists for some time (a few seconds) after the onset of polarity switching. We plan to combine the application of PIV to thermal measurements, changes in the emission spectra of the background plasma, and new MD simulations of the PIV measurement in flowing dusty plasmas, to study how the dust particle energy is redistributed.

Leads: L. Scott, E. Thomas, J. Williams; Experimental phase: particle injection (all experiments)

b) *Self-excited dynamics of multi-chain dust clouds<sup>1</sup>.* The Baylor/UCSD team will investigate how the wave properties and onset of dynamical instabilities of multi-chain dusty plasmas depend on the current and frequency (Table 1.) In-house analysis techniques applied to previously collected PK-4 data have proven able to yield the wave properties, diffusion regime, local confinement potential, and global confinement forces. The results will be compared against two simulations in use at Baylor: i. MD simulation that calculates the dust charge within the ion wake by resolving the dynamics of both dust grains and streaming ions and ii. Particle-in-Cell code that simulates the dynamics of the plasma species including ion collisions in a DC discharge. The obtained dynamics of the multi-chain dust clouds will reveal the underlying interparticle potential and fundamental plasma characteristics.

Leads: T. Hyde, L. Matthews; Experiment phase: extended stable cloud after polarity switching

c) *Nonlinear wave breaking.* Wave breaking is a phenomenon in which an increasing amplitude leads to a nonlinear steepening and an eventual density “break” as the system can no longer support the wave. The study of wave breaking in dusty plasma is very limited and driven flows at high dust number density in PK-4 may allow this phenomenon to be studied. This proposed study builds upon ground-based laboratory studies performed by Lin I and co-workers. This experiment will focus on flowing dust clouds in which the

---

<sup>1</sup> The term *multi-chain dust clouds* refers to dusty plasmas where multiple filamentary structures tend to form. Such structures were observed in PK-4-Campaign 1 (for example, VM1-AVI-151028-134729 and VM1-AVI-151028-093050).

strength of external electric field and increasingly higher dust densities will be used (via additional shakes) to explore the wave-breaking regime in PK-4 (Table 2). The observation would be backed by theoretical model based on Lagrangian formulation of nonlinear DAWs and examine the maximum amplitude (electric field and density) sustained by these DAWs before breaking.

Lead(s): S. Jaiswal; Experiment phase: flowing cloud

d) *Extracting particle charge from dust-dust collisions.* One of the most fundamental parameters of a complex/dusty plasma is the particle charge collected on the dust particle surfaces. The determination of the individual acquired particle charge is difficult. However specifically prepared collision experiments indicate that an effective particle charge can be derived from the analysis of the particle dynamics during a collision. We will investigate the video data obtained from the experiments described above for dust particle pair collisions that are suitable for further analysis. This data-mining project will be accompanied by the analysis of molecular dynamics simulations of collision events in a dense dusty plasma cloud environment to derive suitable criteria for when a collision event can be used to derive a meaningful effective dust particle charge. Experiments with highly chaotic motion or substantial strength of dynamics are suitable for this analysis.

Lead(s): U. Konopka, B. Doyle, D. Funk, E. Thomas; Experiment phase: all configurations

It is proposed that all of these overlapping studies can be performed during a single 30-minute long experiment on PK-4. The timeline is illustrated in Figure 1. The proposed experiment would be performed in two stages during a single, 30-minute campaign on PK-4. Common parameters for the experiments:

Particle size: 3.38  $\mu\text{m}$  Gas: Neon  
Polarity switching frequency: see table Plasma mode: DC only  
PO Cameras FOV: 1600 x 480 (central cutout), 70 fps

**Table 1: Stage 1 conditions for string experiments:  $p = 0.13$  mBar; duty cycle = 50%**

4 injections	Injection 1			Injection 2	Injection 3	Injection 4	
7 datasets	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7
I [mA]	0.7	0.35	1	0.7	0.35	0.7	0.7
f [Hz]	500	500	500	500	500	500	250, 200, 150, 100, 50, 25
Data collection [s]	60	60	60	50	50	60	60
Technical time [s]*	80			80	80	80	
Total time [s]	260			130	130	200	
# of shakes (15-20ms per shake)	1-3			1-3	1-3	1-3	

**Table 2: Stage 2 conditions for wave experiments:  $p = 0.4$  mBar**

3 injections	Injection 1			Injection 2			Injection 3		
9 datasets	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9
I [mA]	0.5	1.0	1.5	0.5	1.0	1.5	0.5	1.0	1.5
f [Hz]	500	500	500	500	500	500	500	500	500
Duty Cycle (%)	0 - 100	0-100	0- 100	0 - 100	0-100	0- 100	0 - 100	0-100	0- 100
	20-80	20-80	20-80	20-80	20-80	20-80	20-80	20-80	20-80
Data collection [s]	50	50	50	50	50	50	50	50	50
Technical time [s] <sup>1</sup>	80			80			80		
Total time [s]	230			230			230		
# of shakes (15-20ms per shake)	3			5			7		

<sup>1</sup>Technical time for each injection assumes 20s for dust particle transport, 20s for PO cameras/laser adjustment, and 40s for flushing the cloud.

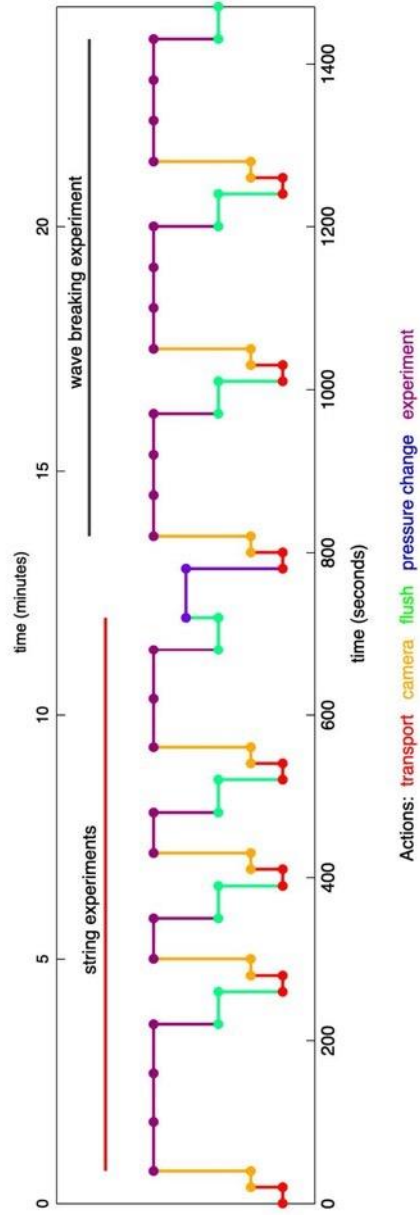



Figure 1: Timeline for stage 1 and 2 experiments



### **A.1.2. Flowchart**

The flowchart for the code that ran our experiment for campaign 7.

	<b>Measurements of charging, energy, and microparticle chains</b>					Operational information				
	Script name	ChargeEnergyChains_C07 .epl					Script executions	1		
Keywords: Charging, chains						Successful trappings	3			
Investigation of energy transfer when stopping a drifting microparticle cloud. Investigation of microparticle chains in a wide parameter range						Gas	neon			
						Crew activity	Microparticle trapping			
Module	Power	HV	RF	TMTA	PO	OM	DCGC	VC	Pos	
Usage	x	x		x	x		x	x	x	

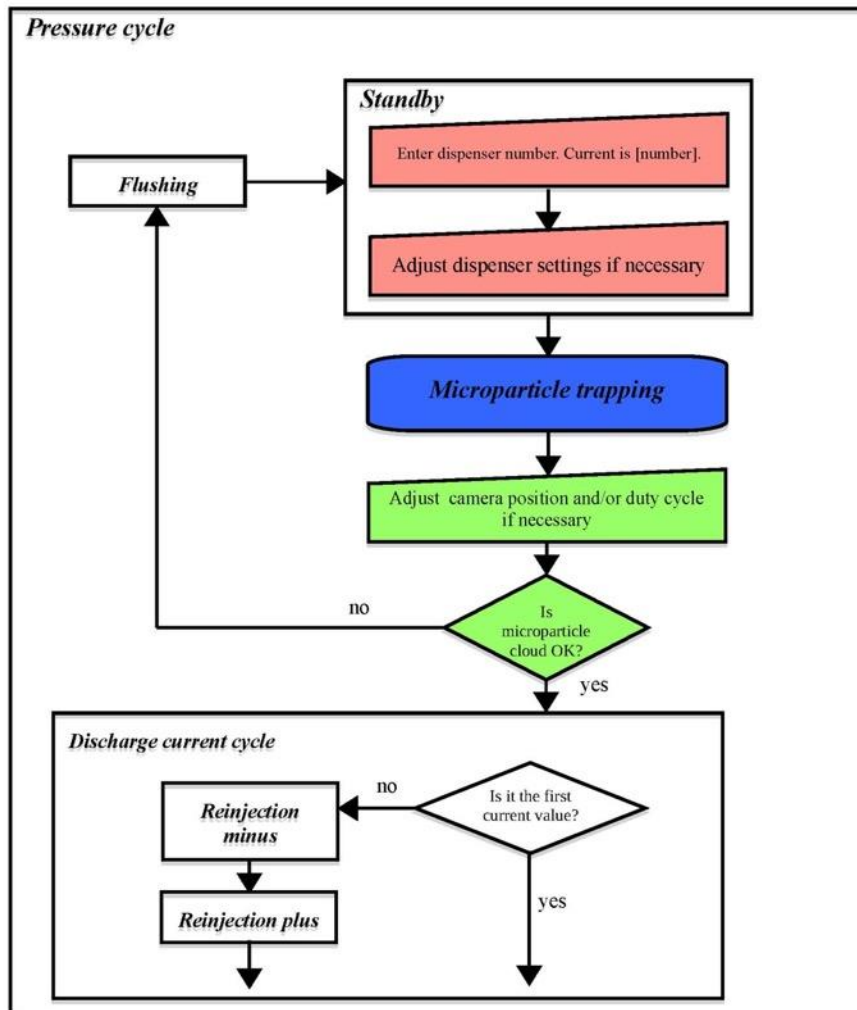
Experiment team		
Name	Institution	E-mail address
Mikhail Pustynnik	DLR	mikhail.pustynnik@dlr.de
Tetyana Antonova	DLR	tetyana.antonova@dlr.de
Lori Scott	Auburn University	lcs0044@tigermail.auburn.edu
Edward Thomas	Auburn University	etjr@auburn.edu
Uwe Konopka	Auburn University	uzk0003@auburn.edu
Surabhi Jaiswal	Auburn University	szj0071@auburn.edu
Jeremiah Williams	Wittenberg University	jwilliams@wittenberg.edu
Truell Hyde	Baylor University	Truell_Hyde@baylor.edu
Eva Kostadinova	Baylor University	eva_kostadinova@baylor.edu
Lorin Matthews	Baylor University	lorin_matthews@baylor.edu
Marlene Rosenberg	UCSD	rosenber@ece.ucsd.edu
Peter Hartmann	Hungarian Academy of Sciences	hartmann.peter@wigner.mta.hu
Andrey Lipaev	JIHT RAS	lipaev@mail.ru

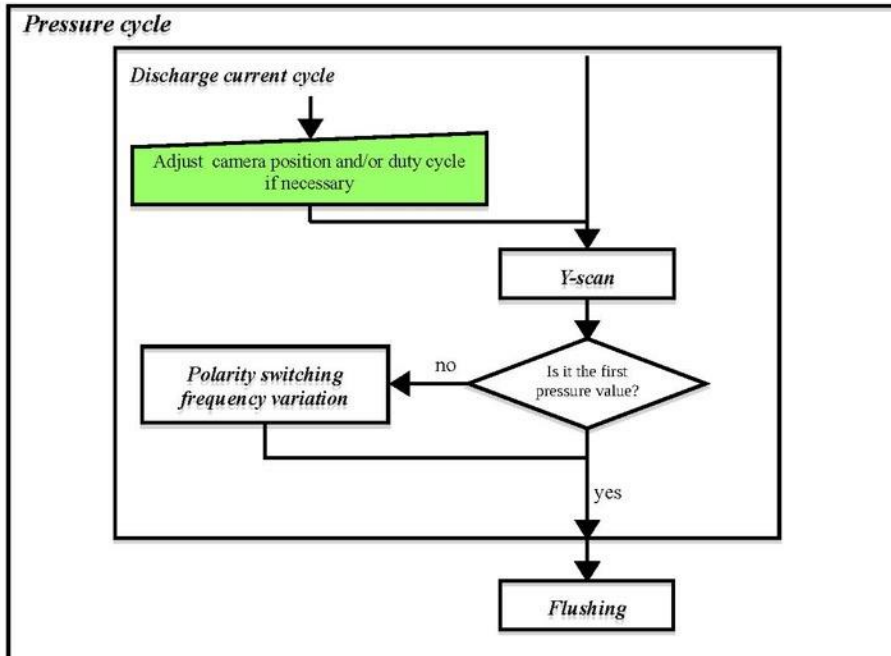
1 | Measurements of charging, energy, and microparticle chains



## Measurements of charging, energy, and microparticle chains

Flowchart






Parameters				
	Constant	Variable within experimental run	Feedback from HCl variation in pauses	Variable between experimental runs
Explf		<p><b>Standby, Discharge current cycle:</b> Interface valve open Bypass valve closed</p> <p><b>Flushing:</b> Two cycles of Interface and Bypass valves to increase the output pressure</p>		<p><b>Pressure cycle:</b> <b>Standby, Discharge current cycle:</b> Gas flow [sccm]: pressure-defined, 0.5, 0.5</p>
HV	EM electrode not used	<p><b>Microparticle trapping</b> Active electrode: pulse generator, polarity switching with 500 Hz frequency, 1 mA current, residual-drift-corrected 50% duty cycle</p> <p><b>Discharge current cycle:</b> pulse generator, polarity switching with 500 Hz frequency, residual-drift-corrected 50% duty cycle, current set [mA]: 1.0, 0.7, 0.35</p> <p><b>Reinjection minus</b> dc discharge with negative polarity</p> <p><b>Reinjection plus</b> dc discharge with positive polarity</p> <p><b>Polarity switching variation cycle:</b> pulse generator, polarity switching frequency, residual-drift-corrected</p>	<p><b>Microparticle trapping:</b> offset (psOffset): 1/2</p>	<p><b>Pressure cycle:</b> <b>Reinjection:</b> Duration of dc discharge [s]: 0.5, 1.0, 2.0</p>

		50% duty cycle, frequency set [Hz]: 250, 150, 100, 50, 25		
TMTA	Not parameterized			
PO	<p>PO cameras: gain 0 black level 600 FoV: 1600x480 central cutout frame rate 70 fps</p> <p>PO camera 1 on full screen</p> <p>PGO cameras: full FoV gain 512 frame rate 15 fps exposure time 30 ms</p> <p>Spectrum readout interval 4 s</p> <p>Spectrometer integration time 750 ms</p>	<p><b>Standby, Microparticle trapping, Reinjection minus, Reinjection plus, Y-scan:</b> Exposure time of PO cameras 14 ms</p> <p><b>Standby, Microparticle trapping, Reinjection minus, Reinjection plus:</b> Illumination laser current 0.9 A</p> <p><b>Discharge current cycle, Flushing:</b> Illumination laser current dispenser-dependent: 0.45 A – for dispenser 6 0.7 A – for dispenser 5 0.9 A – for dispenser 4</p> <p><b>Polarity switching variation cycle:</b> Illumination laser current [A]: dispenser-dependent, dispenser-dependent, dispenser-dependent, 0.9, 1.2</p> <p>Exposure time of PO cameras [ms]: 14, 14, 14, 7, 3.5</p>		
DCGC	<p>Dispenser 5 T_off 100 ms T_on 30 ms Shakes 3</p>	<p><b>Flushing:</b> IQF200 valve open IQP600 valve closed Gas reservoir valve</p>		

5 | Measurements of charging, energy, and microparticle chains

		closed  <b>Standby, Discharge current cycle:</b> IQF200 valve closed IQP600 valve open Gas reservoir valve open		
VC				<b>Pressure cycle</b> [mbar]: 0.2, 0.4, 0.6
Pos	X-axis : 100 mm Illu-laser axis : 115 mm RF axis : 0 OM-laser axis : 0	<b>Standby, Discharge current cycle, Flushing:</b> Y-axis : 15 mm  <b>Y-scan :</b> Y-axis : 8-22 mm velocity 1 mm/s		

		<b>Measurements of charging, energy, and microparticle chains</b>		Step-by-step procedure
Step number	Description	Execution	Remark	
1.	<b><u>Begin Pressure cycle</u></b>	Ground		
1.1.	Set current gas flow	Script		
1.2.	<b><i>Begin Standby</i></b>			
1.3.	Set illumination laser current 0.9 A	Script		
1.4.	Set current pressure	Script	Use setLowPressure	
1.4.1.	User action: Enter dispenser number	Ground	CP 1: Possible LOS break. Range 4-6	
1.4.2.	User action: Adjust dispenser parameters if necessary	Ground	CP 2: Possible LOS break or crew notification. Continue only if crew ready for the activity.	
1.5.	<b><i>End Standby</i></b>			
1.6.	Start recording all cameras	Script		
1.7.	Start spectrometer acquisition	Script		
1.8.	Wait 10 s	Script		
1.9.	<b><i>Microparticle trapping</i></b>	Crew		



7 | **Measurements of charging, energy, and microparticle chains**



1.10.	User action: Adjust camera position if necessary.	Ground	CP 3: Fast execution on the scientists' request
1.11.	Adapt illumination laser position	Script	
1.12.	User action: Is microparticle cloud OK? - y/n	Ground	CP 4: Fast execution on the scientists' request
1.12.1.	n: <b>Flushing</b>	Script	
1.12.2.	n: Wait 10 s	Script	
1.12.3.	n: Stop spectrometer acquisition	Script	
1.12.4.	n: Stop recording all cameras	Script	
1.12.5.	n: <b>Go To</b> 1.2	Script	
1.13.	Readout psOffset	Script	
1.14.	Set dispenser-dependent illumination laser current	Script	
1.15.	<b>Begin Discharge current cycle</b>	Script	
1.15.1.	Is this the very first current value? (y/n)	Script	
1.15.1.1.	n: Set illumination laser current 0.9 A	Script	
1.15.1.2.	n: <b>Begin Reinjection minus</b>		
1.15.1.2.1.	n: Set dc plasma with negative current value	Script	
1.15.1.2.2.	n: Wait current duration	Script	

1.15.1.3.	n: <b>End Reinjection minus</b>		
1.15.1.4.	n: <b>Begin Reinjection plus</b>		
1.15.1.4.1.	n: Set dc plasma with positive current value	Script	
1.15.1.4.2.	n: Wait current duration	Script	
1.15.1.5.	n: <b>End Reinjection plus</b>		
1.15.1.6.	n: Set polarity-switched plasma with current discharge current value, 500 Hz polarity-switching frequency, 50% residual-drift-corrected duty cycle	Script	
1.15.1.7.	Wait 5 s	Script	
1.15.1.8.	n: Set dispenser-dependent illumination laser current	Script	
1.15.1.9.	n: User action: Adjust camera position if necessary.	Ground	<b>CP 5: Fast execution on the scientists' request</b>
1.15.1.10.	n: Adapt illumination laser position	Script	
1.15.1.11.	n: Readout psOffset	Script	
1.16.	Wait 50 s	Script	
1.17.	<b>Y-scan</b>	Script	
1.18.	Is this the very first pressure value? (y/n)	Script	
1.18.1.	n: <b>Begin Polarity switching frequency variation</b>	Script	

1.18.1.1.	n: Set polarity-switched plasma with current discharge current value, current polarity-switching frequency, 50% residual-drift-corrected duty cycle	Script	
1.18.1.2.	n: Set current illumination laser current	Script	
1.18.1.3.	n: Set current exposure time of PO cameras	Script	
1.18.1.4.	n: Wait 10 s	Script	
1.18.2.	n: <b>Repeat Polarity switching frequency variation</b>	Script	
1.19.	<b>Repeat Discharge current cycle</b>	Script	
1.20.	Set dispenser-dependent illumination laser current	Script	
1.21.	Set exposure time of PO cameras 14 ms	Script	
1.22.	<b>Flushing</b>	Script	
1.23.	Wait 10 s	Script	
1.24.	Stop spectrometer acquisition	Script	
1.25.	Stop recording all cameras	Script	
2.	<b>Repeat Pressure cycle</b>	Script	

Step number	User action message/ Experiment block	Measurements of charging, energy, and microparticle chains			To be filled on console			CADMOS OPS			cons. crew Scientist			Console protocol
		P = 0.2 mbar	P = 0.4 mbar	P = 0.6 mbar	Date	Start time								
1.	Standby Enter dispenser number													Possible LOS break. Range 4-6
2.	Adjust dispenser [number] settings													Possible LOS break or crew notification. Continue only if crew ready for the activity.
														
	Microparticle trapping													Crew activity
3.	Adjust camera position if necessary													Fast execution on the scientists' request
4.	Is microparticle cloud OK?													Fast execution on the scientists' request
5.	Reinjection minus													
	Reinjection plus													
	Discharge current cycle													Fast execution on the scientists' request
	Adjust camera position if necessary													
	Y-Scan													
	Polarity switching frequency variation													
	Flushing													
														

### A.1.3. Lori's Excel Timing Sheet

These excel sheets show the timing for frames and timestamps when transitions occur in the campaign 7 experiments.

#### Ground Experiment Timing:

Log File Name	PK4_chargeenergychains_c07_190524_083958.log								
	0.2 mBar Pressure			0.4 mBar Pressure			0.6 mBar Pressure		
Video File Name	VM1_AVL 190524_084739.avi	VM2_AVL 190524_084740.avi	VM3_AVL 190524_084741.avi	VM1_AVL 190524_085542.avi	VM2_AVL 190524_085544.avi	VM3_AVL 190524_085545.avi	VM1_AVL 190524_090631.avi	VM2_AVL 190524_090632.avi	VM3_AVL 190524_090633.avi
	1.0 mA DC Current			1.0 mA DC Current			1.0 mA DC Current		
First particle	1692			1731	??		1663	1636?	
Beginning of cloud	1787			1812	1681		1873	1754	
PS (500 Hz)	1898-1905			1941	1837		2058	1989	
Compression							??	1993-2000	
PO Laser current drop	3883-3884			4457-4460	4353-4354		3884-3887	3816-2818	
Y-scan A (begin)	7337			8252	8046		7557	7479	
Y-scan B (back to mid)	8586			9045	9036		8571	8350	
Y-scan C (end)	9869			10349	10183		9587	9514	
PS (250 Hz)	NA	NA	NA	10575			9935		
PS (150 Hz)	NA	NA	NA	?			?		
PS (100 Hz)	NA	NA	NA	?			?		
PS (50 Hz)	NA	NA	NA						
PS (25 Hz)	NA	NA	NA						
	0.7 mA DC Current			0.7 mA DC Current			0.7 mA DC Current		
Reinjection Minus	9986			14275			13595	13526-13527	
Reinjection Plus	?			14439	14334		13760	13691	
PS (500 Hz)	10328			14617	14511		13939	13870	
Shock Wave	too violent to tell			14617-14625			13939-13950	13870-13884	
PO Laser current drop	10704-10705			14994-14995			14316-14317	14247-14248	
Y-scan A	14793			20889			19391	19406	
Y-scan B	15772			21936			20459	20443	
Y-scan C	17025			23184			21636	21717	
PS (250 Hz)	NA	NA	NA						
PS (150 Hz)	NA	NA	NA				?		
PS (100 Hz)	NA	NA	NA				?		
PS (50 Hz)	NA	NA	NA						
PS (25 Hz)	NA	NA	NA						
	0.35 mA DC Current			0.35 mA DC Current			0.35 mA DC Current		
Reinjection Minus	17228			27114			25627	25558	
Reinjection Plus	17462-17463			27349	27243		25862	25793	
PS (500 Hz)	17712-17713	17643		27597	27491		26111	26042	
Shock Wave	?? 17714-17725			None?			26111-26123	26042-26057	
PO Laser current drop	18089-18090			27978-27980			26490-26494	26423	
Y-scan A	22415-23954-24917			32136-33190-34701			32128-33191-34349	32064-33068-34192	
Y-scan B									
Y-scan C									
PS (250 Hz)	NA	NA	NA				?		
PS (150 Hz)	NA	NA	NA				?		
PS (100 Hz)	NA	NA	NA						
PS (50 Hz)	NA	NA	NA						
PS (25 Hz)	NA	NA	NA						
Flushing	24940			38884			38520	38455	
Analysis Notes				-0.7mA shock wave hits the standing waves below			-Reinjection minus has a particle curve as well! -Look at the Shock wave of PS after reinjection (both currents) 0.7 mA: 13939-13950, 0.35 mA: 26111-26123 0.7 Reinject switch the streaks angles up		

#### Flight Experiment Timing:

Log File Name	02 mBar Pressure										04 mBar Pressure										06 mBar Pressure									
Video File Name	Screen Capture (timestamp)	Screen Capture (GMT top right)	WVA1_AV1 (150726.144755.m)	WVA2_AV1 (150726.144756.m)	WVA3_CALCULATED (FRAME FROM WVA1)	WVA4_AV1 (NA)	Screen Capture (timestamp)	Screen Capture (GMT top right)	WVA1_AV1 (150726.150151.m)	WVA2_AV1 (150726.150152.m)	WVA3_CALCULATED (FRAME FROM WVA1)	WVA4_AV1 (NA)	Screen Capture (timestamp)	Screen Capture (GMT top right)	WVA1_AV1 (150726.151627.m)	WVA2_AV1 (150726.151628.m)	WVA3_CALCULATED (FRAME FROM WVA1)	WVA4_AV1 (NA)												
1st particle	2029	14:48:58	4831	4715	1024.8		2029	14:48:58	4831	4715	1024.8		2029	14:48:58	4831	4715	1024.8		2029	14:48:58	4831	4715	1024.8							
2nd particle	2030	14:48:58	4831	4715	1024.8		2030	14:48:58	4831	4715	1024.8		2030	14:48:58	4831	4715	1024.8		2030	14:48:58	4831	4715	1024.8							
3rd particle	2031	14:48:59	5044	4970.4974	1090.2		2031	14:48:59	5044	4970.4974	1090.2		2031	14:48:59	5044	4970.4974	1090.2		2031	14:48:59	5044	4970.4974	1090.2							
Compression																														
PO laser current drop	2326	Go find on Console info	18780	18885	3979.4		2326	Go find on Console info	18780	18885	3979.4		2326	Go find on Console info	18780	18885	3979.4		2326	Go find on Console info	18780	18885	3979.4							
V-decay (beam) (mg)	2420	14:52:15	21085	21029	4472.6		2420	14:52:15	21085	21029	4472.6		2420	14:52:15	21085	21029	4472.6		2420	14:52:15	21085	21029	4472.6							
V-decay (total) (mg)	2420	14:52:49	21085	21029	4472.6		2420	14:52:49	21085	21029	4472.6		2420	14:52:49	21085	21029	4472.6		2420	14:52:49	21085	21029	4472.6							
PS (150Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (100Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (50Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (25Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (12.5Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
Detection Minus	2424	14:52:32	21106	21259	4519.5		2424	14:52:32	21106	21259	4519.5		2424	14:52:32	21106	21259	4519.5		2424	14:52:32	21106	21259	4519.5							
Detection Plus	2428	14:52:36	21104	21243	4514.2		2428	14:52:36	21104	21243	4514.2		2428	14:52:36	21104	21243	4514.2		2428	14:52:36	21104	21243	4514.2							
5000 Wave	2428	14:52:36	21094	21092	4509.3		2428	14:52:36	21094	21092	4509.3		2428	14:52:36	21094	21092	4509.3		2428	14:52:36	21094	21092	4509.3							
PO laser current drop	2538	Go find on Console info	26655	26593	5565.2		2538	Go find on Console info	26655	26593	5565.2		2538	Go find on Console info	26655	26593	5565.2		2538	Go find on Console info	26655	26593	5565.2							
V-decay A	2538	14:54:07	27804	27818	5897.8		2538	14:54:07	27804	27818	5897.8		2538	14:54:07	27804	27818	5897.8		2538	14:54:07	27804	27818	5897.8							
V-decay B	2538	14:54:07	27804	27818	5897.8		2538	14:54:07	27804	27818	5897.8		2538	14:54:07	27804	27818	5897.8		2538	14:54:07	27804	27818	5897.8							
PS (150Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (100Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (50Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (25Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (12.5Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
Detection Minus	2614	14:54:43	23189	23142	6191.6		2614	14:54:43	23189	23142	6191.6		2614	14:54:43	23189	23142	6191.6		2614	14:54:43	23189	23142	6191.6							
Detection Plus	2618	14:54:47	23189	23142	6191.6		2618	14:54:47	23189	23142	6191.6		2618	14:54:47	23189	23142	6191.6		2618	14:54:47	23189	23142	6191.6							
5000 Wave	2623	14:54:51	23673	23672	6294.3		2623	14:54:51	23673	23672	6294.3		2623	14:54:51	23673	23672	6294.3		2623	14:54:51	23673	23672	6294.3							
PO laser current drop	2920	Go find on Console info	43785	43749	9387.7		2920	Go find on Console info	43785	43749	9387.7		2920	Go find on Console info	43785	43749	9387.7		2920	Go find on Console info	43785	43749	9387.7							
V-decay A	2920	14:58:09	43785	43749	9387.7		2920	14:58:09	43785	43749	9387.7		2920	14:58:09	43785	43749	9387.7		2920	14:58:09	43785	43749	9387.7							
V-decay B	2920	14:58:09	43785	43749	9387.7		2920	14:58:09	43785	43749	9387.7		2920	14:58:09	43785	43749	9387.7		2920	14:58:09	43785	43749	9387.7							
PS (150Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (100Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (50Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (25Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
PS (12.5Hz)	NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA	NA		NA	NA	NA	NA								
Analysis Notes	Wave formation in initial injection, both reflections - / -																													
	5 small waves during initial injection																													
	Waves to the right side of VWA during 1-L0m, PS = 25Hz?																													
	1500-200 PS 40Hz																													
	Merge the large response to																													
	2750Hz is a large perturbation on 1 particle (move up and split; 2 diam; changing!!) - 2604																													
	(c) 2500 Hz the particles have a circular response to the change																													
	68039 WVA2 PS																													

## **A.2. Campaign 12**

Campaign 12 was proposed before we had finished our full analysis of the injections in campaign 7. We wanted to replicate the injections of campaign 7 and gain as many more injection datasets as possible. We also ran this at a higher framerate for a reduced field-of-view (see Section 2.1.3 and Figure 2-2) to gain more temporal information. The full proposal we submitted to the PK-4 operating team is attached, beginning on the next page.

### **A.2.1. Proposal**

The proposal document we submitted to the PK-4 operating team in April 2021.



## PK-4 ISS Campaign 12 Proposal: Follow-up Experiment on Energy Dissipation

### Team Members

Auburn University- Lori Scott, Edward Thomas, Jr., Uwe Konopka  
Wittenberg University – Jeremiah Williams  
Baylor University – Eva Kostadinova, Truell Hyde, Lorin Matthews

### Scientific Goals

The primary goal of this proposed experiment will be to further investigate the redistribution of the kinetic energy in the dust cloud at the application of polarity switching using: (a) improved temporal resolution through a reduced FOV and corresponding increased camera frame rate and (b) determining if there is increased dissipation due to a larger dust cloud thermal mass. From our Campaign 7 experiment, rapid heating occurs in the flight data that was not observed on the ground experiment (*Figure 1*). Based on these results and supporting simulations, we believe there is a change in the plasma density, and therefore, a change in the Debye length and the structure of the dust cloud at the onset of polarity switching that is giving rise to the observed heating.

Our simulation work is showing promising results close to that of the experiment (*Figure 2*). By having a higher frame rate, we hope to have better temporal resolution on rise and decay of the temperature of the dust cloud. There are also several interesting effects that occur in the MD simulations at the application of polarity switching, and we hope to verify they do appear in the experimental data as well. Additionally, to further test the total energy dissipation process in the cloud, we propose two tests. In the first, we will repeat the experiment but using a larger dust cloud thermal mass (i.e., increasing the number of particles by shakes). In the second, we will repeat the experiment using particles with twice the size (larger mass). With these experiments, we will be able to compare the previous datasets to further investigate the effects of Debye length and charge on the dust cloud at the application of polarity switching.

### Proposed Experiment Description

To follow up on the data collected from our Campaign 7 experiment, we propose an injection-focused experiment that will take one experiment slot for Campaign 12. The parameters are as follows:

Gas: Neon	Current: 0.7 mA
Plasma Mode: DC only	PO Cameras FOV:
Particle size(s): 3.38 $\mu$ m, 6.86 $\mu$ m	Standard- 1600 x 480, 70 fps
Polarity Switching frequency: 500 Hz	Reduced- 1600 x 120, ~140 fps
Pressure: 0.6 mBar	

Injection 1: This will be our baseline injection. We want the same number of shakes (3) we previously used in Campaign 7. To start, we will use a standard capture using polarity switching and standard FOV. After the application of polarity switching, hold for 15 seconds to allow for energy dissipation. Y-scan the cloud, to determine dust cloud density. Change the camera settings to reduced FOV/higher framerate. Reinject the cloud and hold for 15s. Repeat- reinject the cloud and hold for 15s. Flush.

Injection 2: This will be a larger cloud with 6 shakes. Repeat the procedure for injection 1.

Injection 3: Repeat injection 1 with the 6.86  $\mu\text{m}$  particles and 3 shakes.

Injection 4: Repeat injection 2 with the 6.86  $\mu\text{m}$  particles and 6 shakes.

### Tables and Figures

Table 1: Proposed Procedure Table

	Injection 1			Injection 2			Injection 3			Injection 4		
P [mBar]	0.6			0.6			0.6			0.6		
I [mA]	0.7			0.7			0.7			0.7		
Particle Size [ $\mu\text{m}$ ]	3.38			3.38			6.86			6.86		
# shakes	3			6			3			6		
dataset	1	2	3	4	5	6	7	8	9	10	11	12
FOV	std.	reduced	reduced	std.	reduced	reduced	std.	reduced	reduced	std.	reduced	reduced
Collection [s]	45	15	15	45	15	15	45	15	15	45	15	15
Technical [s]	210			210			210			210		
Total	285			285			285			285		

Total time ~20 minutes

\*Technical time for each injection assumes 20 s for dust particle transport, 20 s for PO cameras/laser adjustment, 5 s for cloud resetting (moving in and out of the FoV), and 40 s for flushing the cloud. Setup time and shutdown time (~240 s each) are not included in this table.

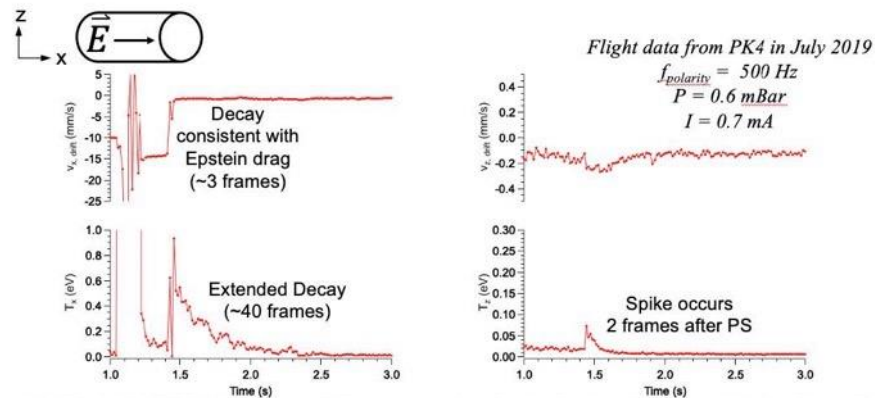


Figure 1: Whole Field Drift Velocity and Temperature Graphs obtained in Campaign 7. Polarity switching occurs when the x-drift velocity drops to 0 (~1.5s). There is an initial spike in the x-direction temperature, and it takes ~40 frames to decay. This temperature spike is not seen on ground-based experiments.

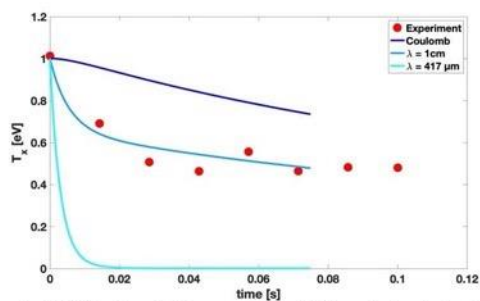



Figure 2: Result from scaled MD simulations using different dust-dust interaction forces of the temperature decay in the particle cloud after the application of polarity switching compared to the data shown in Figure 1 (shifted to have the maximum  $T_x$  of each dataset to occur at  $t = 0$  s). The red dots indicate measurements at 70 fps from the experiment. The higher frame rate measurements would allow for a more accurate temporal resolution of the temperature rise and subsequent decay rate of the energy dissipation in the dust cloud around the application of polarity switching.

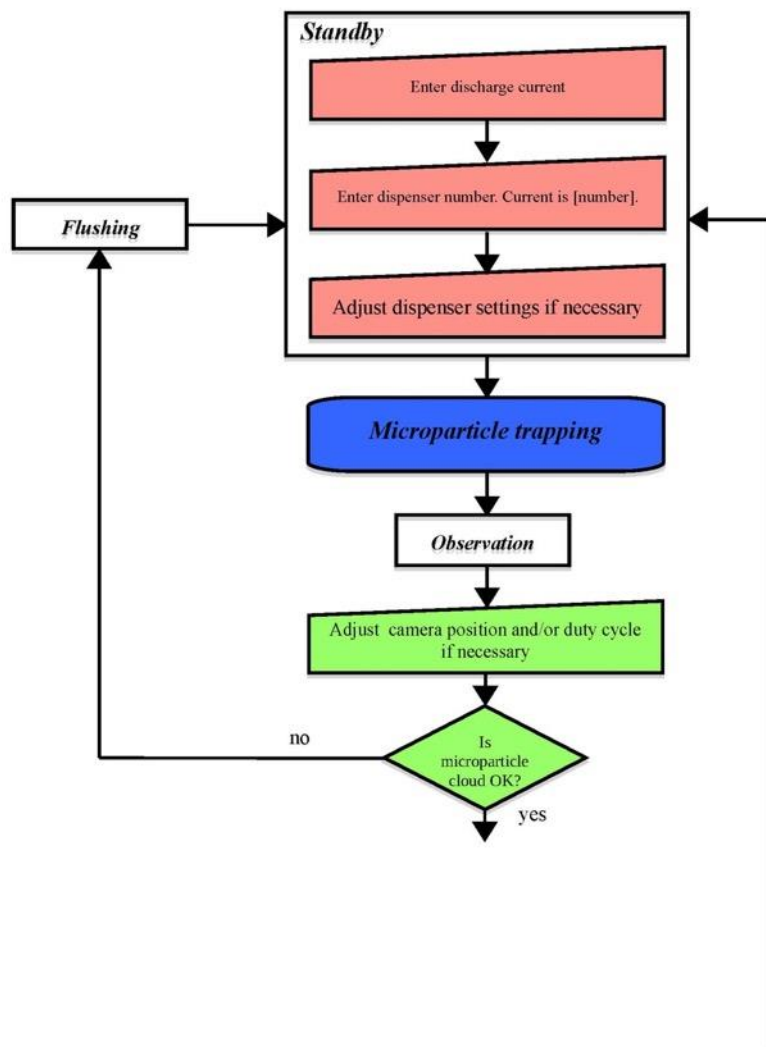
## A.2.2. Flowchart

The flowchart for the code that ran our experiment for campaign 12.

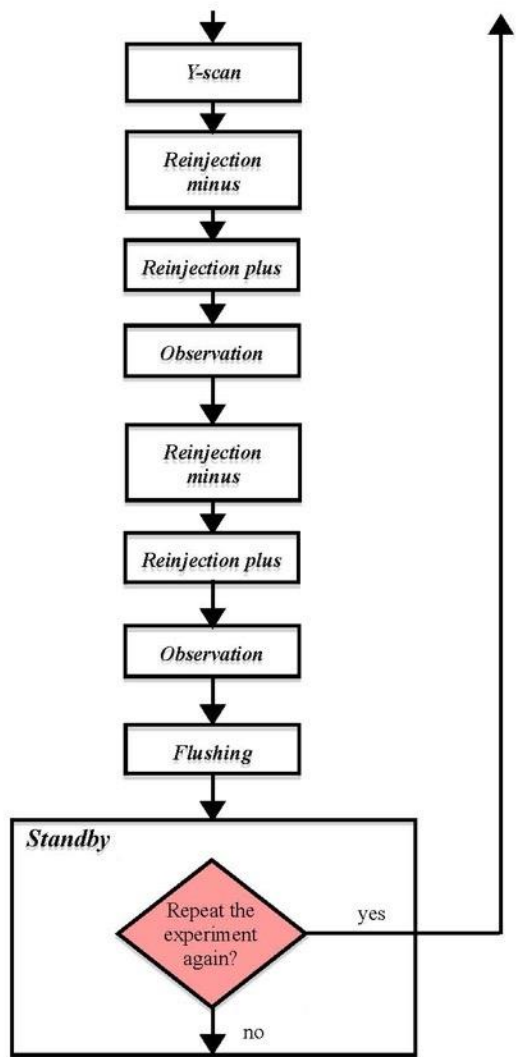
	<b>Energy transport after microparticle trapping</b>					Operational information				
	Script name	EnergyTransport_C12 .epl					Script executions	1		
Keywords: Charging, chains						Successful trappings	4			
Investigation of energy transfer when stopping a drifting microparticle cloud. 04/30/21 11:29:06 AM						Gas	neon			
						Crew activity	Microparticle trapping			
Module	Power	HV	RF	TMTA	PO	OM	DCGC	VC	Pos	
Usage	x	x		x	x		x	x	x	

Experiment team		
Name	Institution	E-mail address
Mikhail Pustylnik	DLR	mikhail.pustylnik@dlr.de
Lori Scott	Auburn University	lcs0044@tigermail.auburn.edu
Edward Thomas	Auburn University	etjr@auburn.edu
Uwe Konopka	Auburn University	uzk0003@auburn.edu
Jeremiah Williams	Wittenberg University	jwilliams@wittenberg.edu
Truell Hyde	Baylor University	Truell_Hyde@baylor.edu
Eva Kostadinova	Baylor University	eva_kostadinova@baylor.edu
Lorin Matthews	Baylor University	lorin_matthews@baylor.edu
Marlene Rosenberg	UCSD	rosenber@ece.ucsd.edu
Peter Hartmann	Hungarian Academy of Sciences	hartmann.peter@wigner.mta.hu
Andrey Lipaev	JlHT RAS	lipaev@mail.ru

	<b>Energy transport after microparticle trapping</b>	Flowchart
---	--	-----------




2 | Energy transport after microparticle trapping



Parameters				
	Constant	Variable within experimental run	Feedback from HCI variation in pauses	Variable between experimental runs
ExpIif		<p><b>Standby, Y-scan, Microparticle trapping, Observation, Reinjection plus, Reinjection minus:</b> Interface valve open Bypass valve closed</p> <p><b>Flushing:</b> Two cycles of Interface and Bypass valves to increase the output pressure</p>		
HV	<p>EM electrode not used</p> <p>Current entered in Pause 1 (pCurrent)</p>	<p><b>Microparticle trapping, Y-scan, Observation:</b> Active electrode: pulse generator, polarity switching with 500 Hz frequency, pCurrent mA current, residual-drift-corrected 50% duty cycle</p> <p><b>Reinjection minus</b> dc discharge with -pCurrent mA current</p> <p><b>Reinjection plus</b> dc discharge with pCurrent mA current</p>	<p><b>Microparticle trapping:</b> offset (psOffset): 1/2</p>	
TMTA	Not parameterized			
PO	<p>PO cameras: gain 0 black level 600 FoV: 1600x120 central cutout frame rate 140 fps Exposure time 7 ms</p>	<p><b>Standby:</b> Illumination laser current 0.75 A</p> <p><b>Microparticle trapping, Reinjection minus, Reinjection plus:</b> Illumination laser current 1.1 A</p>		



	<p>PO camera 1 on full screen</p> <p>PGO cameras: full FoV gain 512 frame rate 15 fps exposure time 30 ms</p> <p>Spectrum readout interval 4 s</p> <p>Spectrometer integration time 750 ms</p>	<p><b>Observation, Y-scan, Flushing:</b> Illumination laser current dispenser-dependent: 0.7 A – for dispenser 6 0.9 A – for dispenser 5 1.1 A – for dispenser 4</p>		
DCGC	<p>Dispenser 5 T_off 100 ms T_on 30 ms Shakes 3</p>	<p><b>Flushing:</b> IQF200 valve open IQP600 valve closed Gas reservoir valve closed</p> <p><b>Standby, Microparticle trapping, Observation, Reinjection plus, Reinjection minus:</b> IQF200 valve closed IQP600 valve open Gas reservoir valve open</p>		
VC	<p>Pressure 0.6 mbar</p>			
Pos	<p>X-axis : 100 mm Illu-laser axis : 115 mm RF axis : 0 OM-laser axis : 0</p>	<p><b>Standby, Microparticle trapping, Observation, Reinjection plus, Reinjection minus, Flushing:</b> Y-axis : 15 mm</p> <p><b>Y-scan :</b> Y-axis : 10-20 mm velocity 2 mm/s</p>		

		<b>Energy transport after microparticle trapping</b>		Step-by-step procedure
Step number	Description	Execution	Remark	
1.	<b>Begin Standby</b>			
1.1.	Set illumination laser current 0.75 A	Script		
1.2.	Set pressure 0.6 mbar	Script		
1.3.	User action: Enter discharge Current (pCurrent)	Ground	<b>CP 1:</b> Possible LOS break. Range 0-3 mA	
1.4.	User action: Enter dispenser number	Ground	<b>CP 1:</b> Possible LOS break. Range 4-6	
1.5.	User action: Adjust dispenser parameters if necessary	Ground	<b>CP 2:</b> Possible LOS break or crew notification. Continue only if crew ready for the activity.	
2.	<b>End Standby</b>			
3.	Start recording all cameras	Script		
4.	Start spectrometer acquisition	Script		
5.	Wait 10 s	Script		
6.	<b>Microparticle trapping</b>	Crew		
7.	<b>Begin Observation</b>			
7.1.	Set dispenser dependent illumination laser current	Script		

6 | **Energy transport after microparticle trapping**

7.2.	Wait 15 s	Script	
8.	<b>End Observation</b>		
9.	User action: Adjust camera position if necessary.	Ground	CP 3: Fast execution on the scientists' request
10.	User action: Is microparticle cloud OK? - y/n	Ground	CP 4: Fast execution on the scientists' request
10.1.	n: <b>Flushing</b>	Script	
10.2.	n: Wait 10 s	Script	
10.3.	n: Stop spectrometer acquisition	Script	
10.4.	n: Stop recording all cameras	Script	
10.5.	n: <b>Go To 1</b>	Script	
11.	Readout psOffset	Script	
12.	<b>Y-scan</b>	Script	
13.	<b>Begin Reinjection minus</b>		
13.1.	Set illumination laser current 1.1 A	Script	
13.2.	Set dc plasma with -pCurrent mA	Script	
13.3.	Wait 2 s	Script	
14.	<b>End Reinjection minus</b>		

7 | Energy transport after microparticle trapping

15.	<b>Begin Reinjection plus</b>		
15.1.	Set dc plasma with pCurrent mA	Script	
15.2.	Wait 2 s	Script	
16.	<b>End Reinjection plus</b>		
17.	<b>Begin Observation</b>		
17.1.	n: Set polarity-switched plasma with 0.7 mA current, 500 Hz polarity-switching frequency, 50% residual-drift-corrected duty cycle	Script	
17.2.	Set dispenser dependent illumination laser current	Script	
17.3.	Wait 15 s	Script	
18.	<b>End Observation</b>		
19.	<b>Begin Reinjection minus</b>		
19.1.	Set illumination laser current 1.1 A	Script	
19.2.	Set dc plasma with -pCurrent mA	Script	
19.3.	Wait 2 s	Script	
20.	<b>End Reinjection minus</b>		
21.	<b>Begin Reinjection plus</b>		
21.1.	Set dc plasma with pCurrent mA	Script	

21.2.	Wait 2 s	Script	
22.	<b>End Reinjection plus</b>		
23.	<b>Begin Observation</b>		
23.1.	n: Set polarity-switched plasma with 0.7 mA current, 500 Hz polarity-switching frequency, 50% residual-drift-corrected duty cycle	Script	
23.2.	Set dispenser dependent illumination laser current	Script	
23.3.	Wait 15 s	Script	
24.	<b>End Observation</b>		
25.	<b>Flushing</b>	Script	
26.	Wait 10 s	Script	
27.	Stop spectrometer acquisition	Script	
28.	<b>Begin Standby</b>		
29.	Set illumination laser current 0.75 A	Script	
30.	User action: Repeat experiment again - y/n?	Ground	<b>CP 5:</b> Possible LOS break.
31.	y: <b>GoTo 1</b>	Script	
32.	<b>End Standby</b>		

Step number	User action message/ Experiment block	To be filled on console			Date	Start time	Console protocol	Remark
		CADMOS OPS	cons.	crew				
1.	Enter discharge current							Possible LOS break. Range 0-3 mA
2.	Standby Enter dispenser number							Possible LOS break. Range 4-6
3.	Adjust dispenser [number] settings							Possible LOS break or crew notification. Continue only if crew ready for the activity.
								
	<i>Microparticle trapping</i>							Crew activity
	<i>Observation</i>							
4.	Adjust camera position if necessary							Fast execution on the scientists' request
5.	Is microparticle cloud OK?							Fast execution on the scientists' request
	<i>Y-scan</i>							
	<i>Reinjection minus</i>							Fast execution on the scientists' request
	<i>Reinjection plus</i>							
	<i>Observation</i>							
	<i>Reinjection minus</i>							
	<i>Reinjection plus</i>							
	<i>Observation</i>							
	<i>Flushing</i>							

10 | Measurements of charging, energy, and microparticle chains

6.	 Repeat the experiment again?	Possible LOS break.
----	--	---------------------

11 | Measurements of charging, energy, and microparticle chains

### **A.3. Other Campaign Descriptions**

While this dissertation discusses one campaign's data in totality (campaign 7) and mentions data from another (campaign 12), I have performed 5 experiments in 4 campaigns, and we have another campaign accepted with 1-2 slots scheduled for later this year (delayed due to technical difficulties and politics due to Russia's invasion of Ukraine). This section is to document the scientific questions for each of these projects for future investigators.

#### **A.3.1. Campaign 9**

Campaign 9 was submitted in November 2019, ground testing was performed in December 2019 at the Facility Science Team (FST) meeting, and successfully completed in France in February 2020 (I flew back to the US 2 days before Auburn issued the international travel recall due to COVID19 pandemic). This was done in collaboration with Baylor and Wittenberg Universities. We used 2 pressures (0.2 and 0.4 mBar) to look at more captures and string formation to replicate campaign 7. We then used the manipulation laser at varying, but predetermined, values to manipulate the particles in the middle of the cloud into a flowing structure. This was designed to look at the opposite effect of polarity switching- to go from a "capture" to a "flowing" cloud. Some initial results from the BU PK-4 experiment show that this does create a change in the dust cloud temperature, seen in section 5.2.3, but a full analysis has not been performed. The data we downlinked was from an injection, not a manipulation section, to replicate the data for this dissertation. The proposal and flowchart are attached, beginning on the next page.



### PK-4 ISS C9 Proposal: Energy Transport in Multi-Chain Dusty Plasma.

**Motivation.** The dynamical state of a driven dissipative system, such as a dusty plasma, is guided by a subtle balance between externally gained energy and frictional loss to the environment. Energy transport processes within dusty plasmas can lead to the formation of vortices (self-excited [1]–[3] or externally-driven [4], [5]), shear instabilities [6], [7], Kolmogorov flows [8], and wave turbulence [9]–[11]. This study investigates energy transport processes in multi-chain dusty plasmas (Fig. 1a), where self-excited dynamics coexist with externally driven perturbations. Two processes are of specific interest: i. quasi-random agitation, and ii. shear flow interface interactions. The proposed experiments will require two particle injections and will make use of eight experimental “sets” [S1-S8] with a range of pressures and plasma currents as summarized in Table I, below.

To investigate how energy is stored and dissipated through the multi-chain structure, a quasi-random agitation using a particle manipulator will be examined. Ground-based experiments have previously shown that the dust thermal energy  $\sim n_d T_d$  is typically much greater than  $n_d T_n$  ( $n_d$ -dust density,  $T_d$ -dust temperature,  $T_n$ -neutral gas temperature) [12]–[14]. These studies have also demonstrated that the dust velocity distribution can deviate from a Maxwellian, which has important implications for the analysis of dust-plasma interactions and the thermodynamics of the driven dissipative system. This phenomenon will be explored in experimental sets [S1, S2, S3, S5, S6 and S7], where the cloud will be excited either by optical manipulation (OM) laser pulses or a generic “randomization” block (provided by the DLR team). (Please see Appendix A for a discussion of the proposed OM excitation.) Preliminary studies from the BU-PK4 facility are shown in Figure 2 where the width of the dust velocity distribution increases whenever the OM laser is active, modulated using a square wave.

Multi-chain dusty plasmas have been observed to exhibit self-excited dust density waves both in the PK-4 ISS (Fig 1b) and using ground experiments [15], [16]. Data from the PK-4 ISS (C1-4) indicates that string-to-wave transitions can occur in the pressure regime,  $p \lesssim 0.20$  mBar. However, the manner in which energy transport is affected by the interaction between such self-excited phenomena and induced shear flows has not yet been determined. Since it has been previously theorized that shear flows can be wholly or partially “blocked” at the interface with the vortical region, leading to damping of the fluctuations beyond a certain ‘sheltering distance’, this process is of interest across multiple physics areas [17]. Examination of both PK-4 ISS and PK-4 BU data indicates that laser-induced shear flows exhibit an unexpectedly localized effect in perturbing the cloud outside the area directly affected by the laser radius (see Fig. 1c). In experimental sets S4 and S8, the manner in which self-excited dynamics within the cloud might shield energy transport from the OM laser through interfacial interactions will be examined. These experiments will also examine the role of energy dissipation due to dust collisions, which can result from thermophoretic and photophoretic forces [18], [19].

**Experiments.** Table I summarizes the relevant parameters for the proposed experiments. All requested experimental sets will require: a pure DC neon plasma, dust particles of  $3.38 \mu\text{m}$  diameter (dispenser 5), and a camera frame rate of 70 fps. In each injection, the dust particles will be transported by a unidirectional DC current of 1 mA and trapped in the FoV using polarity switching (500 Hz, 50% duty cycle<sup>1</sup>). Experimental sets within the same injection will be separated by a resetting of the dust cloud (5 s duration where the cloud is moved in and out of the FoV without flushing, reinjection<sup>+</sup> and reinjection<sup>-</sup>)<sup>2</sup>. The goal of this *re-injection* is to assist the thermalization of the cloud between successive manipulations.

- **Injection 1 (0.2 mBar):** During S1, once the cloud settles in the FoV, a quasi-random agitation of the cloud will be performed employing either a fixed sequence of OM laser pulses with frequency  $\sim 10$  Hz (duration  $\sim 0.1$  s) and an amplitude in the range 6-8 mA or a generic “randomization” block (provided by the DLR team). The goal of this procedure is to simulate short modulated excitations of the cloud,

<sup>1</sup> The duty cycle may be adjusted as needed to ensure the dust cloud is visible within the FoV.

<sup>2</sup> For simplicity, throughout this document, we call the resetting of the cloud a *re-injection*.

which supply external energy to the system without generating substantial shear flow. After this perturbation is complete, the cloud will be observed for 100 s, followed by reinjection. The same procedure will be performed during S2 and S3 but with the PO cameras located at position  $y = 16.4$  mm (0.2 mm off-OM laser center) and  $y = 16.6$  mm (0.4 mm off-OM laser center), respectively, as opposed to  $y = 16.2$  (OM laser center) for S1. These three data sets will allow for observation of the heat transport throughout the cloud volume. During S4, after the cloud is re-injected and settles in the FoV, a continuous OM perturbation (laser current 10 A) will be performed for 10s. The goal of this manipulation is to create a shear flow within the bulk of the cloud in order to investigate dust dynamics at the shear interface. Once the perturbation is complete, the cloud will be observed for 60 s. Throughout S4, the PO cameras should be centered at the OM laser center position, i.e., 16.2 mm.

- **Injection 2 (0.4 mBar):** For S5-S8, the experimental procedure performed in Injection 1 will be repeated at pressure of 0.4 mBar. In other words, [S5-S7] follow the same procedure as [S1-S3] and [S8] follows the same procedure as [S4]. Apart from different pressure values, the only other difference in parameters during Injection 2 is an increased OM laser current (20 A in S8, as opposed to 10A in S4) during the shear flow perturbation.

### Table and Figures

Table I. Proposed conditions for the quasi-random shear experiments.

	Injection 1				$\Delta p$	Injection 2			
<b>P [mBar]</b>	0.2					0.4			
<b>8 datasets</b>	S1	S2	S3	S4		S5	S6	S7	S8
<b>PO y-position [mm]</b> <sup>1</sup>	16.2	16.4	16.6	16.2		16.2	16.4	16.6	16.2
<b>I<sub>DC</sub> [mA]</b>	1	1	1	1		1	1	1	1
<b>I<sub>OM</sub> [A]</b>	6-8	6-8	6-8	10		6-8	6-8	6-8	20
<b>Collection [s]</b> <sup>2</sup>	120 + 120 + 120 + 70 = 430					120 + 120 + 120 + 70 = 430			
<b>Technical [s]</b> <sup>3</sup>	210				60	210			
<b>Total [s]</b>	640				60	640			
<b># of shakes/injection</b>	1-3					1-3			

<sup>1</sup> The reference position of the PO cameras y-axis position should coincide with the center of the OM laser axis:  $y = 16.2$  mm.

<sup>2</sup> The collection time for S1-S3 and S5-S7 may vary based on the selected method for random agitation.

<sup>3</sup> Technical time for each injection assumes 20 s for dust particle transport, 20 s for PO cameras/laser adjustment, 5 s for cloud resetting (moving in and out of the FoV), and 40 s for flushing the cloud. Setup time and shutdown time (~240 s each) *are not* included in this table.

**The total technical, collection, setup, and shutdown time adds up to  $\approx 30$  min.**

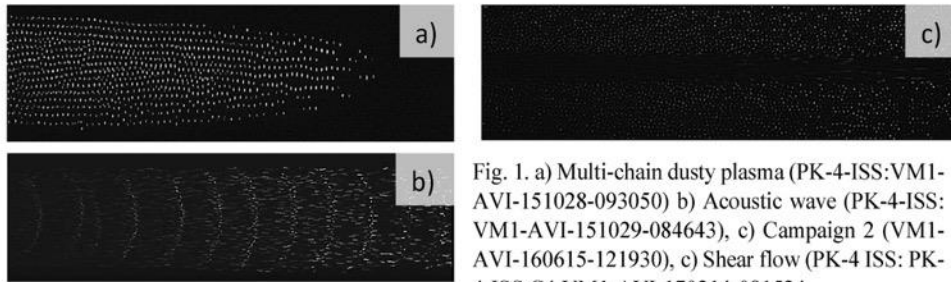


Fig. 1. a) Multi-chain dusty plasma (PK-4-ISS:VM1-AVI-151028-093050) b) Acoustic wave (PK-4-ISS:VM1-AVI-151029-084643), c) Campaign 2 (VM1-AVI-160615-121930), c) Shear flow (PK-4 ISS: PK-4-ISS C4 VM1-AVI-170214-081524).



Version 2

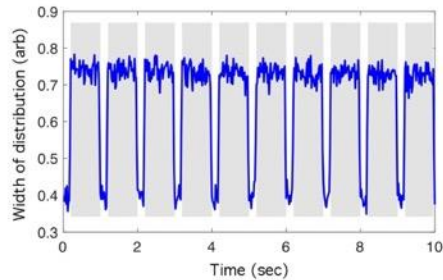


Fig. 2: Width of the velocity-space distribution in the PK4-BU ground experiment during driven modulation from OM laser. The plot shows the width of the velocity distribution during 80% duty cycle variable amplitude, square wave modulation of the OM laser (80% "on" region shown in gray). The particle velocities are obtained using a PIV technique and a velocity distribution function is constructed. The width of the velocity distribution is shown to increase during the laser "on" phase.

### Bibliography

- [1] M. Schwabe, S. Zhdanov, C. R ath, D. B. Graves, H. M. Thomas, and G. E. Morfill, "Collective Effects in Vortex Movements in Complex Plasmas," *Phys. Rev. Lett.*, vol. 112, no. 11, p. 115002, Mar. 2014.
- [2] G. E. Morfill *et al.*, "Highly Resolved Fluid Flows: ``Liquid Plasmas'' at the Kinetic Level," *Phys. Rev. Lett.*, vol. 92, no. 17, p. 175004, Apr. 2004.
- [3] M. Rubin-Zuzic, H. M. Thomas, S. K. Zhdanov, and G. E. Morfill, "Circulation' dynamo in complex plasma," *New J. Phys.*, vol. 9, no. 2, p. 39, 2007.
- [4] M. Klindworth, A. Melzer, A. Piel, and V. A. Schweigert, "Laser-excited intershell rotation of finite Coulomb clusters in a dusty plasma," *Phys. Rev. B*, vol. 61, no. 12, pp. 8404–8410, Mar. 2000.
- [5] G. Uchida, S. Iizuka, T. Kamimura, and N. Sato, "Generation of two-dimensional dust vortex flows in a direct current discharge plasma," *Physics of Plasmas*, vol. 16, no. 5, p. 053707, May 2009.
- [6] V. Nosenko and J. Goree, "Shear Flows and Shear Viscosity in a Two-Dimensional Yukawa System (Dusty Plasma)," *Phys. Rev. Lett.*, vol. 93, no. 15, p. 155004, Oct. 2004.
- [7] R. Heidemann, S. Zhdanov, K. R. S utterlin, H. M. Thomas, and G. E. Morfill, "Shear flow instability at the interface among two streams of a highly dissipative complex plasma," *EPL*, vol. 96, no. 1, p. 15001, 2011.
- [8] A. Gupta, R. Ganesh, and A. Joy, "Kolmogorov flow in two dimensional strongly coupled dusty plasma," *Physics of Plasmas*, vol. 21, no. 7, p. 073707, Jul. 2014.
- [9] S. Zhdanov, M. Schwabe, C. R ath, H. M. Thomas, and G. E. Morfill, "Wave turbulence observed in an auto-oscillating complex (dusty) plasma," *EPL*, vol. 110, no. 3, p. 35001, 2015.
- [10] Y.-Y. Tsai, M.-C. Chang, and L. I., "Observation of multifractal intermittent dust-acoustic-wave turbulence," *Phys. Rev. E*, vol. 86, no. 4, p. 045402, Oct. 2012.
- [11] J. Pramanik, B. M. Veerasha, G. Prasad, A. Sen, and P. K. Kaw, "Experimental observation of dust-acoustic wave turbulence," *Physics Letters A*, vol. 312, no. 1, pp. 84–90, Jun. 2003.
- [12] R. Fisher, K. Avinash, E. Thomas, R. Merlino, and V. Gupta, "Thermal energy density of dust in dusty plasmas: Experiment and theory," *Phys. Rev. E*, vol. 88, no. 3, p. 031101, Sep. 2013.
- [13] R. Fisher and E. Thomas, "Quantitative comparison of the isotropic and anisotropic Maxwellian velocity space distribution function models in a dusty plasma," *Phys. Rev. E*, vol. 86, no. 6, p. 066403, Dec. 2012.
- [14] R. Fisher and E. Thomas, "Observation and model of an ellipsoidally symmetric velocity space distribution in a weakly-coupled dusty plasma," *Physics of Plasmas*, vol. 18, no. 11, p. 113701, Nov. 2011.
- [15] S. Ratynskaia *et al.*, "Experimental Determination of Dust-Particle Charge in a Discharge Plasma at Elevated Pressures," *Phys. Rev. Lett.*, vol. 93, no. 8, p. 085001, Aug. 2004.
- [16] S. A. Khrapak *et al.*, "Particle charge in the bulk of gas discharges," *Phys. Rev. E*, vol. 72, no. 1, p. 016406, Jul. 2005.
- [17] J. C. R. Hunt and P. A. Durbin, "Perturbed vortical layers and shear sheltering," *Fluid Dynamics Research*, vol. 24, no. 6, pp. 375–404, Jun. 1999.

Version 2

- [18] L. S. Matthews, J. B. Kimery, G. Wurm, C. de Beule, M. Kuepper, and T. W. Hyde, "Photophoretic force on aggregate grains," *Mon Not R Astron Soc*, vol. 455, no. 3, pp. 2582–2591, Jan. 2016.
- [19] M. Küpper, C. de Beule, G. Wurm, L. S. Matthews, J. B. Kimery, and T. W. Hyde, "Photophoresis on polydisperse basalt microparticles under microgravity," *Journal of Aerosol Science*, vol. 76, pp. 126–137, Oct. 2014.

## Appendix A: Quasi-random agitation

### 1. OM Laser excitation in PK-4 BU

Figure 3 provides an image of the random signal created at Baylor and employed in the PK-4 BU to induce a modulated laser agitation in PK-4 BU experiments. This 1 second waveform was created in MatLab by adding a series of square waves with the parameters of the waveform limited in amplitude due to the response of the laser. Once the waveform was created (there are 10000 sample points), the data was injected to a HP signal generator and this signal generator was connected to the laser. The waveform was set at 1 Hz with an amplitude of 1 Volt to allow a one to one correlation between the resulting signal amplitude and the signal generator amplitude.

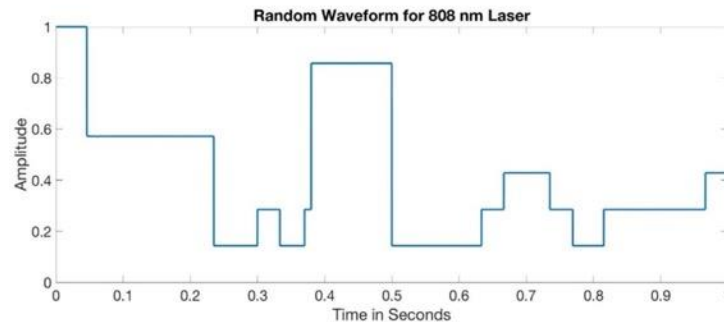



Fig. 3. One-second modulated laser waveform used as a quasi-random agitation in PK-4 BU ground experiments.

We would like to recreate a similar OM perturbation for the proposed PK-4 ISS experiments, if possible. The goal of this perturbation is to supply energy to the cloud without creating substantial shear flow. For simplicity, we propose fixing the timestep for the laser amplitude variation. If possible, this would allow agitation of the cloud with a quasi-random laser strength in the range 6 A – 8 A, changing the value every 0.1 s (10 Hz frequency). (This timescale is on the order of the expected dust response.)

If an OM agitation of this kind is not recommended by the DLR team, we would like to conduct the experiment described using generic "randomization" block in place at DLR.

### 2. "Randomization" block

Can you give us more information on this procedure: which manipulator is used, what parameters for the manipulator, and what is the expected duration of the perturbation?

	<b>Energy Transport in Multi-Chain Dusty Plasma</b>					Operational information				
	Script name	EnergyTransport_C09 .epl					Script executions	1		
Keywords: heat propagation, strings  Random agitation of a string fluid by a manipulation laser and subsequent observation of heat propagation.  13/12/2019 03:48:03 PM						Successful trappings per execution	2			
						Gas	neon			
						Crew activity	Microparticle trapping			
Module	Power	HV	RF	TMTA	PO	OM	DCGC	VC	Pos	
Usage	x	x		x	x	x	x	x	x	

Experiment team		
Name	Institution	E-mail address
Mikhail Pustynik	DLR	mikhail.pustynik@dlr.de
Lori Scott	Auburn University	lcs0044@tigermail.auburn.edu
Edward Thomas	Auburn University	etjr@auburn.edu
Uwe Konopka	Auburn University	uzk0003@auburn.edu
Jeremiah Williams	Wittenberg University	jwilliams@wittenberg.edu
Truell Hyde	Baylor University	Truell_Hyde@baylor.edu

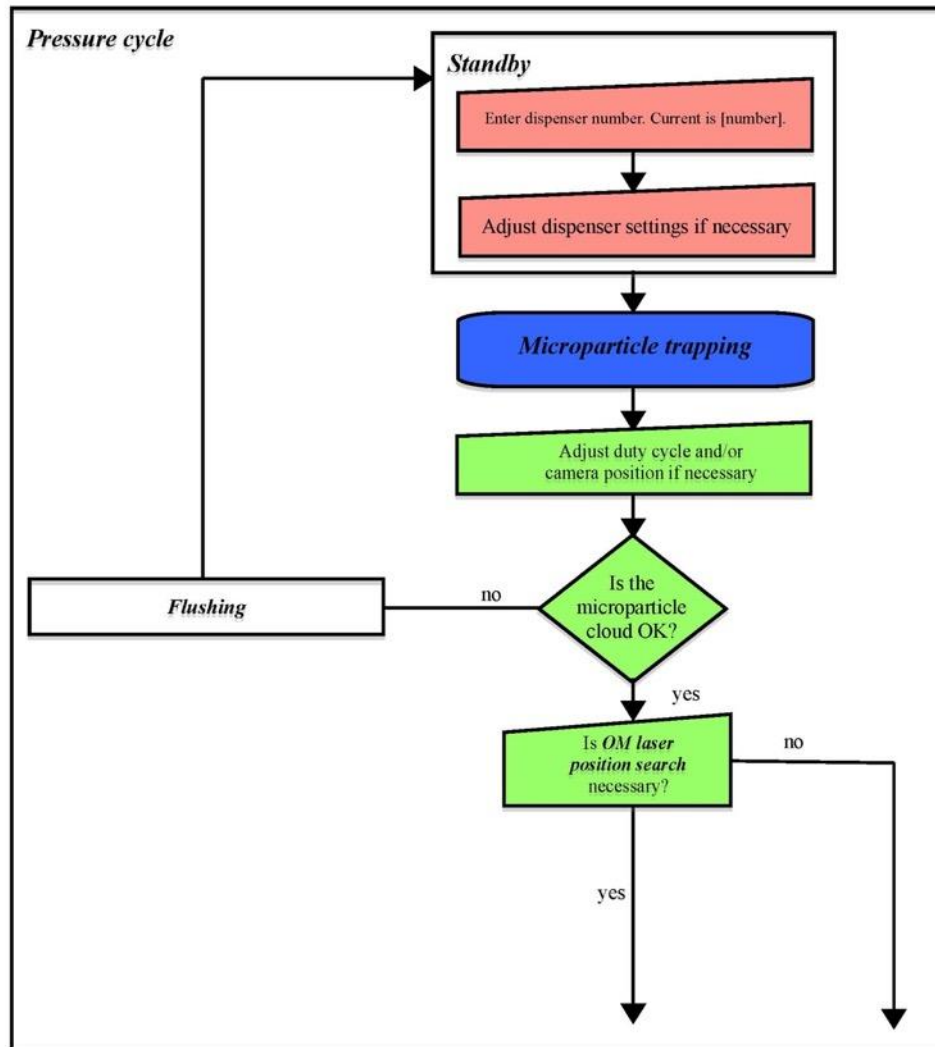
1 | Energy Transport in Multi-Chain Dusty Plasma

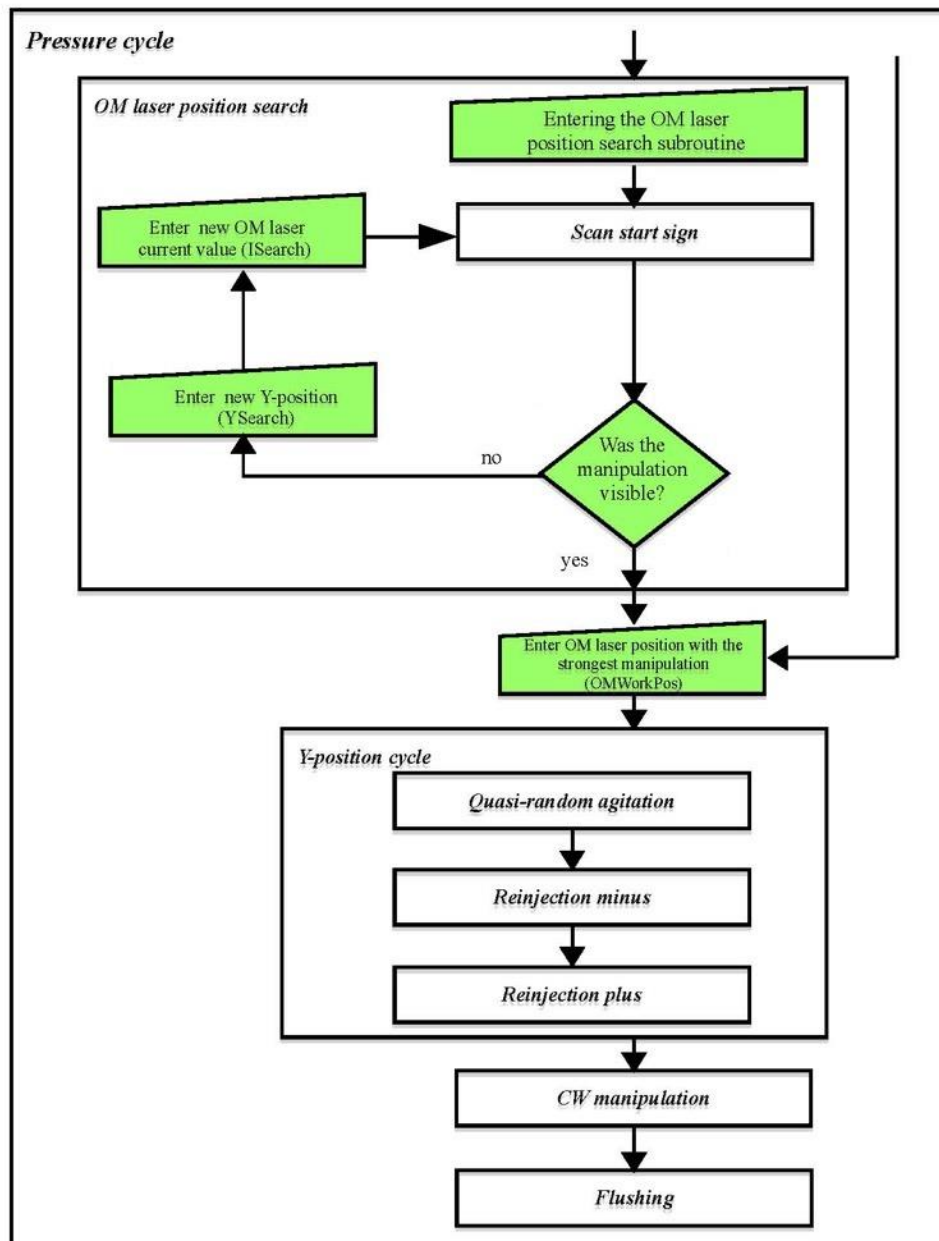
Eva Kostadinova	Baylor University	eva_kostadinova@baylor.edu
Lorin Matthews	Baylor University	lorin_matthews@baylor.edu
Marlene Rosenberg	UCSD	rosenber@ece.ucsd.edu
Peter Hartmann	Hungarian Academy of Sciences	hartmann.peter@wigner.mta.hu
Andrey Lipaev	JIHT RAS	lipaev@mail.ru



## Energy Transport in Multi-Chain Dusty Plasma

Flowchart








Parameters				
	Constant	Variable within experimental run	Feedback from HCl variation in pauses	Variable between experimental runs
Explf		<p><b>Standby, Microparticle trapping, OM laser position search, Y-position cycle, CW Manipulation:</b> Interface valve open Bypass valve closed</p> <p><b>Flushing:</b> Two cycles of Interface and Bypass valves to increase the output pressure</p>		<p><b>Pressure cycle:</b> <b>Standby, Microparticle trapping, OM laser position search, Y-position cycle:</b>  Gas flow [sccm]: pressure-dependent, 0.5</p>
HV	EM electrode off	<p><b>Standby, Plasma off:</b> Active electrode off</p> <p><b>Microparticle trapping, OM laser position search, Quasi-random agitation, CW Manipulation:</b> Active electrode: pulse generator, polarity switching with 500 Hz frequency, 1 mA current, residual-drift-corrected 50% duty cycle</p> <p><b>Reinjection minus</b> dc plasma with -1 mA current</p> <p><b>Reinjection plus</b> dc plasma with 1 mA current</p>	<p><b>Microparticle trapping:</b> offset (psOffset): 1/2</p>	<p><b>Pressure cycle:</b> <b>Reinjection:</b> Duration of dc discharge [s]: 0.5, 1.0</p>
TMTA	Not parameterized			

PO	<p>PO cameras: FoV: 1600x480 central cutout gain 0 black level 600 frame rate 70 fps exposure time 14 PO camera 1 on full screen ms</p> <p>PGO cameras: full FoV gain 512 frame rate 25 fps exposure time 30 ms</p> <p>Spectrometer off</p>	<p><b>Standby:</b> Illumination laser current 0.75 A</p> <p><b>Microparticle trapping, OM laser position search:</b> Illumination laser current 0.9 A</p> <p><b>Quasi-random agitation, CW manipulation:</b> Illumination laser current dispenser-dependent: 0.45 A – for dispenser 6 0.6 A – for dispenser 5 0.9 A – for dispenser 4</p>		
OM		<p><b>Quasi-random agitation:</b> Laser current values [A]: 8.000, 7.143, 6.286, 6.571, 7.023, 7.714, 6.153, 6.571, 6.987, 6.571, 6.153, 7.468, 7.832, 6.342, 6.763, 7.591, 6.194, 6.891, 7.412, 0</p> <p>This set should be repeated 5 times</p>	<p><b>OM laser position search:</b> constant diode current (ISearch): 12 A</p>	<p><b>Pressure cycle: CW manipulation:</b> Constant OM laser power [A]: 10, 20</p>
DCGC	<p>Dispenser number and settings set in Pauses 2 and 3 respectively</p>	<p><b>Microparticle trapping, OM laser position search, Quasi-random agitation, CW manipulation:</b> IQF200 valve closed IQP600 valve open Gas reservoir valve open</p>		

		<p><b>Flushing:</b> IQF200 valve open Gas reservoir valve closed</p> <p><b>Flushing:</b> IQP600 valve closed</p>		
VC				<p><b>Pressure cycle:</b> Pressure [mbar]: 0.2, 0.4</p>
Pos	RF axis : 0	<p><b>Standby, Microparticle trapping :</b> X-axis : 100 mm</p> <p>Nozzle : X-axis : 10 mm Illu-laser axis : 25 mm</p> <p><b>Microparticle trapping:</b> Y-axis: 15 mm</p> <p><b>OM laser position search</b> OM-Laser axis: motion from 0 to 4.8, velocity 0.1 mm/s</p>	<p><b>OM laser position search, Reinjecton plus, Reinjecton minus, CW manipulation:</b> Y-axis (YSearch): 16.2 mm (Illu-laser positioned accordingly)</p> <p><b>Y-position cycle:</b> Y position [mm]: YSearch, YSearch+0.2, YSearch+0.4</p> <p><b>Y-position cycle, CW manipulation:</b> OM laser position (omWorkPos) 2.7 mm</p>	

**General comment to the step-by-step procedure:** From this campaign on, after each green CP, the illumination laser position will be adapted to the X-Y camera position

		<b>Energy Transport in Multi-Chain Dusty Plasma</b>		Step-by-step procedure
Step number	Description	Execution	Remark	
1.	<b>Begin <i>Pressure cycle</i></b>			
1.1.	<b>Begin <i>Standby</i></b>			
1.1.1.	Set gas flow 0.5 sccm	Script		
1.1.2.	Set current pressure	Script	Use setLowPressure	
1.1.3.	User action: Enter dispenser number	Ground	<b>CP 1:</b> Possible LOS break. Range 4-6	
1.1.4.	User action: Adjust dispenser parameters if necessary	Ground	<b>CP 2:</b> Possible LOS break or crew notification. Continue only if crew ready for the activity.	
1.2.	<b>End <i>Standby</i></b>			
1.3.	Start recording all cameras	Script		
1.4.	Wait 10 s	Script		
1.5.	<b><i>Microparticle trapping</i></b>	Crew	Use asymmetry given by psOffset	
1.6.	User action: Adjust duty cycle and/or camera position if necessary	Ground	<b>CP 3:</b> Fast execution on the scientists' request	

1.7.	Readout psOffset	Script	Residual drift compensation basis: recorded setting = 50% duty cycle
1.8.	User action: Is microparticle cloud OK? - y/n	Ground	<b>CP 4:</b> Fast execution on the scientists' request
1.8.1.	n: <b>Flushing</b>	Script	
1.8.2.	n: Wait 10 s	Script	
1.8.3.	n: Stop recording all cameras	Script	
1.8.4.	n: <b>Go To 1.1</b>	Script	
1.9.	Is <b>OM laser position search</b> necessary? - y/n	Ground	<b>CP 5:</b> Fast execution on the scientists' request
1.9.1.	n: <b>Go To 1.12</b>	Script	
1.10.	<b>Begin OM laser position search</b>		
1.10.1.	Entering the <b>OM laser position search</b> subroutine	Ground	<b>OMLPS CP 1:</b> Fast execution on the scientists' request
1.10.2.	Move Y-axis to YSearch	Script	
1.10.3.	<b>Begin Scan start sign</b>		
1.10.3.1.	Set illumination laser current to 0.4 A	Script	
1.10.3.2.	Wait 3 s	Script	
1.10.3.3.	Set illumination laser current to 0.9 A	Script	





1.10.4.	<b>End Scan start sign</b>		
1.10.5.	Switch on OM-laser at position 0, constant diode current ISearch	Script	
1.10.6.	Move the OM-laser axis from 0 to 4.8 mm, velocity 0.1 mm/s	Script	
1.10.7.	Switch off OM laser	Script	
1.10.8.	User action: Was the laser manipulation visible? - y/n	Ground	<b>OMLPS CP 2:</b> Fast execution on the scientists' request
1.10.8.1.	n: User action: Enter new Y-position (YSearch)	Ground	<b>OMLPS CP 3:</b> Fast execution on the scientists' request Range 0-30 mm
1.10.8.2.	n: Readout YSearch	Script	
1.10.8.3.	n: User action: Enter new OM laser current value (ISearch)	Ground	<b>OMLPS CP 4:</b> Fast execution on the scientists' request Range 0-40 A
1.10.8.4.	n: Readout ISearch	Script	
1.10.8.5.	n: <b>Go To</b> <a href="#">1.10.2</a>	Script	
1.11.	<b>End OM laser position search</b>		
1.12.	User action: Enter OM laser position with the strongest manipulation (omWorkPos)	Ground	<b>CP 6:</b> Fast execution on the scientists' request Range 0-4.8 mm



1.13.	<b>Begin Y-position cycle</b>		
1.13.1.	Move Y-axis to current Y position	Script	
1.13.2.	Set dispenser-dependent illumination laser current	Script	
1.13.3.	Switch OM laser on	Script	
1.13.4.	<b>Begin Quasi-random agitation</b>	Script	100 iterations
1.13.4.1.	Set current OM laser current	Script	Use low level command
1.13.4.2.	Wait 0.1 s	Script	
1.13.5.	<b>End Quasi-random agitation</b>	Script	
1.13.6.	Switch off OM laser	Script	
1.13.7.	Wait 100 s	Script	
1.13.8.	Move Y-axis to th position YSearch	Script	
1.13.9.	<b>Begin Reinjection minus</b>		
1.13.9.1.	Set dc plasma with -1 mA current	Script	
1.13.9.2.	Wait current waiting time	Script	
1.13.10.	<b>End Reinjection minus</b>		
1.13.11.	<b>Begin Reinjection plus</b>		
1.13.11.1.	Set dc plasma with 1 mA current	Script	
1.13.11.2.	Wait current waiting time	Script	
1.13.12.	<b>End Reinjection plus</b>		

1.13.13.	Set polarity-switched plasma with 1 mA current, 500 Hz frequency, and residual-drift-corrected 50% duty cycle	Script	Use asymmetry given by psOffset
1.13.14.	Wait 10 s	Script	
1.14.	<b>Repeat Y-position cycle</b>	Script	
1.15.	<b>Begin CW manipulation</b>		
1.15.1.	Move cameras to Y position YSearch	Script	
1.15.2.	Set current constant OM laser current	Script	
1.15.3.	Wait 10 s	Script	
1.15.4.	Turn off OM laser	Script	
1.16.	<b>End CW manipulation</b>		
1.17.	Wait 60 s	Script	
1.18.	<b>Flushing</b>	Script	
1.19.	Wait 10 s	Script	
1.20.	Stop recording all cameras	Script	
2.	<b>Repeat Pressure Cycle</b>	Script	



Step number	User action message/ Experiment block	Pressure 0.2 mbar	Pressure 0.4 mbar	Remark		
 <b>Energy Transport in Multi-Chain Dusty Plasma: main scientific run</b>						
		To be filled on console	CADMOS OPS	cons. crew	Scientist	Console protocol
		Date	Start time			
1.	Enter dispenser number. Current is [number].			Possible LOS break. Range 4-6		
2.	Standby Adjust dispenser [number] settings			Possible LOS break or crew notification. Continue only if crew ready for the activity.		
						
	Microparticle trapping			Crew activity		
3.	Adjust duty cycle and/or camera position if necessary			Fast execution on the scientists' request		
4.	Is microparticle cloud OK?			Fast execution on the scientists' request		
	Flushing					
						
5.	Is OMLPS necessary?			Fast execution on the scientists' request		
	OMLPS			See separate protocol		
6.	Enter OM laser position with the strongest manipulation			Fast execution on the scientists' request. Range 0-4.8		
	Y-Position	Quasi-random agitation				
	cycle	Reinjection minus				
		Reinjection plus				
	CW Manipulation					
	Flushing					
				Possible LOS break.		

Step number	User action message/ Experiment block	To be filled on console			Date	Start time	Console protocol
		CADMOS OPS	oons.	crew			
1.	Entering the <i>OM laser position search</i> routine						Fast execution on the scientists' request
	Scan start sign						
2.	Was the manipulation visible?						Fast execution on the scientists' request
3.	Enter new Y-position (YSearch)						Fast execution on the scientists' request Range 0-30 mm
4.	Enter new OM laser current value (ISearch)						Fast execution on the scientists' request Range 0-40 A

### **A.3.2. Campaign 14**

Campaign 14 was successfully completed from Auburn's campus in February 2022, thanks to the support from NASA's Marshall Space Flight Center in Huntsville, AL. This experiment is designed to look at compression in a dust cloud during both static and dynamic states. We use a combination rf/dc plasma and we were allowed two experiment slots, one for neon and one for argon. This experiment was led by Dr. Konopka, so he has the proposal and script flowcharts.

### **A.3.3. Campaign 15**

Campaign 15 is proposed as a follow-up to our experiments in campaign 14. This time we will focus on just neon plasma but using 3 varying pressures to look at the dependence of the compression properties to varying operating conditions of one gas. This is supposed to be conducted in June 2022 but is delayed due to politics over the experimental apparatus due to the Russian invasion of Ukraine. Dr. Williams has the proposal file, and since this experiment was not performed, there is not a script flowchart.

#### **A.4. Experiment Proposal Process and Timeline**

Typically, proposals are called for the week after a previous PK-4 campaign. Our team writes and submits a proposal, and I was typically the point-of-contact (due to my graduate student scheduling being freer than professors' scheduling). Proposals are reviewed and accepted in about 2 weeks, and then the operating team creates the scripts. We are able to walk through a test of the scripts with the operators (either in person or zoom) and based on our results, they make changes.

The set of experiments for a campaign are then conducted throughout a week on the ISS, with the help of the ground team at CADMOS, cosmonauts, and of course the operating team. Pre-COVID, we traveled to Toulouse, France to operate these experiments. Post-COVID, we registered as NASA interns to be able to receive the direct feeds from the ISS to a professor's office on campus. The on-campus viewing requires the help of COSAM IT, AU OIT, NASA Marshall, and of course, CADMOS, to set up the various ports and permissions. All of these steps are documented and sent to the MPRL professors, but not listed here for publishing purposes.

A small set of data (2-10s, depending on data size availability for the day) can be downlinked that day, and is distributed by the end of the campaign week. Data hard drives are then shipped down with the next set of returning cosmonauts, and after they are distributed internally to CAMDOS and the operating team, we can request a copy of our experiment data. This is put onto a hard drive and shipped to the US from France, we copy it onto our own drives, and ship the original drive back to CADMOS.

## **Appendix B: Analysis Code Files**

This appendix is for all of the code used throughout this work. The real code files can be found in a useable form in the MPRL Auburn Box for future users.

### **B.1. YOAK $\mu$ M Code**

This is the current code and documentation for YOAK $\mu$ M when used for PK-4 replication. A full copy of code can be downloaded from the MPRL (previously PSL, and still named such on github) [github repository](#). I will add each file individually in this appendix, pointing out the important settings in each file needed to run the code.

### B.1.1. Driver.hpp

The header file for the driver.

```

/Users/loriscott/Documents/Yoakum/driver.hpp
Printed: 6/14/22, 5:06:39 PM
Page 1/1
Printed for: Lori Scott
1 // Used for printing output
2 #include <iostream>
3
4 // ODE solver library
5 #include <boost/numeric/odeint.hpp>
6 //Other boost libraries
7 #include <boost/random/normal_distribution.hpp>
8 #include <boost/random/mersenne_twister.hpp>
9 #include <boost/random/uniform_01.hpp>
10 #include <boost/random/uniform_real_distribution.hpp>
11 // IO observers
12 #include "io_observers.hpp"
13
14 // Type for vector values
15 #include "Vector.hpp"
16 // Type for representing dust particle states
17 #include "Dust_State.hpp"
18 // Configuration file
19 #include "configure.hpp"
20 // Represents a system of ODEs
21 #include "System.hpp"
22
23 // Basic forces
24 #include "basic_forces.hpp"
25
26 //Interaction Forces
27 #include "interaction_forces.hpp"
28
```

### **B.1.2. Driver.cpp**

The driver file, is where all important information for the simulation are located. All forces to include in the system of ODES are lines 112-131 (where the most common ones I use are all listed, and I use comments to turn on/off), size and charge of the dust particles are at line 95-97, and running time and timestep size are lines 250-265, depending on which timestep you use (again my frequent use ones are permanently listed and I use comments to change values). If you are importing a previous file as the initial dust state, the filename is located at line 54, and the turn on/off (true/false) setting is at line 213. The file starts on the next page.

```
1 #include "driver.hpp"
2
3 #include <cstdlib>
4 #include <ctime>
5
6 #include <iostream>
7 #include <fstream>
8 #include <sstream>
9 #include <vector>
10 #include <string>
11
12
13 using namespace yoakum;
14 using namespace boost::numeric::odeint;
15 using namespace boost::random;
16
17
18 // Code to read input data
19 bool moveToStartOfLine(std::ifstream& fs);
20 std::string getLastLineInFile(std::ifstream& fs);
21 std::vector<float> readLastLine();
22
23
24 bool moveToStartOfLine(std::ifstream& fs)
25 {
26     fs.seekg(-1, std::ios_base::cur);
27     for(long i = fs.tellg(); i > 0; i--)
28     {
29         if(fs.peek() == '\n')
30         {
31             fs.get();
32             return true;
33         }
34         fs.seekg(i, std::ios_base::beg);
35     }
36     return false;
37 }
38
39 std::string getLastLineInFile(std::ifstream& fs)
40 {
41     // Go to the last character before EOF
42     fs.seekg(-1, std::ios_base::end);
43     if (!moveToStartOfLine(fs))
44         return "";
45
46     std::string lastline = "";
47     getline(fs, lastline);
48     return lastline;
```



```
49 }
50
51
52 std::vector<float> readLastLine()
53 {
54     // Define file to read in
55     // const std::string filename = "blank.dat"; // Use when starting new
... code
56     const std::string filename =
... "P0p6_I0p7_N5000_2D_Y1p75_Injection_ms.dat";
57     // const std::string filename = "Pp06_I07_injection_splitformicro.dat";
58     // const std::string filename =
... "Pp6_Ip7_N1000_ThermBM1_microdt_Yukawa1p75mm_cutoffTxp75_piecewise3a.dat";
59     // const std::string filename =
... "P06_I07_N1000_2D_newHeatingCutoff_NoTherm.dat";
60
61
62     // Create variables that are needed
63     std::string field; // dummy string
64     std::vector<float> lastLine; // array to store values
... from file
65     std::ifstream fs; // input stream
66
67     // open file
68     fs.open(filename.c_str(), std::fstream::in);
69     if(!fs.is_open()) // error if file can't be opened
70     {
71         std::cout << "Could not open file" << std::endl;
72         std::exit(-1);
73         // return -1;
74     }
75
76     // read in last line of file
77     std::stringstream ss(getLastLineInFile(fs));
78
79     // Store in an array using commas as delimiting field
80     while (getline(ss,field,','))
81     {
82         lastLine.push_back(std::stof(field)); // add each field to the 1D
... array
83     }
84
85
86
87     return lastLine;
88 }
89
90
```

```
91 int main( int argc , char **argv )
92 {
93
94 //*****
95 *
96 // DUST INFORMATION
97 float radius = 3.38e-6 / 2; // m
98 float density = 1502; // kg/m^3
99 float num_electrons = 7000 * 1 ; //6680 ~OML Charge
100
101 float single_mass = (4.0 / 3.0) * 3.14 * density * radius * radius *
102 radius; // kg
103 // float single_mass = 1; //kg
104 float charge_mag = num_electrons * 1.6e-19; // C
105
106 // Initialize masses for all of the particles in Kg
107 Scalar_Group mass;
108 for ( size_t i=0 ; i<NUM_DUST ; i++ ) mass[i] = single_mass;
109
110 //*****
111 *
112
113 //*****
114 *
115 // FORCE INFORMATION
116
117 // Create the system of ODES
118 System
119 <
120 // Particle_Interaction< Coulomb >,
121 Particle_Interaction< Yukawa >,
122
123 // Particle_Interaction< Coulomb_q_Sin >,
124 Particle_Interaction< Yukawa_q_Sin >,
125 // Particle_Interaction< ShieldedIon3 >,
126
127 E_Field,
128 // t_Sin_E_Field,
129
130 // t_Sin_E_Field_q_Sin,
131
132 Drag,
133 // Other_Drag,
134
135 Thermal_Heater2D_BMTtransform
136 > system;
```

```
132
133 // set info for RNG
134 std::shared_ptr<boost::random::mt11213b> gen =
135     std::make_shared<boost::random::mt11213b>();
136 // std::shared_ptr<boost::random::normal_distribution<float>> dist =
137 //     std::make_shared<boost::random::normal_distribution<float>> (0,
138 // ... 1);
139
140 std::shared_ptr<boost::random::uniform_real_distribution<float>> dist =
141
142 ... std::make_shared<boost::random::uniform_real_distribution<float>>(0,1);
143 int seed_val = 2;
144 gen->seed( seed_val );
145
146 system.Thermal_Heater2D_BMTransform::set_mass( mass );
147 system.Thermal_Heater2D_BMTransform::set_magnitude( 1e-14 ); //room
148 ... temp
149 system.Thermal_Heater2D_BMTransform::set_rand_gen( gen );
150 system.Thermal_Heater2D_BMTransform::set_uniform_dist( dist );
151
152 // system.Drag::set_drag_coeff( 57.12 ); // 1/s // P = 0.2 mBar
153 // system.Drag::set_drag_coeff( 114.25 ); // 1/s // P = 0.4 mBar
154 system.Drag::set_drag_coeff( 171.372 ); // 1/s // P = 0.6 mBar
155
156 // system.Other_Drag::set_other_drag_coeff( 236.4 ); // 1/s // (0.6,0.7,
157 // ... INJECTION)
158 // system.Other_Drag::set_other_drag_coeff( 324.31); // 1/s
159
160
161 system.E_Field::set_mass( mass );
162 system.E_Field::set_e_field( {227, 0, 0} ); // V/m // (0.6,0.7)
163 // system.E_Field::set_e_field( {227, 0, 0} ); // V/m
164
165 // system.t_Sin_E_Field::set_mass( mass ); // kg
166 // system.t_Sin_E_Field::set_e_field( {227, 0, 0} ); // V/m // (0.6,0.7)
167 // system.t_Sin_E_Field::set_e_field( {216.8, 0, 0} ); // V/m
168 // system.t_Sin_E_Field::set_freq( 250 ); // f, s^-1
169
170 // system.t_Sin_E_Field_q_Sin::set_mass( mass ); // kg
171 // system.t_Sin_E_Field_q_Sin::set_e_field( {227, 0, 0} ); // V/m
172 // system.t_Sin_E_Field_q_Sin::set_e_freq( 500 ); // s^-1
173 // system.t_Sin_E_Field_q_Sin::set_q_amp( .5 );
174 // system.t_Sin_E_Field_q_Sin::set_q_freq( 100000 ); // f, 1/s
175
176 // system.Particle_Interaction< Coulomb >::set_mass( mass ); // kg
177 // system.Particle_Interaction< Coulomb >::set_epsilon_0( 8.84e-12 );//
178 // ... F/m
179
```

```
175 // system.Particle_Interaction< Coulomb_q_Sin >::set_mass( mass ); // kg
176 // system.Particle_Interaction< Coulomb_q_Sin >::set_epsilon_0( 8.84e-12
... );// F/m
177 // system.Particle_Interaction< Coulomb_q_Sin >::set_q_amp( .5 );
178 // system.Particle_Interaction< Coulomb_q_Sin >::set_q_freq( 100000 );
... // f, 1/s
179
180 system.Particle_Interaction< Yukawa >::set_mass( mass ); // kg
181 system.Particle_Interaction< Yukawa >::set_epsilon_0( 8.84e-12 ); //
... F/m
182 system.Particle_Interaction< Yukawa >::set_debye( 1.75e-3 ); // m
183
184
185 // system.Particle_Interaction< Yukawa_q_Sin >::set_mass( mass ); // kg
186 // system.Particle_Interaction< Yukawa_q_Sin >::set_debye( 417.8e-6 );
... // m
187 // system.Particle_Interaction< Yukawa_q_Sin >::set_epsilon_0( 8.84e-12
... ); // F/m
188 // system.Particle_Interaction< Yukawa_q_Sin >::set_q_amp( 0.5 ); // []
189 // system.Particle_Interaction< Yukawa_q_Sin >::set_q_freq( 100000 ); //
... f, 1/s
190
191 // system.Particle_Interaction< ShieldedIon3 >::set_mass( mass ); // kg
192 // system.Particle_Interaction< ShieldedIon3 >::set_epsilon_0( 8.84e-12
... ); // F/m
193 // system.Particle_Interaction< ShieldedIon3 >::set_debye( 200e-6 ); //
... m
194 // system.Particle_Interaction< ShieldedIon3 >::set_alpha( 1.5 ); // []
195 // system.Particle_Interaction< ShieldedIon3 >::set_alpha( 2.0 ); // []
196 // system.Particle_Interaction< ShieldedIon3 >::set_beta( 3.5 ); // []
197 // system.Particle_Interaction< ShieldedIon3 >::set_beta( 4.25 ); // []
198 // system.Particle_Interaction< ShieldedIon3 >::set_gamma ( 0.85 ); //
... []
199 // system.Particle_Interaction< ShieldedIon3 >::set_gamma ( 0.95 ); //
... []
200
201
202 //*****
... *
203
204 //*****
... *
205 // POSITION AND TIME INFORMATION
206
207 srand( time(NULL) );
208 int pos_mult = NUM_DUST * 100;
```



```
209 // create the initial state
210 Plasma_State state;
211 float x_pos, y_pos, z_pos;
212
213 // bool readdataFromFile = true; // Read from file (T) or generate
... Random values (F)
214 bool readdataFromFile = false;
215
216 for( size_t i=0 ; i<NUM_DUST ; i++)
217 {
218     if(readdataFromFile)
219     {
220         std::vector<float> data = readLastLine();
221         //ADDED TO CHANGE THE CHARGE, NOT READ IN PREVIOUS
222         state[i].charge = -charge_mag; // C
223
224         // state[i].charge = data[7 * i + 7]; // C
225         state[i].vel = {data[7 * i + 4], data[7 * i + 5], data[7 * i +
... 6]}; // m/s
226         state[i].pos = {data[7 * i + 1], data[7 * i + 2], data[7 * i +
... 3] }; // m
227     }
228     else
229     {
230         state[i].charge = -charge_mag; // C
231
232         state[i].vel = {0.0, 0.0, 0.0}; // m/s
233         // state[i].vel = {-0.0002, 0, 0}; // m/s
234         // vx = float(float( ((rand() % 2) - 1) ) / 1000);
235         // vy = float(float( ((rand() % 2) - 1) ) / 1000);
236         // vz = 0;
237         // state[i].vel = {vx, vy, vz}; // m/s
238
239         x_pos = float(float( ((rand() % (2*pos_mult)) - pos_mult) ) *
... .01 / pos_mult);
240         y_pos = float(float( ((rand() % (2*pos_mult)) - pos_mult) ) *
... .01 / pos_mult);
241         z_pos = 0.0;
242         // z_pos = float(float( ((rand() % (2*pos_mult)) - pos_mult) ) *
... .01 / pos_mult);
243         // z_pos = (float( rand() % pos_mult ) / pos_mult) * .0003 ;
244         state[i].pos = { x_pos, y_pos, z_pos }; // m
245
246     } // else
247 } //for
248
249
250 // time step setup- Normal Dust
```

```
251     double dt = 10e-7; // stepsize 1ms
252 //     double t0 = 0.0; // beginning time initial
253     double t0 = 1.55; // end time initial; beginning time injection
254     double t1 = 1.75; // end time injection
255
256     // time step setup- Charging
257 //     double dt = 10e-7; // stepsize 1µs
258 //     double t0 = 1.7513; // beginning time heating
259 //     double t1 = 1.7713; // end time heating test- to find cutoff value in
... data
260 //     double t0 = 1.7522; // end time heating real; beginning time cooling
261 //     double t1 = 2.2522; // .1s for testing
262 //     double t1 = 2.2513; // 0.25s
263 //     double t1 = 2.0013;
264
265 //     double t0 = 1.9523; // end time; rounded- 2 frames = 0.029, 4 frames
... = 0.058s
266
267
268 //*****
... *
269 //
270 // integrate
271     integrate_const(
272         runge_kutta4< Plasma_State >(),
273         system,
274         state,
275         t0, t1, dt,
276         Streaming_Observer( std::cout )
277
278     );
279
280 return 0;
281 } // main
282
```

### B.1.3. Basic\_forces.hpp

File containing the definition of all possible forces used in the simulation.

```

/Users/Loriscott/Documents/Yoakum/basic_forces.hpp
Printed: 6/14/22, 5:05:47 PM
Page 1/22
Printed for: Lori Scott
1 #ifndef YOAKUM_BASIC_FORCES_INCLUDED
2 #define YOAKUM_BASIC_FORCES_INCLUDED
3
4 /*!
5  * \file basic_forces.hpp
6  *
7  * \author Dustin Sanford
8  * \date July 17, 2018
9  *
10 * A set of force classes.
11 */
12
13 // Required for Plasma_State
14 #include "configure.hpp"
15 // Required for sin() in t_Sin_E_Field
16 #include <cmath>
17 // Required by Thermal_Heater
18 #include <boost/random/mersenne_twister.hpp>
19 #include <boost/random/normal_distribution.hpp>
20 // Required by Thermal_Heater
21 #include <memory>
22
23 namespace yoakum {
24
25     //! Constant Electric Field (ODE Functor)
26     /*!
27     * E_Field follows the \a ODE \a functor concept. The ODE models a
28     ... change
29     * in particle velocity due to an electric field. The electric field
30     * is held constant in space and time. For a single particle:
31     *
32     *  $\vec{v}' = \frac{q\vec{E}}{m}$ 
33     * -  $q$  is the particle charge. ( $C$ )
34     * -  $\vec{E}$  is the electric field. ( $V/m$ )
35     * -  $m$  is the particle mass. ( $Kg$ )
36     */
37     class E_Field
38     {
39     public:
40
41         //! ODE Calculation
42         /*!
43         * Perform the calculation specified by the ODE. For more
44         ... information
45         * about the contract provided by operator() please see the \a
46         ... ODE \a
47         * functor concept documentation.
48         *

```

```
46     * \param[in] state The initial Plasma_State of the system.
47     * \param[in,out] dsdt The cumulative change in Plasma_State.
48     * \param[in] t The current time.
49     */
50     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
51     {
52         for( size_t i=0 ; i<NUM_DUST ; i++ ) {
53             dsdt[i].vel += e_field * state[i].charge / mass[i];
54         }
55     }
56
57     //!< Set the electric field vector. (\f$V/m\f$)
58     void set_e_field( Spatial_Vector e_field_ ) { e_field = e_field_;
... }
59     //!< Set the particle masses. (\f$Kg\f$)
60     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
61
62     private:
63         Spatial_Vector e_field;
64         Scalar_Group mass;
65     };
66
67     //!< Constant Magnetic Field (ODE Functor)
68     /*!
69     * B_Field follows the \a ODE \a functor concept. The ODE models a
... change
70     * in particle velocity due to a magnetic field. The magnetic field
71     * is held constant in space and time. For a single particle:
72     *
73     * 
$$\vec{v}' = \frac{q\vec{v} \times \vec{B}}{m}$$

74     * -  $q$  is the particle charge. (\f$C\f$)
75     * -  $\vec{v}$  is the particle velocity. (\f$m/s\f$)
76     * -  $\vec{B}$  is the magnetic field. (\f$T\f$)
77     * -  $m$  is the particle mass. (\f$Kg\f$)
78     */
79     class B_Field
80     {
81     public:
82
83         //!< ODE Calculation
84         /*!
85         * Perform the calculation specified by the ODE. For more
... information
86         * about the contract provided by operator() please see the \a
... ODE \a
87         * functor concept documentation.
88         *
```



```
89     * \param[in] state The initial Plasma_State of the system.
90     * \param[in,out] dsdt The cumulative change in Plasma_State.
91     * \param[in] t The current time.
92     */
93     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
94     {
95         for( size_t i=0 ; i<NUM_DUST ; i++ ) {
96             dsdt[i].vel += state[i].charge * cross(state[i].vel,
... b_field) / mass[i];
97         }
98     }
99
100     //! Set the magnetic field vector. (\f$T\f$)
101     void set_b_field( Spatial_Vector b_field_ ) { b_field = b_field_;
... }
102     //! Set the particle masses. (\f$Kg\f$)
103     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
104
105 private:
106     Spatial_Vector b_field;
107     Scalar_Group mass;
108 };
109
110     //! Radial Electric Field (ODE Functor)
111     /*!
112     * Radial_E_Field follows the \a ODE \a functor concept. The ODE
... models a
113     * change in particle velocity due to an electric field. The electric
... field
114     * is in the radial direction, has a constant magnitude, and is held
... constant
115     * in time. For a single particle:
116     *
117     *  $v' = -\frac{q E}{m} \hat{r}$ 
118     * -  $q$  is the particle charge. (\f$C\f$)
119     * -  $E$  is the electric field strength. (\f$V/m\f$)
120     * -  $m$  is the particle mass. (\f$Kg\f$)
121     * -  $\hat{r}$  is a unit vector that points from the origin to
... the particle.
122     */
123     class Radial_E_Field
124     {
125     public:
126
127         //! ODE Calculation
128         /*!
129         * Perform the calculation specified by the ODE. For more
```

```
129... information
130     * about the contract provided by operator() please see the \a
... ODE \a
131     * functor concept documentation.
132     *
133     * \param[in] state The initial Plasma_State of the system.
134     * \param[in,out] dsdt The cumulative change in Plasma_State.
135     * \param[in] t The current time.
136     */
137     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
138     {
139         Spatial_Vector r_hat;
140         for( size_t i=0 ; i<NUM_DUST ; i++ ) {
141             r_hat = state[i].pos / abs(state[i].pos); // radial unit
... vector
142             dsdt[i].vel += -e_field * state[i].charge * r_hat /
... mass[i];
143         }
144     }
145
146     //! Set the electric field strength. (\f$V/m\f$)
147     void set_e_field( float e_field_ ) { e_field = e_field_; }
148     //! Set the particle masses. (\f$Kg\f$)
149     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
150
151     private:
152         float e_field;
153         Scalar_Group mass;
154     };
155
156     //! Drag Force (ODE Functor)
157     /*!
158     * Drag follows the \a ODE \a functor concept. The ODE models a
159     * change in particle velocity due to a drag force. The drag force
... apposes
160     * the direction of motion. For a single particle:
161     *
162     * 
$$\vec{v}' = -\frac{\vec{v}C}{m}$$

163     * -  $\vec{v}$  is the particle velocity. (\f$m/s\f$)
164     * -  $C$  is the drag coefficient. (\f$Ns/m\f$)
165     * -  $m$  is the particle mass. (\f$Kg\f$)
166     */
167     class Drag
168     {
169     public:
170
171         //! ODE Calculation
```

```
172     /*!  
173     * Perform the calculation specified by the ODE. For more  
... information  
174     * about the contract provided by operator() please see the \a  
... ODE \a  
175     * functor concept documentation.  
176     *  
177     * \param[in] state The initial Plasma_State of the system.  
178     * \param[in,out] dsdt The cumulative change in Plasma_State.  
179     * \param[in] t The current time.  
180     */  
181     void operator()( const Plasma_State& state, Plasma_State& dsdt,  
... const double t )  
182     {  
183         for( size_t i=0 ; i<NUM_DUST ; i++ ) {  
184             dsdt[i].vel += - drag_coeff * state[i].vel;  
185         }  
186     }  
187  
188     /*! Set the drag coeficient. (\f$Ns/m\f$)  
189     void set_drag_coeff( float drag_coeff_ ) { drag_coeff =  
... drag_coeff_; }  
190  
191     private:  
192         float drag_coeff;  
193     };  
194  
195     /*! Other Drag Force (ODE Functor)  
196     /*!  
197     * Drag follows the \a ODE \a functor concept. The ODE models a  
198     * change in particle velocity due to a drag force. The drag force  
... apposes  
199     * the direction of motion. For a single particle:  
200     *  
201     *  $v' = -\frac{\vec{v}C}{m}$   
202     * -  $\vec{v}$  is the particle velocity. (\f$m/s\f$)  
203     * -  $C$  is the drag coefitient. (\f$Ns/m\f$)  
204     * -  $m$  is the particle mass. (\f$Kg\f$)  
205     */  
206     class Other_Drag  
207     {  
208     public:  
209  
210         /*! ODE Calculation  
211         /*!  
212         * Perform the calculation specified by the ODE. For more  
... information  
213         * about the contract provided by operator() please see the \a
```

```
213... ODE \a
214     * functor concept documentation.
215     *
216     * \param[in] state The initial Plasma_State of the system.
217     * \param[in,out] dsdt The cumulative change in Plasma_State.
218     * \param[in] t The current time.
219     */
220     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
221     {
222         for( size_t i=0 ; i<NUM_DUST ; i++ ) {
223             dsdt[i].vel += - other_drag_coeff * state[i].vel;
224         }
225     }
226
227     //! Set the drag coefficient. (\f$Ns/m\f$)
228     void set_other_drag_coeff( float other_drag_coeff_ )
... {other_drag_coeff = other_drag_coeff_; }
229
230     private:
231         float other_drag_coeff;
232     };
233
234     //! Electron Drag Force (ODE Functor)
235     /*!
236     * Electron drag follows the \a ODE \a functor concept. The ODE
... models a
237     * change in particle velocity due to a electron drag force. The drag
... force in
238     * the direction of the electron motion. The force is in the
... x-direction and a
239     * normal distribution centered at the beam_center and having a width
... of beam_width
240     * in the y-direction. For a single particle:
241     *
242     * \f[v' = 2 \pi r_{d}^{2} n_{e, beam} K_{e, beam} \exp{\left[
... \frac{\left( x - x_{o,beam} \right)^{2}}{2 \sigma_{e, beam}^{2}} \right]}
... /m
243     * - \f$\vec{v}\f$ is the particle velocity. (\f$m/s\f$)
244
245     * - \f$r_{d}\f$ is the dust radius. (\f$m/s\f$)
246     * - \f$n_{e, beam}\f$ is the density of the electron beam. (\f$
... m^{-3} \f$)
247     * - \f$K_{e, beam}\f$ is the energy of the electron beam. (\f$J\f$)
248     * - \f$x_{o,beam}\f$ is the beam center. (\f$m \f$)
249     * - \f$\sigma_{e, beam}\f$ is the beam width. (\f$m \f$)
250     */
251     class Electron_Drag
```



```
252     {
253     public:
254
255         //! ODE Calculation
256         /*!
257         * Perform the calculation specified by the ODE. For more
... information
258         * about the contract provided by operator() please see the \a
... ODE \a
259         * functor concept documentation.
260         *
261         * \param[in] state The initial Plasma_State of the system.
262         * \param[in,out] dsdt The cumulative change in Plasma_State.
263         * \param[in] t The current time.
264         */
265         void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
266         {
267             for( size_t i=0 ; i<NUM_DUST ; i++ ) {
268                 dsdt[i].vel += 2 * 3.14159265359 * dust_radius *
... dust_radius * beam_density *
269                 beam_energy * exp(-pow(state[i].pos[1] - beam_center,2) /
... (2 * pow(beam_width,2)) ) /mass[i];
270                 // dsdt[i].charge += PHYSICS TO CHANGE CHARGE HERE
271             }
272         }
273
274         //! Set the drag coeficient. (\f$Ns/m\f$)
275         void set_dust_radius( float dust_radius_ ) { dust_radius =
... dust_radius_; }
276         void set_beam_density( float beam_density_ ) { beam_density =
... beam_density_; }
277         void set_beam_energy( Spatial_Vector beam_energy_ ) { beam_energy =
... beam_energy_; }
278         void set_mass( Scalar_Group mass_ ) { mass = mass_; }
279
280         void set_beam_center( float beam_center_ ) { beam_center =
... beam_center_; }
281         void set_beam_width( float beam_width_ ) { beam_width =
... beam_width_; }
282
283
284
285     private:
286         float dust_radius;
287         float beam_density;
288         Spatial_Vector beam_energy;
289         Scalar_Group mass;
```

```
290     float beam_center;
291     float beam_width;
292
293 };
294
295     //!< Gravity (ODE Functor)
296     /*!
297     * Gravity follows the \a ODE \a functor concept. The ODE models a
298     * change in particle velocity due to gravity. For a single particle:
299     *
300     * \f[v' = g\hat{u}\f]
301     * - \f$g\f$ is the acceleration due to gravity. (\f$m/s^2\f$)
302     * - \f$\hat{u}\f$ is the unit vector that points in the direction
... that
303     *         gravity acts along.
304     */
305     class Gravity
306     {
307     public:
308
309         //!< ODE Calculation
310         /*!
311         * Perform the calculation specified by the ODE. For more
... information
312         * about the contract provided by operator() please see the \a
... ODE \a
313         * functor concept documentation.
314         *
315         * \param[in] state The initial Plasma_State of the system.
316         * \param[in,out] dsdt The cumulative change in Plasma_State.
317         * \param[in] t The current time.
318         */
319         void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
320         {
321             for( size_t i=0 ; i<NUM_DUST ; i++ ) {
322                 dsdt[i].vel += gravity * direction;
323             }
324         }
325
326         //!< Set the gravitational strength. (\f$m/s^2\f$)
327         void set_gravity( float gravity_ ) { gravity = gravity_; }
328         //!< Set the direction that gravity acts along.
329         /*!
330         * \param[in] direction_ is the direction that gravity acts
... along. It
331         *         should be a unit vector. However if its length is not one,
... it
```

```
332     *         is normalized to one (its direction is preserved).
333     */
334     void set_direction( Spatial_Vector direction_ )
335     {
336         direction = direction_ / abs(direction_);
337     }
338
339     private:
340         float gravity;
341         Spatial_Vector direction;
342     };
343
344     //! Electric Field Varying Sinusoidally In Time (ODE Functor)
345     /*!
346     * t_Sin_E_Field follows the \a ODE \a functor concept. The ODE
... models a
347     * change in particle velocity due to an electric field. The electric
... field
348     * is held constant in space and varies sinusoidally in time. For a
... single
349     * particle:
350     *
351     *  $\vec{v} = \frac{q\vec{E}}{m}\sin{\omega t}$ 
352     * -  $q$  is the particle charge. (C)
353     * -  $\vec{E}$  is the electric field. (V/m)
354     * -  $m$  is the particle mass. (Kg)
355     * -  $t$  is the current time. (s)
356     * -  $\omega$  is the frequency of the electric field oscillation.
... ( $s^{-1}$ )
357     */
358     class t_Sin_E_Field
359     {
360     public:
361
362         //! ODE Calculation
363         /*!
364         * Perform the calculation specified by the ODE. For more
... information
365         * about the contract provided by operator() please see the \a
... ODE \a
366         * functor concept documentation.
367         *
368         * \param[in] state The initial Plasma_State of the system.
369         * \param[in,out] dsdt The cumulative change in Plasma_State.
370         * \param[in] t The current time.
371         */
372         void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
```

```
373     {
374         for( size_t i=0 ; i<NUM_DUST ; i++ ) {
375             dsdt[i].vel += e_field * state[i].charge * sin( 2 *
... 3.14159265358979 * freq * t) / mass[i];
376         }
377     }
378
379     //! Set the electric field vector. (\f$ V/m \f$)
380     void set_e_field( Spatial_Vector e_field_ ) { e_field = e_field_;
... }
381     //! Set the particle masses. (\f$ Kg \f$)
382     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
383     //! Set the frequency of the electric field oscillation. (\f$
... s^{-1} \f$)
384     void set_freq( float freq_ ) { freq = freq_; }
385
386     private:
387         Spatial_Vector e_field;
388         Scalar_Group mass;
389         float freq;
390     };
391
392     //! Electric Field Varying Sinusoidally In Time (ODE Functor)
393     /*!
394     * t_Sin_E_Field follows the \a ODE \a functor concept. The ODE models a
395     * change in particle velocity due to an electric field. The electric
... field
396     * is held constant in space and varies sinusoidally in time. For a
... single
397     * particle:
398     *
399     * 
$$\vec{v}' = \frac{q\vec{E}}{m}\sin\{t\omega\}$$

400     * - \f$q\f$ is the particle charge. (\f$ C \f$)
401     * - \f$\vec{E}\f$ is the electric field. (\f$ V/m \f$)
402     * - \f$m\f$ is the particle mass. (\f$ Kg \f$)
403     * - \f$t\f$ is the current time. (\f$ s \f$)
404     * - \f$\omega\f$ is the frequency of the electric field oscillation.
... (\f$ s^{-1} \f$)
405     */
406     class t_Sin_E_Field_q_Sin
407     {
408     public:
409
410         //! ODE Calculation
411         /*!
412         * Perform the calculation specified by the ODE. For more information
413         * about the contract provided by operator() please see the \a ODE \a
414         * functor concept documentation.
```



```
415     *
416     * \param[in] state The initial Plasma_State of the system.
417     * \param[in,out] dsdt The cumulative change in Plasma_State.
418     * \param[in] t The current time.
419     */
420     void operator()( const Plasma_State& state, Plasma_State& dsdt, const
... double t )
421     {
422         for( size_t i=0 ; i<NUM_DUST ; i++ ) {
423             dsdt[i].vel += e_field * sin( 2 * 3.14159265358979 * e_freq *
... t) * state[i].charge * (1 + q_amp * sin(- 2 * 3.14159265358979 * q_freq *
... t)) / mass[i];
424         }
425     }
426
427     //! Set the electric field vector. (\f$ V/m \f$)
428     void set_e_field( Spatial_Vector e_field_ ) { e_field = e_field_; }
429     //! Set the particle masses. (\f$ Kg \f$)
430     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
431     //! Set the frequency of the electric field oscillation. (\f$ s^{-1}
... \f$)
432     void set_e_freq( float e_freq_ ) { e_freq = e_freq_; }
433     //! Set the charge variation amplitude
434     void set_q_amp( float q_amp_ ) { q_freq = q_amp_; }
435     //! Set the frequency of the charge fluctuation
436     void set_q_freq( float q_freq_ ) { q_freq = q_freq_; }
437
438 private:
439     Spatial_Vector e_field;
440     Scalar_Group mass;
441     float e_freq;
442     float q_amp;
443     float q_freq;
444 };
445
446     //! Linear Radial Electric Field (ODE Functor)
447     /*!
448     * Linear_Radial_E_Field follows the \a ODE \a functor concept.
449     * The ODE models a change in particle velocity due to an electric
450     * field. The electric field is held constant in time. The field
... points
451     * towards the origin and increases in magnitude linearly with
... distance
452     * from the origin. For a single particle:
453     *
454     * 
$$\vec{v}'_i = -\frac{q x_i E_i}{m} \hat{e}_i$$

455     * -  $q$  is the particle charge. (\f$ C \f$)
456     * -  $\vec{E}$  is the electric field. (\f$ V/m \f$)
```

```
457     * - \f$\vec{x}\f$ is the particle position. (\f$ m \f$)
458     * - \f$m\f$ is the particle mass. (\f$ Kg \f$)
459     */
460     class Linear_Radial_E_Field
461     {
462     public:
463
464         //! ODE Calculation
465         /*!
466         * Perform the calculation specified by the ODE. For more
... information
467         * about the contract provided by operator() please see the \a
... ODE \a
468         * functor concept documentation.
469         *
470         * \param[in] state The initial Plasma_State of the system.
471         * \param[in,out] dsdt The cumulative change in Plasma_State.
472         * \param[in] t The current time.
473         */
474         void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
475         {
476             for( size_t i=0 ; i<NUM_DUST ; i++ ) {
477                 // e_field and state.pos are both Spatial_Vectors and are
... multiplied
478                 // component wise. The result is another Spatial_Vector.
479                 dsdt[i].vel += - e_field * state[i].pos * state[i].charge
... / mass[i];
480             }
481         }
482
483         //! Set the electric field vector. (\f$V/m\f$)
484         void set_e_field( Spatial_Vector e_field_ ) { e_field = e_field_;
... }
485         //! Set the particle masses. (\f$Kg\f$)
486         void set_mass( Scalar_Group mass_ ) { mass = mass_; }
487
488     private:
489         Spatial_Vector e_field;
490         Scalar_Group mass;
491     };
492
493     //! Thermal Heater (ODE Functor)
494     /*!
495     * Thermal_Heater follows the \a ODE \a functor concept. The ODE
... models
496     * a change in particle velocity due to a "thermal force." The
... current
```

```
497     * implementation is highly heuristic in nature. The force is
... calculated
498     * by selecting a random number from a normal distribution and
... scaling it.
499     * For a single particle:
500     *
501     *  $v'_i = \frac{h\phi_i}{m}$ 
502     * -  $h$  is a scaler which determines the magnitude of the
... average
503     *     "thermal force." ( $N$ )
504     * -  $\phi$  is a number selected from a normal distribution where
505     *      $\mu = 0$  and  $\sigma = 1$ . A new number is
... selected
506     *     for each time step and spatial axis.
507     *     -  $m$  is the particle mass.
508     *
509     * The normal distribution is generated with the generator
510     *     boost::random::mt11213b and distribution
... boost::random::normal_distribution.
511     * For more information see the random number documentation.
512     */
513     class Thermal_Heater
514     {
515     public:
516
517         //! ODE Calculation
518         /*!
519         * Perform the calculation specified by the ODE. For more
... information
520         * about the contract provided by operator() please see the \a
... ODE \a
521         * functor concept documentation.
522         *
523         * \param[in] state The initial Plasma_State of the system.
524         * \param[in,out] dsdt The cumulative change in Plasma_State.
525         * \param[in] t The current time.
526         */
527         void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
528         {
529             for( size_t i=0 ; i<NUM_DUST ; i++ )
530             {
531                 // calculate a "thermal force" for each spatial axis
532                 for( size_t j=0 ; j<DIM ; j++ )
533                 {
534                     /* Both normal_dist and rand_gen are stored as
... pointers.
535                     * Thus (*normal_dist)(*rand_gen) is semantically
```

```
535... equivalent
536             * to normal_dist( rand_gen )
537             */
538             dsdt[i].vel[j] += magnitude *
... (*normal_dist)(*rand_gen) / mass[i];
539         }
540     }
541 }
542
543     //!< Set the magnitude of the average "thermal force." (\f$ N \f$)
544     void set_magnitude( float magnitude_ ) { magnitude = magnitude_; }
545     //!< Set the particle masses. (\f$ Kg \f$)
546     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
547     //!< Assign a random number generator.
548     /*!
549     * The random number generator can be shared between multiple
550     * ODE functors and should be initialized outside of
... Thermal_Heater.
551     * For more information see the random number generation
... documentation.
552     */
553     void set_rand_gen( std::shared_ptr< boost::random::mt11213b >
... rand_gen_ )
554     {
555         rand_gen = rand_gen_;
556     }
557     //!< Assign a normal distribution generator
558     /*!
559     * The distribution function can be shared between multiple ODE
560     * functors and should be initialized outside of
... Thermal_Heater.
561     * For more information see the random number generation
... documentation.
562     */
563     void set_normal_dist
564     (
565         std::shared_ptr< boost::random::normal_distribution<float> >
... normal_dist_
566     )
567     {
568         normal_dist = normal_dist_;
569     }
570
571     private:
572         float magnitude;
573         Scalar_Group mass;
574         // A random number generator
575         std::shared_ptr< boost::random::mt11213b > rand_gen;
```



```
576         // A normal distribution functor with sigma = 1 and mu = 0
577         std::shared_ptr< boost::random::normal_distribution<float> >
... normal_dist;
578     };
579
580     //! Thermal Heater 2D (ODE Functor)
581     /*!
582     * Thermal_Heater follows the \a ODE \a functor concept. The ODE
... models
583     * a change in particle velocity due to a "thermal force." The
... current
584     * implementation is highly heuristic in nature. The force is
... calculated
585     * by selecting a random number from a normal distribution and
... scaling it.
586     * For a single particle:
587     *
588     *  $v'_i = \frac{h\phi_i}{m}h$ 
589     * -  $h$  is a scaler which determines the magnitude of the
... average
590     * "thermal force." ( $N$ )
591     * -  $\phi$  is a number selected from a normal distribution where
592     *  $\mu = 0$  and  $\sigma = 1$ . A new number is
... selected
593     * for each time step and spatial axis.
594     * -  $m$  is the particle mass.
595     *
596     * The normal distribution is generated with the generator
597     * boost::random::mt11213b and distribution
... boost::random::normal_distribution.
598     * For more information see the random number documentation.
599     */
600     class Thermal_Heater2D
601     {
602     public:
603
604         //! ODE Calculation
605         /*!
606         * Perform the calculation specified by the ODE. For more
... information
607         * about the contract provided by operator() please see the \a
... ODE \a
608         * functor concept documentation.
609         *
610         *  $state$  The initial Plasma_State of the system.
611         *  $dsdt$  The cumulative change in Plasma_State.
612         *  $t$  The current time.
613         */
```

```
614     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
615     {
616         for( size_t i=0 ; i<NUM_DUST ; i++ )
617         {
618             // calculate a "thermal force" for each spatial axis
619             // for( size_t j=0 ; j<DIM ; j++ )
620             for( size_t j=0 ; j<DIM-1 ; j++ ) //Modified so 2D
621             {
622                 /* Both normal_dist and rand_gen are stored as
... pointers.
623                 * Thus (*normal_dist)(*rand_gen) is semantically
... equivalent
624                 * to normal_dist( rand_gen )
625                 */
626                 dsdt[i].vel[j] += magnitude *
... (*normal_dist)(*rand_gen) / mass[i];
627             }
628             dsdt[i].vel[DIM-1] += 0;
629         }
630     }
631
632     //! Set the magnitude of the average "thermal force." (\f$ N \f$)
633     void set_magnitude( float magnitude_ ) { magnitude = magnitude_; }
634     //! Set the particle masses. (\f$ Kg \f$)
635     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
636     //! Assign a random number generator.
637     /*!
638     * The random number generator can be shared between multiple
639     * ODE functors and should be initialized outside of
... Thermal_Heater.
640     * For more information see the random number generation
... documentation.
641     */
642     void set_rand_gen( std::shared_ptr< boost::random::mt11213b >
... rand_gen_ )
643     {
644         rand_gen = rand_gen_;
645     }
646     //! Assign a normal distribution generator
647     /*!
648     * The distribution function can be shared between multiple ODE
649     * functors and should be initialized outside of
... Thermal_Heater.
650     * For more information see the random number generation
... documentation.
651     */
652     void set_normal_dist
```

```
653     (
654         std::shared_ptr< boost::random::normal_distribution<float> >
... normal_dist_
655     )
656     {
657         normal_dist = normal_dist_;
658     }
659
660 private:
661     float magnitude;
662     Scalar_Group mass;
663     // A random number generator
664     std::shared_ptr< boost::random::mt11213b > rand_gen;
665     // A normal distribution functor with sigma = 1 and mu = 0
666     std::shared_ptr< boost::random::normal_distribution<float> >
... normal_dist;
667     };
668
669 //! Thermal Heater 2D (ODE Functor)
670 /*!
671 * Thermal_Heater follows the \a ODE \a functor concept. The ODE models
672 * a change in particle velocity due to a "thermal force." The current
673 * implementation is highly heuristic in nature. The force is calculated
674 * by selecting a random number from a normal distribution and scaling
... it.
675 * For a single particle:
676 *
677 *  $v'_i = \frac{h\phi_i}{m}f$ 
678 * -  $h$  is a scaler which determines the magnitude of the average
679 *   "thermal force." ( $N$  is)
680 * -  $\phi$  is a number selected from a normal distribution where
681 *    $\mu = 0$  and  $\sigma = 1$ . A new number is selected
682 *   for each time step and spatial axis.
683 * -  $m$  is the particle mass.
684 *
685 * The normal distribution is generated with the generator
686 *   boost::random::mt11213b and distribution
... boost::random::normal_distribution.
687 * For more information see the random number documentation.
688 */
689 class Thermal_Heater2D_BMTransform
690 {
691 public:
692
693     //! ODE Calculation
694     /*!
695     * Perform the calculation specified by the ODE. For more information
696     * about the contract provided by operator() please see the \a ODE \a
```

```
697     * functor concept documentation.
698     *
699     * \param[in] state The initial Plasma_State of the system.
700     * \param[in,out] dsdt The cumulative change in Plasma_State.
701     * \param[in] t The current time.
702     */
703     void operator()( const Plasma_State& state, Plasma_State& dsdt, const
... double t )
704
705     {
706         float randx1;
707         float randx2;
708         float randz1;
709         float randz2;
710
711         for( size_t i=0 ; i<NUM_DUST ; i++ )
712         {
713             // calculate a "thermal force" for each spatial axis
714             // Use Box-Muller Transformation of 2 rand# to get normal dist
715             randx1 = (*uniform_dist)(*rand_gen);
716             randx2 = (*uniform_dist)(*rand_gen);
717
718             randz1 = sqrt(-2 * log( randx1 )) * cos( 2 * 3.141592658 *
... randx2 );
719             randz2 = sqrt(-2 * log( randx1 )) * sin( 2 * 3.141592658 *
... randx2 );
720
721
722             dsdt[i].vel[0] += magnitude * randz1 / mass[i];
723             dsdt[i].vel[1] += magnitude * randz2 / mass[i];
724         }
725     }
726
727     //! Set the magnitude of the average "thermal force." (\f$ N \f$)
728     void set_magnitude( float magnitude_ ) { magnitude = magnitude_; }
729     //! Set the particle masses. (\f$ Kg \f$)
730     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
731     //! Assign a random number generator.
732     /*!
733     * The random number generator can be shared between multiple
734     * ODE functors and should be initialized outside of Thermal_Heater.
735     * For more information see the random number generation
... documentation.
736     */
737     void set_rand_gen( std::shared_ptr< boost::random::mt11213b >
... rand_gen_ )
738     {
739         rand_gen = rand_gen_;
```



```
740     }
741
742     void set_uniform_dist
743     (
744         std::shared_ptr< boost::random::uniform_real_distribution<float> >
745         uniform_dist_
746     )
747     {
748         uniform_dist = uniform_dist_;
749     }
750 private:
751     float magnitude;
752     Scalar_Group mass;
753     // A random number generator
754     std::shared_ptr< boost::random::mt11213b > rand_gen;
755     // A normal distribution functor with sigma = 1 and mu = 0
756     std::shared_ptr< boost::random::uniform_real_distribution<float> >
757     uniform_dist;
758 };
759
760     //!< Tangential Force (ODE Functor)
761     /*!
762     * Tangential_Force follows the \a ODE \a functor concept. The ODE
763     ... models a
764     * change in particle velocity due to a force that acts tangentially
765     ... to a given
766     * axis. The force varies linearly with distance from the axis. For a
767     ... single
768     * particle:
769     *
770     *  $\vec{v}' = \frac{\vec{x} \times \hat{u}}{m} f$ 
771     * -  $f$  is the magnitude of the force. (N/m)
772     * -  $\vec{x}$  is the particle position. (m)
773     * -  $\hat{u}$  is a unit vector in the direction of the
774     ... specified axis.
775     * -  $m$  is the particle mass. (Kg)
776     */
777     class Tangential_Force
778     {
779     public:
780
781         //!< ODE Calculation
782         /*!
783         * Perform the calculation specified by the ODE. For more
784         ... information
```

```
781     * about the contract provided by operator() please see the \a
... ODE \a
782     * functor concept documentation.
783     *
784     * \param[in] state The initial Plasma_State of the system.
785     * \param[in,out] dsdt The cumulative change in Plasma_State.
786     * \param[in] t The current time.
787     */
788     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
789     {
790         for( size_t i=0 ; i<NUM_DUST ; i++ )
791         {
792             dsdt[i].vel += magnitude * cross(state[i].pos, axis) /
... mass[i];
793         }
794     }
795
796     //! Set the force magnitude. (\f$N/m\f$)
797     void set_magnitude( float magnitude_ ) { magnitude = magnitude_; }
798     //! Set the particle masses. (\f$Kg\f$)
799     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
800     //! Set the axis unit vector.
801     /*!
802     * \param[in] axis_ is the axis that the tangential force acts
... about.
803     *     It should be a unit vector. However if its length is not
... one, it
804     *     is normalized to one (its direction is preserved).
805     */
806     void set_axis( Spatial_Vector axis_ )
807     {
808         axis = axis_ / abs(axis_);
809     }
810
811     private:
812         Spatial_Vector axis;
813         float magnitude;
814         Scalar_Group mass;
815     };
816
817     //! Electric Potential Well (ODE Functor)
818     /*!
819     * Confining_E_Potential follows the \a ODE \a functor concept. The
... ODE models a
820     * change in particle velocity due to an electric potential well. The
... force due
821     * to the potential varies linearly with distance from the z axis and
```

```
821... inversly
822     * with distance along the z axis. For a single particle:
823     *
824     * \f[\vec{v}' = \frac{\alpha_1 q}{m} \hat{e}_3 - \sqrt{x_1^2 +
... x_2^2}
825     * \frac{\alpha_2 q}{m} \hat{r}] \f]
826     * - \f$q\f$ is the dust charge. (\f$ C \f$)
827     * - \f$m\f$ is the dust mass. (\f$ kg \f$)
828     * - \f$\vec{x}\f$ is the dust position. (\f$ m \f$)
829     * - \f$\alpha_1\f$ is the electric field strength in the
... \f$\hat{z}\f$
830     * direction. (\f$ V \f$)
831     * - \f$\alpha_2\f$ is the electric field strength in the radial
832     * direction. (\f$ V/m^2 \f$)
833     * - \f$ \hat{r} \f$ is the unit vector parallel to the xy-plane that
834     * points from the z-axis to the dust position.
835     */
836     class Confining_E_Potential
837     {
838     public:
839
840         //! ODE Calculation
841         /*!
842         * Perform the calculation specified by the ODE. For more
... information
843         * about the contract provided by operator() please see the \a
... ODE \a
844         * functor concept documentation.
845         *
846         * \param[in] state The initial Plasma_State of the system.
847         * \param[in,out] dsdt The cumulative change in Plasma_State.
848         * \param[in] t The current time.
849         */
850         void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
851         {
852             for( size_t i=0 ; i<NUM_DUST ; i++ )
853             {
854                 dsdt[i].vel +=
855                 {
856                     // alpha_1 * state[i].charge *
... state[i].pos[0] / mass[i], // x
857                     // alpha_1 * state[i].charge *
... state[i].pos[1] / mass[i], // y
858                     alpha_1 * state[i].charge *
... sqrtf(pow(state[i].pos[0],2) + pow(state[i].pos[1],2))*
... cos(atan2f(state[i].pos[1], state[i].pos[0])) / mass[i], // x
859                     alpha_1 * state[i].charge *
```

```
859... sqrtf(pow(state[i].pos[0],2) + pow(state[i].pos[1],2))*  
... sin(atan2f(state[i].pos[1], state[i].pos[0])) / mass[i],          // y  
860  
861          //          alpha_1 * state[i].charge *  
... sqrtf(pow(state[i].pos[0],2) + pow(state[i].pos[1],2))* state[i].pos[0] *  
... fabs(cos(atan2f(state[i].pos[1], state[i].pos[0]))) / mass[i],      //  
... x  
862          //          alpha_1 * state[i].charge *  
... sqrtf(pow(state[i].pos[0],2) + pow(state[i].pos[1],2))* state[i].pos[1] *  
... fabs(sin(atan2f(state[i].pos[1], state[i].pos[0]))) / mass[i],      //  
... y  
863          // alpha_2 * state[i].charge * fabs(state[i].pos[2]) /  
... mass[i] // z  
864          alpha_2 * state[i].charge * state[i].pos[2] / mass[i]  
... // z  
865  
866          };  
867      }  
868  }  
869  
870      //! Set the electric field strength in the  $\hat{z}$  direction. ( $V$ )  
... direction. ( $V$ )  
871      void set_alpha_1( float alpha_1_ ) { alpha_1 = alpha_1_; }  
872      //! Set the electric field strength in the radial direction. ( $V/m^2$ )  
...  $V/m^2$   
873      void set_alpha_2( float alpha_2_ ) { alpha_2 = alpha_2_; }  
874      //! Set the particle masses. ( $kg$ )  
875      void set_mass( Scalar_Group mass_ ) { mass = mass_; }  
876  
877      private:  
878          float alpha_1;  
879          float alpha_2;  
880          Scalar_Group mass;  
881      };  
882  
883 } // namespace yoakum  
884 #endif // include guard  
885
```



## B.1.4. Interaction\_forces.hpp

File containing the definition of all types of interaction forces used in the simulation.

```
/Users/loriscott/Documents/Yoakum/interaction_forces.hpp Page 1/13
Printed: 6/14/22, 5:07:10 PM Printed for: Lori Scott

1 #ifndef YOAKUM_INTERACTION_FORCES_INCLUDED
2 #define YOAKUM_INTERACTION_FORCES_INCLUDED
3
4 // Required for Plasma_State
5 #include "configure.hpp"
6 // Required for exp() in Yukawa and ShieldedIon
7 #include <cmath>
8
9 namespace yoakum {
10
11     //! Particle Interaction (ODE Functor)
12     /*!
13     * Particle_Interaction follows the \a ODE \a functor concept.
14     * The ODE models a change in particle velocity due to an interaction
15     * between two particles. The exact behavior of the ODE depends on the
16     * Interaction_ODE template parameter, which follows the interaction
17     * concept. The ODE is calculated for every particle pair in the set
18     * \f$ \{(p_i, p_j) \, | \, 0 \leq i < j \leq N \} \f$, where
19     * \f$N\f$
20     * is the number of dust particles. For a single particle pair:
21     * \f[ \vec{v}'_1 = F(p_1, p_2, t) \frac{1}{m_1} \f]
22     * \f[ \vec{v}'_2 = -F(p_1, p_2, t) \frac{1}{m_2} \f]
23     * - \f$F\f$ is the ODE defined by Interaction_ODE.
24     * - \f$p_1\f$ is the first particle's initial state.
25     * - \f$p_2\f$ is the second particle's initial state.
26     * - \f$m_1\f$ is the first particle's mass. (\f$ Kg \f$)
27     * - \f$m_2\f$ is the second particle's mass. (\f$ Kg \f$)
28     * - \f$t\f$ is the current time. (\f$ s \f$)
29     *
30     * \note The Interaction_ODE base class defines additional
31     * initialization
32     * functions. For more information see the interaction force concept
33     * documentation.
34     */
35     template< typename Interaction_ODE >
36     class Particle_Interaction : public Interaction_ODE
37     {
38     public:
39         //! ODE Calculation
40         /*!
41         * Perform the calculation specified by the ODE. For more
42         * information
43         * about the contract provided by operator() please see the \a ODE
44         * \a
45         * functor concept documentation.
46         *
47         */
48     };
49 }
```

```
43     * \param[in] state The initial Plasma_State of the system.
44     * \param[in,out] dsdt The cumulative change in Plasma_State.
45     * \param[in] t The current time.
46     */
47     void operator()( const Plasma_State& state, Plasma_State& dsdt,
... const double t )
48     {
49         Spatial_Vector force;
50         for( size_t i=1 ; i<NUM_DUST ; i++ )
51         {
52             for( size_t j=0 ; j<i ; ++j )
53             {
54                 force = Interaction_ODE::operator() ( state[i],
... state[j], t );
55                 dsdt[i].vel += force / mass[i];
56                 dsdt[j].vel -= force / mass[j];
57             }
58         }
59     }
60
61     //! Set the particle masses. (\f$ Kg \f$)
62     void set_mass( Scalar_Group mass_ ) { mass = mass_; }
63
64     private:
65         Scalar_Group mass;
66     };
67
68
69     //! Coulomb (Interaction Force)
70     /*!
71     * Coulomb follows the \a Interaction \a Force concept. The force
... results
72     * from interparticle coulomb potentials. For a single particle pair:
73     *
74     * 
$$\vec{F}_1 = -\vec{F}_2 =$$

75     * 
$$\frac{1}{4\pi\epsilon_0} \frac{q_1$$

... 
$$q_2}{|\vec{x}_1 - \vec{x}_2|^2} \hat{u}$$

76     * - \f$q_1\f$ is the the first particle's charge. (\f$ C \f$)
77     * - \f$q_2\f$ is the the second particle's charge. (\f$ C \f$)
78     * - \f$\vec{x}_1\f$ is the position of the first particle. (\f$ m
... \f$)
79     * - \f$\vec{x}_2\f$ is the position of the second particle. (\f$ m
... \f$)
80     * - \f$\epsilon_0\f$ is the permittivity of free space. (\f$ F/m \f$)
81     * - \f$\hat{u}\f$ is the unit vector pointing from the first particle
82     *   away from the second particle.
83     */
84     class Coulomb
```

```
85     {
86     public:
87
88         //! Force Calculation
89         /*!
90         * Perform the calculation specified by the Force. For more
... information
91         * about the contract provided by operator() please see the \a
... Interaction
92         * \a Force concept documentation.
93         *
94         * \param[in] p1 The initial state of the first particle.
95         * \param[in] p2 The initial state of the second particle.
96         * \param[in] t The current time.
97         * \return The force on the first particle.
98         */
99         Spatial_Vector operator()( const Dust_State& p1, const Dust_State&
... p2, const double t )
100         {
101             // distance between the two particles
102             float dist = abs(p1.pos - p2.pos);
103             Spatial_Vector r_hat = (p1.pos - p2.pos) / dist;
104             Spatial_Vector fCoulomb = r_hat / ( dist * dist );
105
106             return multiplier * p1.charge * p2.charge * fCoulomb;
107             // return multiplier * p1.charge * p2.charge * r_hat
... / ( dist * dist );
108
109         }
110
111         //! Set the permittivity of free space. (\f$ F/m \f$)
112         // In addition to setting epsilon_0, this function also calculates
... and
113         // caches the constant fraction used by operator().
114         void set_epsilon_0( float epsilon_0 ) {
115             multiplier = 1 / ( 4 * epsilon_0 * 3.141592653589793238463);
116         }
117
118     private:
119         float multiplier;
120     };
121
122     //! Coulomb_Sin (Interaction Force)
123     /*!
124     * Coulomb follows the \a Interaction \a Force concept. The force results
125     * from interparticle coulomb potentials. For a single particle pair:
126     *
127     *  $\vec{F}_1 = -\vec{F}_2 =$ 
```

```
128 *      \frac{1}{4\pi\epsilon_0} \frac{q_1
... q_2}{|\vec{x}_1-\vec{x}_2|^2}\hat{u}\f]
129 * - \f$q_1\f$ is the the first particle's charge. (\f$ C \f$)
130 * - \f$q_2\f$ is the the second particle's charge. (\f$ C \f$)
131 * - \f$\vec{x}_1\f$ is the position of the first particle. (\f$ m \f$)
132 * - \f$\vec{x}_2\f$ is the position of the second particle. (\f$ m \f$)
133 * - \f$\epsilon_0\f$ is the permittivity of free space. (\f$ F/m \f$)
134 * - \f$\hat{u}\f$ is the unit vector pointing from the first particle
135 *      away from the second particle.
136 */
137 class Coulomb_q_Sin
138 {
139 public:
140
141     //! Force Calculation
142     /*!
143     * Perform the calculation specified by the Force. For more
... information
144     * about the contract provided by operator() please see the \a
... Interaction
145     * \a Force concept documentation.
146     *
147     * \param[in] p1 The initial state of the first particle.
148     * \param[in] p2 The initial state of the second particle.
149     * \param[in] t The current time.
150     * \return The force on the first particle.
151     */
152     Spatial_Vector operator()( const Dust_State& p1, const Dust_State& p2,
... const double t )
153     {
154         // distance between the two particles
155         float dist = abs(p1.pos - p2.pos);
156         Spatial_Vector r_hat = (p1.pos - p2.pos) / dist;
157         Spatial_Vector fCoulomb = r_hat / ( dist * dist );
158
159         return multiplier * p1.charge * p2.charge * pow(( 1 + q_amp *
... sin(-2 * 3.14159265358979 * q_freq * t)),2) * fCoulomb;
160         //      return multiplier * p1.charge * p2.charge * r_hat / (
... dist * dist );
161     }
162
163
164     //! Set the permittivity of free space. (\f$ F/m \f$)
165     // In addition to setting epsilon_0, this function also calculates and
166     // caches the constant fraction used by operator().
167     void set_epsilon_0( float epsilon_0 ) {
168         multiplier = 1 / (4 * epsilon_0 * 3.141592653589793238463);
169     }
```



```
170
171     //!< Set the charge variation amplitude
172     void set_q_amp( float q_amp_ ) { q_freq = q_amp_; }
173     //!< Set the frequency of the charge fluctuation
174     void set_q_freq( float q_freq_ ) { q_freq = q_freq_; }
175
176
177 private:
178     float multiplier;
179     float q_amp;
180     float q_freq;
181 };
182
183
184     //!< Yukawa (Interaction Force)
185     /*!
186     * Yukawa follows the \a Interaction \a Force concept. The force
187     ... results
188     * from interparticle coulomb potentials. For a single particle pair:
189     *
190     * 
$$\vec{F}_1 = -\vec{F}_2 =$$

191     * 
$$\frac{1}{4\pi\epsilon_0} \frac{q_1 q_2}{\exp\{-|\vec{x}_1 - \vec{x}_2|/\lambda\}} \frac{|\vec{x}_1 - \vec{x}_2|^2 \hat{u}}{|\vec{x}_1 - \vec{x}_2|^3}$$

192     * - \f$q_1\f$ is the the first particle's charge. (\f$ C \f$)
193     * - \f$q_2\f$ is the the second particle's charge. (\f$ C \f$)
194     * - \f$\vec{x}_1\f$ is the position of the first particle. (\f$ m \f$)
195     * - \f$\vec{x}_2\f$ is the position of the second particle. (\f$ m
196     ... \f$)
197     * - \f$\lambda\f$ is the debye length of the system. (\f$ m \f$)
198     * - \f$\epsilon_0\f$ is the permittivity of free space. (\f$ F/m \f$)
199     * - \f$\hat{u}\f$ is the unit vector pointing from the first particle
200     * away from the second particle.
201     */
202     class Yukawa
203     {
204     public:
205
206         //!< Force Calculation
207         /*!
208         * Perform the calculation specified by the Force. For more
209         ... information
210         * about the contract provided by operator() please see the \a
211         ... Interaction
212         * \a Force concept documentation.
213         *
214         * \param[in] p1 The initial state of the first particle.
215         * \param[in] p2 The initial state of the second particle.
216         * \param[in] t The current time.
```

```
213     * \return The force on the first particle.
214     */
215     Spatial_Vector operator()( const Dust_State& p1, const Dust_State&
... p2, const double t)
216     {
217         // distance between the two particles
218         float dist = abs(p1.pos - p2.pos);
219         Spatial_Vector r_hat = (p1.pos - p2.pos) / dist;
220         // calculates the unit vectored force (ignoring constants)
221         Spatial_Vector fYukawa = exp(-dist/debye) * (1/(dist*dist) +
... 1/(dist*debye)) * r_hat;
222
223         return multiplier * p1.charge * p2.charge * fYukawa;
224     }
225
226     //! Set the permittivity of free space. (\f$ F/m \f$)
227     // In addition to setting epsilon_0, this function also calculates
... and
228     // caches the constant fraction used by operator().
229     void set_epsilon_0( float epsilon_0 )
230     {
231         multiplier = 1 / (4 * epsilon_0 * 3.141592653589793238463);
232     }
233
234     //! Set the debye length. (\f$m\f$)
235     void set_debye( float debye_ ) {debye = debye_; }
236
237     private:
238         float multiplier;
239         float debye;
240     };
241
242     //! Yukawa_q_Sin (Interaction Force)
243     /*!
244     * Yukawa follows the \a Interaction \a Force concept. The force results
245     * from interparticle coulomb potentials. For a single particle pair:
246     *
247     * 
$$\vec{F}_1 = -\vec{F}_2 =$$

248     * 
$$\frac{1}{4\pi\epsilon_0} \frac{q_1 q_2}{\exp\left(-\frac{|\vec{x}_1 - \vec{x}_2|}{\lambda}\right) |\vec{x}_1 - \vec{x}_2|^2} \hat{u}$$

249     * - \f$q_1\f$ is the the first particle's charge. (\f$ C \f$)
250     * - \f$q_2\f$ is the the second particle's charge. (\f$ C \f$)
251     * - \f$\vec{x}_1\f$ is the position of the first particle. (\f$ m \f$)
252     * - \f$\vec{x}_2\f$ is the position of the second particle. (\f$ m \f$)
253     * - \f$\lambda\f$ is the debye length of the system. (\f$ m \f$)
254     * - \f$\epsilon_0\f$ is the permittivity of free space. (\f$ F/m \f$)
255     * - \f$\hat{u}\f$ is the unit vector pointing from the first particle
256     * away from the second particle.
```

```
257 */
258 class Yukawa_q_Sin
259 {
260 public:
261
262     //! Force Calculation
263     /*!
264     * Perform the calculation specified by the Force. For more
265     ... information
266     * about the contract provided by operator() please see the \a
267     ... Interaction
268     * \a Force concept documentation.
269     *
270     * \param[in] p1 The initial state of the first particle.
271     * \param[in] p2 The initial state of the second particle.
272     * \param[in] t The current time.
273     * \return The force on the first particle.
274     */
275     Spatial_Vector operator()( const Dust_State& p1, const Dust_State& p2,
276     ... const double t)
277     {
278         // distance between the two particles
279         float dist = abs(p1.pos - p2.pos);
280         Spatial_Vector r_hat = (p1.pos - p2.pos) / dist;
281         // calculates the unit vectored force (ignoring constants)
282         Spatial_Vector fYukawa = exp(-dist/debye) * (1/(dist*dist) +
283     ... 1/(dist*debye)) * r_hat;
284
285         return multiplier * p1.charge * p2.charge * pow(( 1 + q_amp * sin(-
286     ... 2 * 3.14159265358979 * q_freq * t)),2) * fYukawa;
287     }
288
289     //! Set the permittivity of free space. (\f$ F/m \f$)
290     // In addition to setting epsilon_0, this function also calculates and
291     // caches the constant fraction used by operator().
292     void set_epsilon_0( float epsilon_0 )
293     {
294         multiplier = 1 / (4 * epsilon_0 * 3.141592653589793238463);
295     }
296
297     //! Set the debye length. (\f$m\f$)
298     void set_debye( float debye_ ) {debye = debye_; }
299     //! Set the charge variation amplitude
300     void set_q_amp( float q_amp_ ) { q_freq = q_amp_; }
301     //! Set the frequency of the charge fluctuation
302     void set_q_freq( float q_freq_ ) { q_freq = q_freq_; }
303 private:
```



```
300     float multiplier;
301     float debye;
302     float q_amp;
303     float q_freq;
304 };
305
306
307     //!< ShieldedIon (Interaction Force)
308     /*!
309     * ShieldedIon follows the \a Interaction \a Force concept. The force
... results
310     * from interparticle coulomb potentials. For a single particle pair:
311     *
312     * 
$$\vec{F}_1 = -\vec{F}_2 = \frac{1}{4\pi\epsilon_0} \frac{q_1 q_2}{|\vec{x}_1 - \vec{x}_2|^2} \hat{u}$$

313     *
314     * - \f$q_1\f$ is the the first particle's charge. (\f$ C \f$)
315     * - \f$q_2\f$ is the the second particle's charge. (\f$ C \f$)
316     * - \f$\vec{x}_1\f$ is the position of the first particle. (\f$ m \f$)
317     * - \f$\vec{x}_2\f$ is the position of the second particle. (\f$ m
... \f$)
318     * - \f$\epsilon_0\f$ is the permittivity of free space. (\f$ F/m \f$)
319     * - \f$\hat{u}\f$ is the unit vector pointing from the first particle
320     *     away from the second particle.
321     */
322     class ShieldedIon
323     {
324     public:
325
326         //!< Force Calculation
327         /*!
328         * Perform the calculation specified by the Force. For more
... information
329         * about the contract provided by operator() please see the \a
... Interaction
330         * \a Force concept documentation.
331         *
332         * \param[in] p1 The initial state of the first particle.
333         * \param[in] p2 The initial state of the second particle.
334         * \param[in] t The current time.
335         * \return The force on the first particle.
336         */
337         Spatial_Vector operator()( const Dust_State& p1, const Dust_State&
... p2, const double t)
338         {
339             // distance between the two particles and the wake around dust
... 2
340             float dist_d = abs(p1.pos - p2.pos);
```

```
341     Spatial_Vector r_d = (p1.pos - p2.pos) / dist_d;
342     float dist_lw = abs(p1.pos - (p2.pos - wake_dist));
343     Spatial_Vector r_lw = (p1.pos - (p2.pos - wake_dist)) /
... dist_lw;
344     float dist_rw = abs(p1.pos - (p2.pos + wake_dist));
345     Spatial_Vector r_rw = (p1.pos - (p2.pos + wake_dist)) /
... dist_rw;
346
347     // calculates the unit vectored force (ignoring constants)
348     Spatial_Vector fShieldedIon = exp(-dist_d/debye) * r_d *
... (1/(dist_d*dist_d) + 1/(dist_d*debye))
349     - charge_ratio * exp(-dist_lw/debye) * r_lw *
... (1/(dist_lw*dist_lw) + 1/(dist_lw*debye))
350     - charge_ratio * exp(-dist_rw/debye) * r_rw *
... (1/(dist_rw*dist_rw) + 1/(dist_rw*debye));
351
352     return multiplier * p1.charge * p2.charge * fShieldedIon;
353 }
354
355     //! Set the permittivity of free space. (\f$ F/m \f$)
356     // In addition to setting epsilon_0, this function also calculates
... and
357     // caches the constant fraction used by operator().
358     void set_epsilon_0( float epsilon_0 )
359     {
360         multiplier = 1 / (4 * epsilon_0 * 3.141592653589793238463);
361     }
362
363     //! Set the debye length. (\f$m\f$)
364     void set_debye( float debye_ ) { debye = debye_; }
365     //! Set the ion charge ratio.
366     void set_charge_ratio( float charge_ratio_ ) { charge_ratio =
... charge_ratio_; }
367     //! Set the ion-wake distance from the dust particle. (\f$m\f$)
368     void set_wake_dist( Spatial_Vector wake_dist_ ) { wake_dist =
... wake_dist_; }
369
370     private:
371         float multiplier;
372         float debye;
373         float charge_ratio;
374         Spatial_Vector wake_dist;
375 };
376
377
378     //! ShieldedIon2 (Interaction Force)
379     /*!
380     * ShieldedIon2 follows the \a Interaction \a Force concept. The force
```

```
380... results
381     * from interparticle coulomb potentials. For a single particle pair:
382     *
383     *  $\vec{F}_1 = -\vec{F}_2 =$ 
384     *  $\frac{1}{4\pi\epsilon_0} \frac{q_1 q_2}{|\vec{x}_1 - \vec{x}_2|^2} \hat{u}$ 
...
385     * -  $q_1$  is the the first particle's charge. (C)
386     * -  $q_2$  is the the second particle's charge. (C)
387     * -  $\vec{x}_1$  is the position of the first particle. (m)
388     * -  $\vec{x}_2$  is the position of the second particle. (m)
...
389     * -  $\epsilon_0$  is the permittivity of free space. (F/m)
390     * -  $\hat{u}$  is the unit vector pointing from the first particle
391     * away from the second particle.
392     */
393     class ShieldedIon2
394     {
395     public:
396
397         //! Force Calculation
398         /*!
399         * Perform the calculation specified by the Force. For more
...
400         * information
...
401         * about the contract provided by operator() please see the \a
...
402         * Interaction
403         * \a Force concept documentation.
404         *
405         * \param[in] p1 The initial state of the first particle.
406         * \param[in] p2 The initial state of the second particle.
407         * \param[in] t The current time.
408         * \return The force on the first particle.
409         */
410         Spatial_Vector operator()( const Dust_State& p1, const Dust_State&
...
411         p2, const double t)
412         {
413             // distance between the two particles and the wake around dust
...
414             2
415             float dist = abs(p1.pos - p2.pos);
416             float x = fabs(p1.pos[0] - p2.pos[0]);
417             Spatial_Vector r_hat = (p1.pos - p2.pos) / dist;
418
419             // calculates the unit vectored force (ignoring constants)
420             Spatial_Vector fShieldedIon2 =
421             exp( - dist / debye ) * r_hat * ( 1 / ( dist * dist ) + 1 / (
...
422             dist * debye ) )
423             * ( 1 - powf( sin( beta * x ), 2.0 ) * ( exp( - alpha * x )
...
424             + exp( alpha * x ) ) )
```

```
420
421
422     + ( exp( - dist / debye ) / dist ) * wake_dir
423     * ( 2 * beta * cos( beta * x ) * sin( beta * x ) * ( exp(
... - alpha * x ) + exp( alpha * x ))
424     + alpha * powf( sin( beta * x ), 2.0 ) * ( - exp( -
... alpha * x ) + exp ( alpha * x )));
425
426     return multiplier * p1.charge * p2.charge * fShieldedIon2;
427 }
428
429     //!< Set the permittivity of free space. (\f$ F/m \f$)
430     //!< In addition to setting epsilon_0, this function also calculates
... and
431     //!< caches the constant fraction used by operator().
432     void set_epsilon_0( float epsilon_0 )
433     {
434         multiplier = 1 / ( 4 * epsilon_0 * 3.141592653589793238463);
435     }
436
437     //!< Set the debye length. (\f$m\f$)
438     void set_debye( float debye_ ) { debye = debye_; }
439     //!< Set the exponential decay constant.
440     void set_alpha( float alpha_ ) { alpha = alpha_; }
441     //!< Set the sine coefficient.
442     void set_beta( float beta_ ) { beta = beta_; }
443     //!< Set the wake direction
444     void set_wake_dir( Spatial_Vector wake_dir_ ) { wake_dir =
... wake_dir_; }
445
446     private:
447         float multiplier;
448         float debye;
449         float alpha;
450         float beta;
451         Spatial_Vector wake_dir;
452
453 };
454
455     //!< ShieldedIon3 (Interaction Force)
456     /*!
457     * ShieldedIon3 follows the \a Interaction \a Force concept. The force
... results
458     * from interparticle coulomb potentials. For a single particle pair:
459     *
460     * 
$$\vec{F}_1 = -\vec{F}_2 =$$

461     * 
$$\frac{1}{4\pi\epsilon_0} \frac{q_1$$

... 
$$q_2}{|\vec{x}_1 - \vec{x}_2|^2} \hat{u}$$

```



```
462     * - \f$q_1\f$ is the the first particle's charge. (\f$ C \f$)
463     * - \f$q_2\f$ is the the second particle's charge. (\f$ C \f$)
464     * - \f$\vec{x}_1\f$ is the position of the first particle.(\f$ m \f$)
465     * - \f$\vec{x}_2\f$ is the position of the second particle. (\f$ m
... \f$)
466     * - \f$\epsilon_0\f$ is the permittivity of free space. (\f$ F/m \f$)
467     * - \f$\hat{u}\f$ is the unit vector pointing from the first particle
468     *     away from the second particle.
469     */
470     class ShieldedIon3
471     {
472     public:
473
474         //! Force Calculation
475         /*!
476         * Perform the calculation specified by the Force. For more
... information
477         * about the contract provided by operator() please see the \a
... Interaction
478         * \a Force concept documentation.
479         *
480         * \param[in] p1 The initial state of the first particle.
481         * \param[in] p2 The initial state of the second particle.
482         * \param[in] t The current time.
483         * \return The force on the first particle.
484         */
485         Spatial_Vector operator()( const Dust_State& p1, const Dust_State&
... p2, const double t)
486         {
487             // distance between the two particles and the wake around dust
... 2
488             float dist = abs(p1.pos - p2.pos);
489             float x = fabs(p1.pos[0] - p2.pos[0]);
490             float y = fabs(p1.pos[1] - p2.pos[1]);
491             Spatial_Vector r_hat = (p1.pos - p2.pos) / dist;
492
493             //Calculates the wake terms
494             float wake_terms_x = 2 * beta * cos( beta * x ) * ( exp( -
... alpha * x ) + exp( alpha * x ) )
495             + alpha * sin( beta * x ) * ( - exp( -
... alpha * x ) + exp( alpha * x ) );
496             float wake_terms_y = 2 * gamma * y * ( exp( - alpha * x ) +
... exp( alpha * x ) );
497
498             Spatial_Vector wake_terms = { wake_terms_x, wake_terms_y, 0 };
499
500             // calculates the unit vectored force (ignoring constants)
501             Spatial_Vector fShieldedIon3 =
```



```
502         exp( - dist / debye ) * r_hat * ( 1 / ( dist * dist ) + 1 / (
... dist * debye ) )
503         * ( 1 - powf( sin( beta * x ), 2.0 ) * ( exp( - alpha * x ) +
... exp( alpha * x ) ) )
504         -exp( - dist / debye ) * sin( beta * x ) * exp( - gamma * powf(
... y , 2.0 ) ) * wake_terms;
505
506         return multiplier * p1.charge * p2.charge * fShieldedIon3;
507     }
508
509     //!< Set the permittivity of free space. (\f$ F/m \f$)
510     //!< In addition to setting epsilon_0, this function also calculates
... and
511     //!< caches the constant fraction used by operator().
512     void set_epsilon_0( float epsilon_0 )
513     {
514         multiplier = 1 / ( 4 * epsilon_0 * 3.141592653589793238463 );
515     }
516
517     //!< Set the debye length. (\f$m\f$)
518     void set_debye( float debye_ ) { debye = debye_; }
519     //!< Set the x- exponential decay constant.
520     void set_alpha( float alpha_ ) { alpha = alpha_; }
521     //!< Set the sine coefficient.
522     void set_beta( float beta_ ) { beta = beta_; }
523     //!< Set the x- exponential decay constant.
524     void set_gamma( float gamma_ ) { gamma = gamma_; }
525
526     private:
527         float multiplier;
528         float debye;
529         float alpha;
530         float beta;
531         float gamma;
532
533     };
534
535 } // namespace yoakum
536 #endif // include guard
537
```

### B.1.5. Dust\_state.hpp

Creates the state (variables describing a system) documentation for the dust cloud. The number of dimensions for cloud on line 15.

```
/Users/Loriscott/Documents/Yoakum/Dust_State.hpp Page 1/3  
Printed: 6/14/22, 5:06:56 PM Printed for: Lori Scott  


---

1 #ifndef YOAKUM_DUST_STATE_INCLUDED  
2 #define YOAKUM_DUST_STATE_INCLUDED  
3  
4 // Used by Dust_State to extend operator definitions  
5 #include <boost/operators.hpp>  
6 // Used to print the contents of a Dust_State with operator <<  
7 #include <ostream>  
8 // Vector used by Dust_State  
9 #include "Vector.hpp"  
10  
11 namespace yoakum {  
12  
13 // The number of spacial dimensions in the simulation  
14 // Should be 1, 2, or 3  
15 const size_t DIM = 3;  
16  
17 // A type for spatial vector values such as dust position  
18 // or E/B field  
19 using Spatial_Vector = Vector< float, DIM >;  
20  
21 /*  
22 * The state of a single dust particle  
23 */  
24 class Dust_State :  
25     boost::additive1< Dust_State ,  
26     boost::additive2< Dust_State , float ,  
27     boost::multiplicative2< Dust_State , float  
28     > > >  
29 {  
30 public:  
31  
32     // The position of the dust particle  
33     Spatial_Vector pos;  
34     // The velocity of the dust particle  
35     Spatial_Vector vel;  
36     // The charge on the dust particle  
37     float charge;  
38  
39     // Sets all data members to 0  
40     void clear() {  
41         pos = 0;  
42         vel = 0;  
43         charge = 0;  
44     }  
45  
46     // Add the data members of two Dust_States component wise  
47     Dust_State& operator+=( const Dust_State& other )  
48     {
```

```
49     pos += other.pos;
50     vel += other.vel;
51     charge += other.charge;
52     return *this;
53 }
54
55 // Take the difference of the data members of two Dust_States
56 // component wise
57 Dust_State& operator-=( const Dust_State& other )
58 {
59     pos -= other.pos;
60     vel -= other.vel;
61     charge -= other.charge;
62     return *this;
63 }
64
65 // Multiply the data members of Dust_State by val
66 Dust_State& operator*=( const float& val )
67 {
68     pos *= val;
69     vel *= val;
70     charge *= val;
71     return *this;
72 }
73
74 // Devide the data members of Dust_State by val
75 Dust_State& operator/=( const float& val )
76 {
77     pos /= val;
78     vel /= val;
79     charge /= val;
80     return *this;
81 }
82
83 };
84
85 // Negate all of the data members of Dust_State
86 Dust_State operator-( const Dust_State& other )
87 {
88     Dust_State tmp;
89     tmp.pos = -other.pos;
90     tmp.vel = -other.vel;
91     tmp.charge = -other.charge;
92     return tmp;
93 }
94
95 // Print all of the data members of Dust_State to out as a comma seperated
... list
```

```
96 // The list starts with ", "  
97 std::ostream& operator<<( std::ostream &out , const Dust_State& v )  
98 {  
99     out << v.pos << v.vel << ", " << v.charge;  
100     return out;  
101 }  
102  
103 } // namespace yoakum  
104 #endif // include guard  
105
```

## B.1.6. Io\_observers.hpp

The function that prints the data out to a write file.

```

/Users/Loriscott/Documents/Yoakum/io_observers.hpp
Printed: 6/14/22, 5:07:24 PM
Page 1/1
Printed for: Lori Scott
1 #ifndef YOAKUM_IO_OBSERVERS_INCLUDED
2 #define YOAKUM_IO_OBSERVERS_INCLUDED
3
4 // Used by Streaming_Observer to print output
5 #include <iostream>
6 // Needed for NUM_DUST
7 #include "configure.hpp"
8
9 namespace yoakum {
10 /*
11 * Functor for printing a state to out
12 */
13 struct Streaming_Observer
14 {
15     // The stream Streaming_Observer prints to
16     std::ostream& out;
17
18     // Constructor
19     Streaming_Observer( std::ostream &out_ ) : out( out_ ) {
20         out.precision(17);
21     }
22
23     // Prints a State and Time to out
24     template< typename State >
25     void operator()( const State& state , double t ) const
26     {
27         //added if statement when running LARGE files
28         if( fmod(t_counter, print_mod) == 0 )
29         {
30             // print the time
31             out << t;
32             // print the state for each dust particle
33             for( size_t i=0 ; i<NUM_DUST ; i++ ) out << state[i];
34             out << '\n';
35             t_counter++;
36         }
37         else
38         {
39             t_counter++;
40         }
41     }
42 };
43
44 } // namespace yoakum
45 #endif // include guard
46
47
```

### B.1.7. Configure.hpp

The configuration of the dust cloud file. Number of particles is on line 12; print\_mod which is how frequently to write out data to the .dat file is on line 25.

```

/Users/Loriscott/Documents/Yoakum/configure.hpp
Printed: 6/14/22, 5:06:06 PM
Page 1/1
Printed for: Lori Scott
1 #ifndef YOAKUM_CONFIGURE_INCLUDED
2 #define YOAKUM_CONFIGURE_INCLUDED
3
4 // Used by Plasma_State
5 #include <array>
6 // Used by Plasma_State
7 #include "Dust_State.hpp"
8
9 namespace yoakum {
10
11 // The number of dust particles
12 const size_t NUM_DUST = 5000;
13
14 // Represents the state of the plasma as a collection of dust states
15 using Plasma_State = std::array< Dust_State, NUM_DUST >;
16
17 // A scalar for each dust particle
18 using Scalar_Group = std::array< float, NUM_DUST >;
19
20 // Gravitational constant ( units )
21 const double GRAVITY_CONST = 2.95912208286e-4;
22
23 //print less info
24 double t_counter = 0;
25 const double print_mod = 50;
26
27 } // namespace yoakum
28 #endif // include guard
29
```



## B.1.8. System.hpp

The file that creates the system of ODEs that is solved each timestep.

```
/Users/Loriscott/Documents/Yoakum/System.hpp Page 1/1  
Printed: 6/14/22, 5:07:45 PM Printed for: Lori Scott
```

---

```
1 #ifndef YOAKUM_SYSTEM_INCLUDED  
2 #define YOAKUM_SYSTEM_INCLUDED  
3  
4 // System uses NUM_DUST and Plasma_State  
5 #include "configure.hpp"  
6  
7 namespace yoakum {  
8  
9 /*  
10 * System  
11 *  
12 * Represents a system of ODEs that governs the motion of dust particles.  
13 *  
14 * Operator() uses the system of ODEs to calculate a delta state (dsdt)  
15 * from an initial state.  
16 *  
17 * The ODE  $x' = v$  is automatically included in a call to operator().  
18 * Additional ODEs are added by inheriting from functors. When operator()  
19 * is called, it calls operator() for each base class.  
20 */  
21 template< typename... Bases >  
22 class System : public Bases...  
23 {  
24 public:  
25  
26     // Determine a delta state (dsdt) from a given initial state (state) at  
27     // time t  
28     // using the given set of ODEs.  
29     void operator()( const Plasma_State& state, Plasma_State& dsdt, const  
30     // double t )  
31     {  
32         // Set dsdt to 0  
33         for( size_t i=0 ; i<NUM_DUST; i++ ) dsdt[i].clear();  
34  
35         // call operator() for each base class  
36         (Bases::operator()( state, dsdt, t ), ...);  
37  
38         // d( position )dt = velocity  
39         for( size_t i=0 ; i<NUM_DUST ; i++ )  
40         {  
41             dsdt[i].pos = state[i].vel;  
42         }  
43     }  
44 };  
45 } // namespace yoakum  
46 #endif // include guard
```

## B.1.9. Vector.hpp

The file that defines vector mathematics for the code.

```
/Users/Loriscott/Documents/Yoakum/Vector.hpp Page 1/8  
Printed: 6/14/22, 5:07:57 PM Printed for: Lori Scott
```

---

```
1 #ifndef YOAKUM_VECTOR_INCLUDED  
2 #define YOAKUM_VECTOR_INCLUDED  
3  
4 /*!  
5  * \file Vector.hpp  
6  *  
7  * \author Dustin Sanford  
8  * \date July 16, 2018  
9  *  
10 * Contains the Vector class and free function definitions.  
11 */  
12  
13 // Used by Vector to extend operator definitions.  
14 #include <boost/operators.hpp>  
15 // Used to print the contents of a Vector in operator<<.  
16 #include <ostream>  
17 // Used by a Vector constructor.  
18 #include <initializer_list>  
19  
20 namespace yoakum {  
21  
22 //! Container for vector values  
23 /*!  
24  * Provides support for vector arithmetic and semantics. The container  
25  * is templated on a component type \a T and the number of dimensions  
26  * \a Dim. Vector can contain any type (T) that is copy assignable and  
27  * supports the operators +,-,*,/, and <<. A dimensionality (Dim) less  
28  * than one will result in undefined behavior.  
29  *  
30  * Each element in the Vector is referred to as a component. Each  
31  * component  
32  * is of type \a T.  
33  *  
34  * \note Vector inherits from several classes in boost::operators. This  
35  * extends several arithmetic compound assignment operator definitions to  
36  * binary arithmetic operators. For more information see the  
37  * boost::operator  
38  * documentation.  
39  */  
40 template< class T , size_t Dim >  
41 class Vector :  
42     boost::additive1< Vector< T , Dim > ,  
43     boost::additive2< Vector< T , Dim > , T ,  
44     boost::multiplicative2< Vector< T , Dim > , T  
45     > > >  
46 {  
47 public:
```



```
47     //! Default Constructor
48     /*!
49     * Set all vector components to zero.
50     */
51     Vector( void )
52     {
53         for( size_t i=0 ; i<Dim ; ++i ) data[i] = 0.0;
54     }
55
56     //! Single Value Constructor
57     /*!
58     * Set all Vector components to a specified value.
59     *
60     * \param[in] data_ All Vector components are set to this value.
61     */
62     Vector( T data_ )
63     {
64         for( size_t i=0 ; i<Dim ; ++i ) data[i] = data_;
65     }
66
67     //! Full Specification Constructor
68     /*!
69     * Set each Vector component individually.
70     *
71     * \param[in] data_ Each element sets a different vector component.
72     * The number of elements in \a data_ must correspond to the
73     * dimensionality of the Vector.
74     *
75     * \warning An exception is thrown if the number of components
76     * in data_ does not match the dimensionality of the vector.
77     */
78     Vector( std::initializer_list< T > data_ )
79     {
80         // Check that the initializer list has the correct number of
81         ... elements
82         if( data_.size() == Dim ) {
83             // get an iterator for data_
84             typename std::initializer_list< T >::iterator it =
85             ... data_.begin();
86             // loop through the elements in data_ and assign them to data
87             for( size_t i=0 ; i<Dim ; ++i ) data[i] = *it++;
88             } else {
89                 throw;
90             }
91         }
92
93     //! Const Subscript Operator
94     /*!
```

```
93     * \param[in] i The index of the component to return.
94     * \return A copy of the ith Vector component.
95     *
96     * \warning No bounds checking is performed.
97     */
98     T operator[]( size_t i ) const { return data[i]; }
99
100    //! Subscript Operator
101    /*!
102     * \param[in] i The index of the component to return.
103     * \return A reference to the ith Vector component.
104     *
105     * \warning No bounds checking is performed.
106     */
107    T& operator[]( size_t i ) { return data[i]; }
108
109    //! Vector Addition Operator(s)
110    /*!
111     * Add two Vectors component wise. The binary + operator is supported
112     * through boost::operators.
113     *
114     * \param[in] other The vector to add to \a this.
115     */
116    Vector< T, Dim >& operator+=( const Vector< T, Dim >& other )
117    {
118        for( size_t i=0 ; i<Dim ; ++i ) data[i] += other[i];
119        return *this;
120    }
121
122    //! Vector Subtraction Operator(s)
123    /*!
124     * Take the difference of two Vectors component wise. The binary -
125     * operator is supported through boost::operators.
126     *
127     * \param[in] other The vector to subtract from \a this.
128     */
129    Vector< T, Dim >& operator-=( const Vector< T, Dim >& other )
130    {
131        for( size_t i=0 ; i<Dim ; ++i ) data[i] -= other[i];
132        return *this;
133    }
134
135    //! Compound Vector "Multiplication" Assignment Operator
136    /*!
137     * Multiply two vectors component wise.
138     *
139     * \param[in] other The vector to multiply \a this with.
140     */
```

```
141     Vector< T, Dim >& operator*=( const Vector< T, Dim >& other )
142     {
143         for( size_t i=0 ; i<Dim ; i++ ) data[i] *= other[i];
144         return *this;
145     }
146
147     //! Vector "Multiplication" Operator
148     /*!
149     * Multiply two vectors component wise.
150     *
151     * \param[in] v1 The left Vector operand.
152     * \param[in] v2 The right Vector operand.
153     * \return The result of the component wise multiplication.
154     */
155     friend Vector< T, Dim > operator*( const Vector< T, Dim >& v1, const
... Vector< T, Dim >& v2)
156     {
157         Vector< T, Dim > out = v1;
158         return out *= v2;
159     }
160
161     //! Compound Vector "Devision" Assignment Operator
162     /*!
163     * Divide two vectors component wise.
164     *
165     * \param[in] other The vector to Divide \a this with.
166     */
167     Vector< T, Dim >& operator/=( const Vector< T, Dim >& other )
168     {
169         for( size_t i=0 ; i<Dim ; i++ ) data[i] /= other[i];
170         return *this;
171     }
172
173     //! Vector "Devision" Operator
174     /*!
175     * Divide two Vectors component wise.
176     *
177     * \param[in] v1 The Vector in the numerator.
178     * \param[in] v2 The Vector in the denominator.
179     * \return The result of v1 divided by v2 component wise.
180     */
181     friend Vector< T, Dim > operator/( const Vector< T, Dim >& v1, const
... Vector< T, Dim >& v2)
182     {
183         Vector< T, Dim > out = v1;
184         return out /= v2;
185     }
186
```

```
187     //! Scalar addition Operator(s)
188     /*!
189     * Add a scaler to each vector component. The binary + operator
190     * is supported through boost::operators.
191     *
192     * \param[in] val The scaler to add to \a this.
193     */
194     Vector< T, Dim >& operator+=( const T& val )
195     {
196         for( size_t i=0 ; i<Dim ; ++i ) data[i] += val;
197         return *this;
198     }
199
200     //! Scalar Subtraction Operator(s)
201     /*!
202     * Subtract a scaler value from each vector component. The binary -
203     ... operator
204     * is supported through boost::operators.
205     *
206     * \param[in] val The scaler to subtract from \a this.
207     */
208     Vector< T, Dim >& operator-=( const T& val )
209     {
210         for( size_t i=0 ; i<Dim ; ++i ) data[i] -= val;
211         return *this;
212     }
213
214     //! Scalar Multiplication Operator(s)
215     /*!
216     * Multiply each Vector component by a scaler value. The
217     * binary * operator is supported through boost::operators.
218     *
219     * \param[in] val The scaler to multiply \a this by.
220     */
221     Vector< T, Dim >& operator*=( const T& val )
222     {
223         for( size_t i=0 ; i<Dim ; ++i ) data[i] *= val;
224         return *this;
225     }
226
227     //! Scalar Devisision Operator(s)
228     /*!
229     * Divide each vector component by a scaler value. The binary /
230     ... operator
231     * is supported through boost::operators.
232     *
233     * \param[in] val The scaler to divide \a this by.
234     *
```

```
233     * \warning Devision of a scaler by a vector is not supported.
234     */
235     Vector< T, Dim >& operator/=( const T& val )
236     {
237         for( size_t i=0 ; i<Dim ; ++i ) data[i] /= val;
238         return *this;
239     }
240
241 private:
242     /*! The components of the vector.
243     T data[Dim];
244     */;
245
246
247     /*! Negate each element of a Vector.
248     */
249     * \relates Vector
250     *
251     * \param[in] v The Vector to negate.
252     * \return A negated copy of \a v.
253     */
254     template< class T , size_t Dim >
255     Vector< T , Dim > operator-( const Vector< T , Dim >& v )
256     {
257         Vector< T , Dim > tmp;
258         for( size_t i=0 ; i<Dim ; ++i ) tmp[i] = -v[i];
259         return tmp;
260     }
261
262     /*! Take the dot product of two vectors.
263     */
264     * \relates Vector
265     *
266     * \param[in] v1 The left Vector operand.
267     * \param[in] v2 The right Vector operand.
268     * \return The dot product of the two vectors.
269     */
270     template< class T , size_t Dim >
271     T dot( const Vector< T , Dim > &v1 , const Vector< T , Dim > &v2 )
272     {
273         T tmp = 0.0;
274         for( size_t i=0 ; i<Dim ; ++i ) tmp += v2[i] * v1[i];
275         return tmp;
276     }
277
278     /*! Take the norm of a Vector
279     */
280     * \relates Vector
```

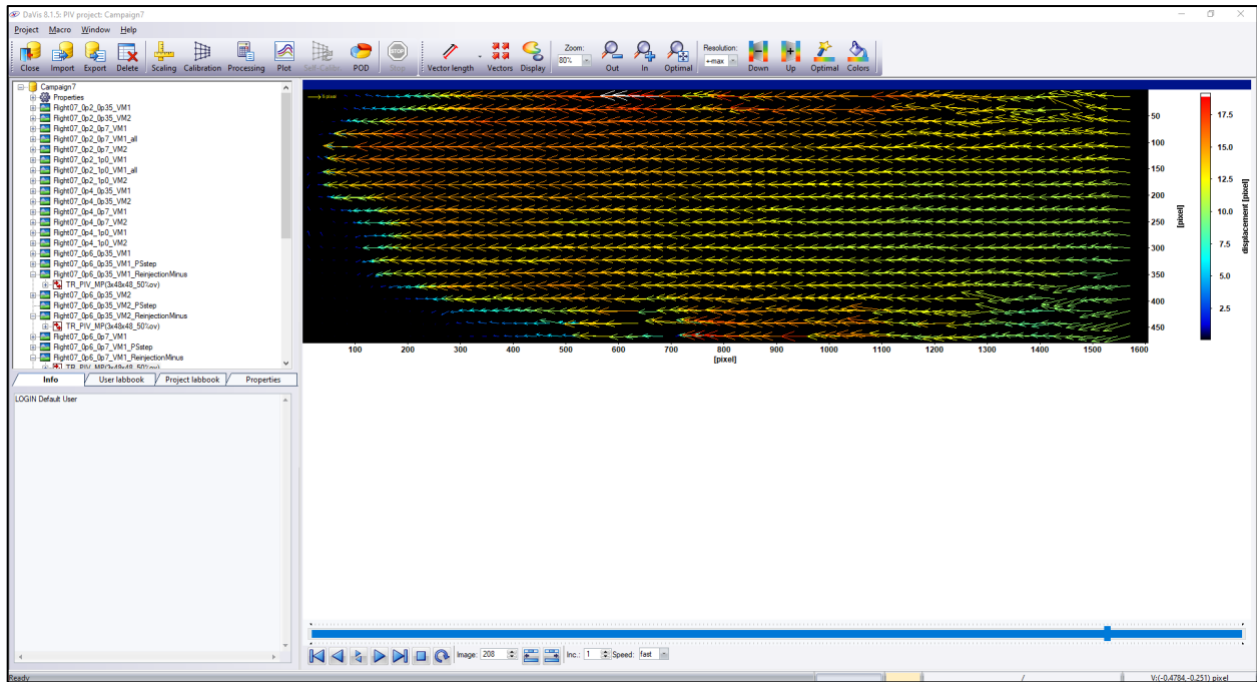


```
281 * Take the dot product of a Vector with itself.
282 *
283 * \param[in] Vector and itself.
284 * \return The normal of the Vector.
285 */
286 template< class T , size_t Dim >
287 T norm( const Vector< T , Dim >& v )
288 {
289     return dot( v , v );
290 }
291
292 //! Get the length of a Vector.
293 /*!
294 * \relates Vector
295 * Take the square root of the norm() of a Vector.
296 *
297 * \param[in] v The Vector to calculate the length of.
298 * \return The length of the Vector.
299 */
300 template< class T , size_t Dim >
301 T abs( const Vector< T , Dim >& v )
302 {
303     return sqrt( norm( v ) );
304 }
305
306 //! Vector Cross Product
307 /*!
308 * \relates Vector
309 * Take the cross product of two Vectors.
310 *
311 * \param[in] lhs The left hand Vector operand.
312 * \param[in] rhs The right hand Vector operand.
313 * \return The cross product of the two Vectors.
314 *
315 * \warning Only supports 3D vectors.
316 */
317 template< class T>
318 Vector< T , 3 > cross( const Vector< T , 3>& lhs, const Vector< T , 3>& rhs )
319 {
320     Vector< T , 3 > out;
321     out[0] = lhs[1] * rhs[2] - lhs[2] * rhs[1];
322     out[1] = lhs[2] * rhs[0] - lhs[0] * rhs[2];
323     out[2] = lhs[0] * rhs[1] - lhs[1] * rhs[0];
324     return out;
325 }
326
327 //! Vector Stream Operator
328 /*!
```

```
329 * \relates Vector
330 * Prints the contents of a Vector as a comma separated list.
331 *
332 * \param[in] out The ostream to print to.
333 * \param[in] v The Vector to print.
334 * \return A reference to the ostream is returned.
335 *
336 * \note The output is started with ", "
337 */
338 template< class T , size_t Dim >
339 std::ostream& operator<<( std::ostream &out , const Vector< T , Dim > &v )
340 {
341     for( size_t i=0 ; i<Dim ; ++i ) out << ", " << v[i];
342     return out;
343 }
344
345 } // namespace yoakum
346 #endif // include guard
347
```

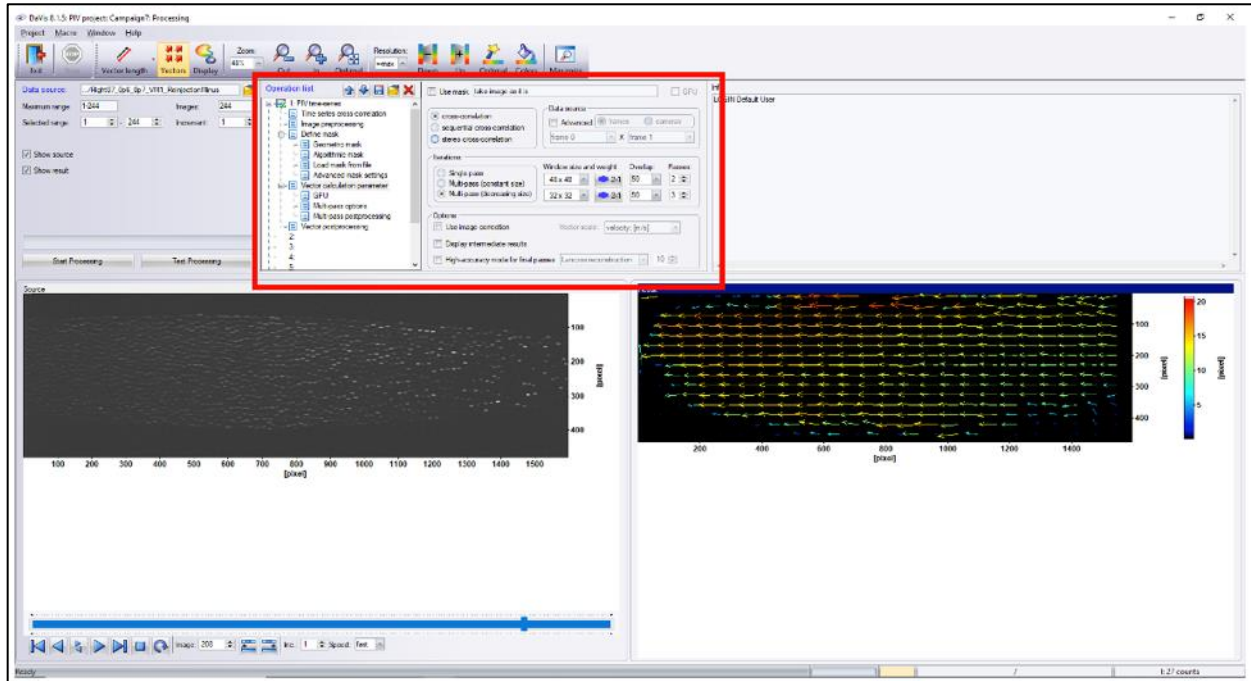
## B.2. PIV Settings

This section shows screenshots of the DaVis PIV software, to help guide future users using this analysis technique. Descriptions of each figure and important settings are mentioned in the captions.

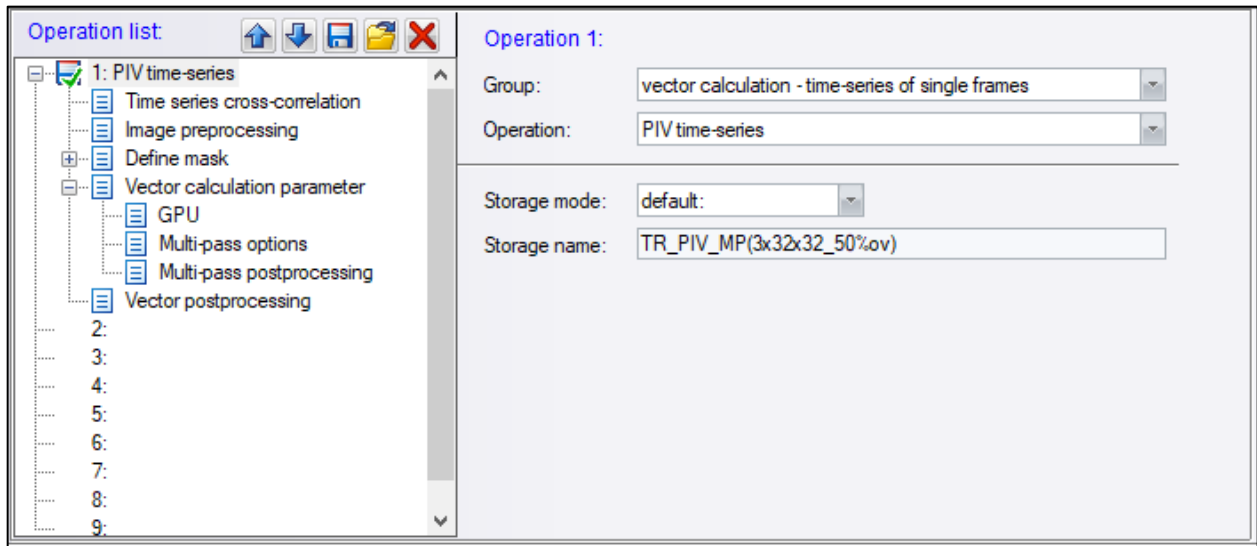


Appendix Figure 1: Home screen of the DaVis project. Important buttons on the toolbar are “import”, “export”, “processing” and “resolution” is usually set to ~128 when viewing PK-4 images (for brightness).

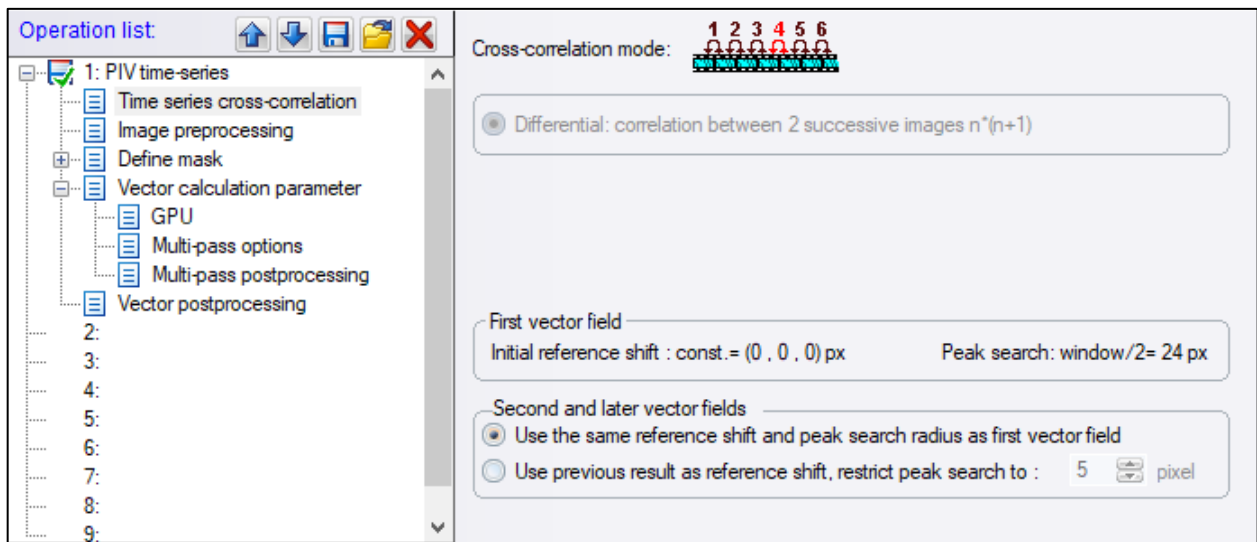




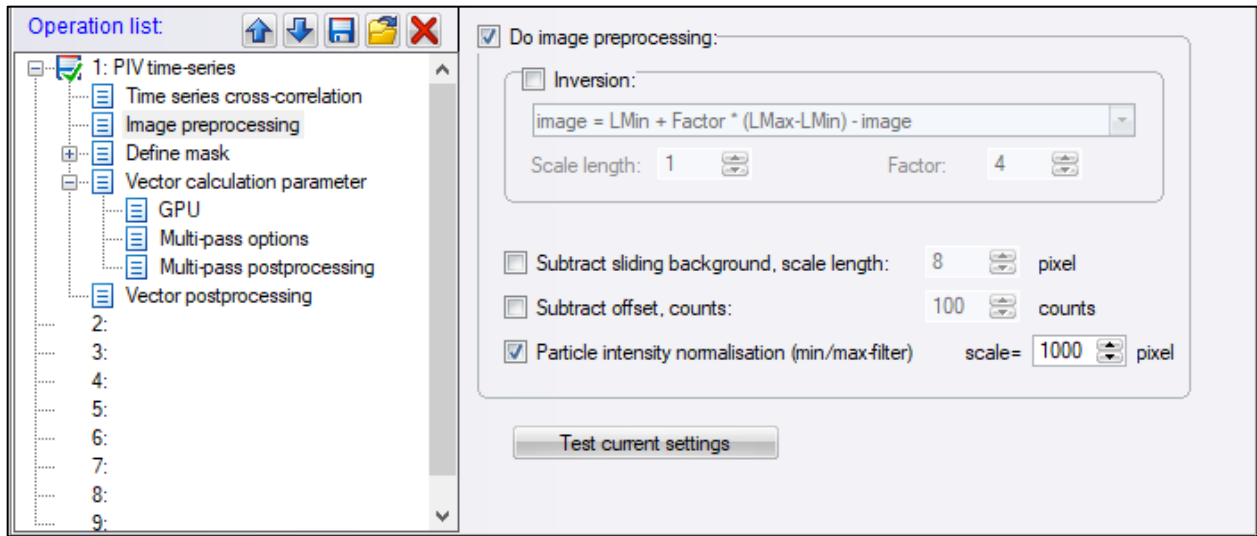
**Appendix Figure 2: Vector processing home screen. All settings discussed in Section 2.2.1 are found in the red box, and zoomed in screenshots of each tab in this area are shown below. The “show source” and “show result” checkbox on the left trigger the views below. This is good for viewing the “test processing” but for efficiency, should be turned off when processing the entire video.**



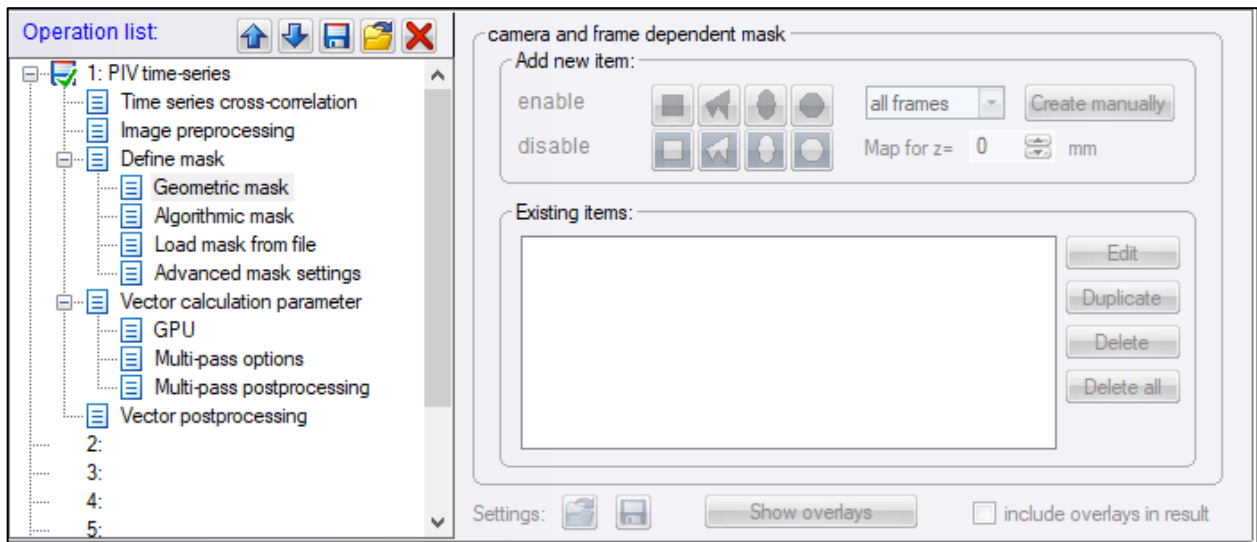
Appendix Figure 3: PIV time-series operation.



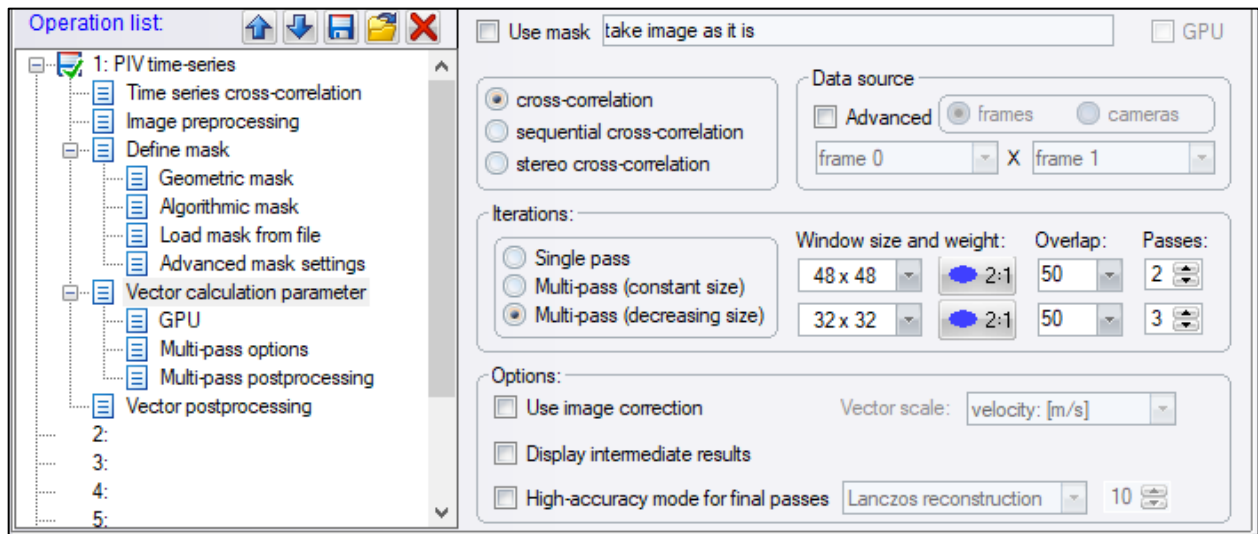
Appendix Figure 4: Time series cross-correlation.



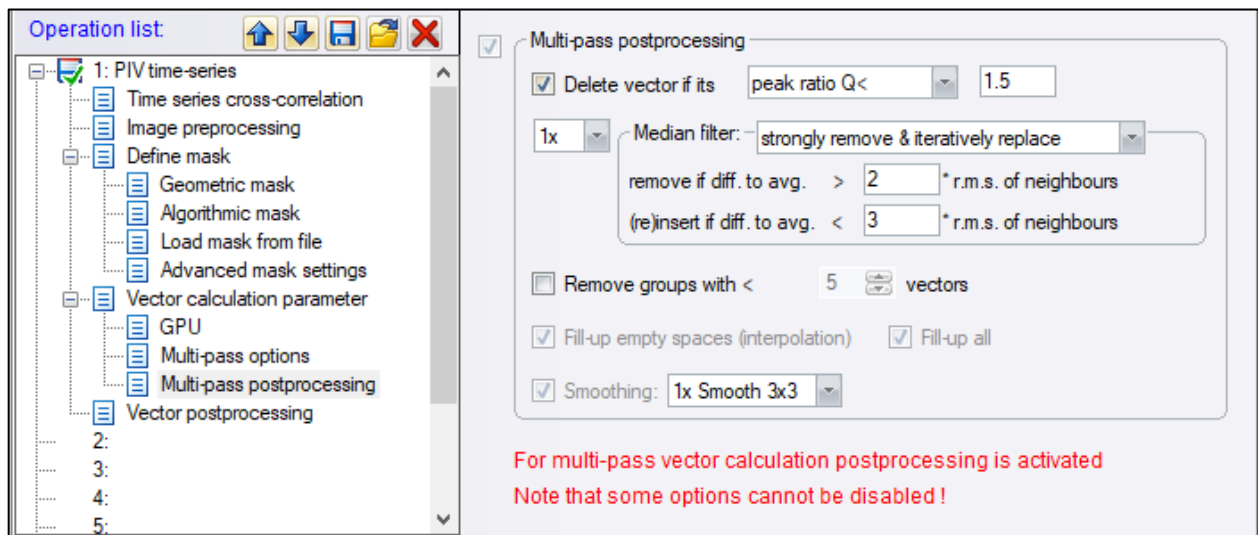
Appendix Figure 5: Image preprocessing settings.



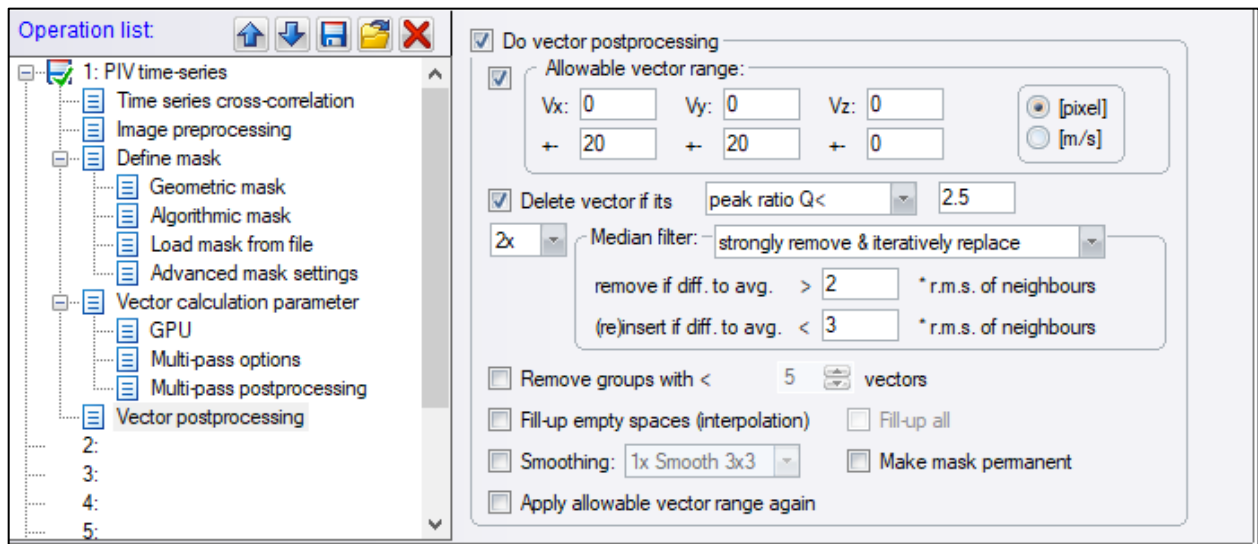
Appendix Figure 6: Geometric mask. This can be used to exclude sections of the frame from the PIV analysis.



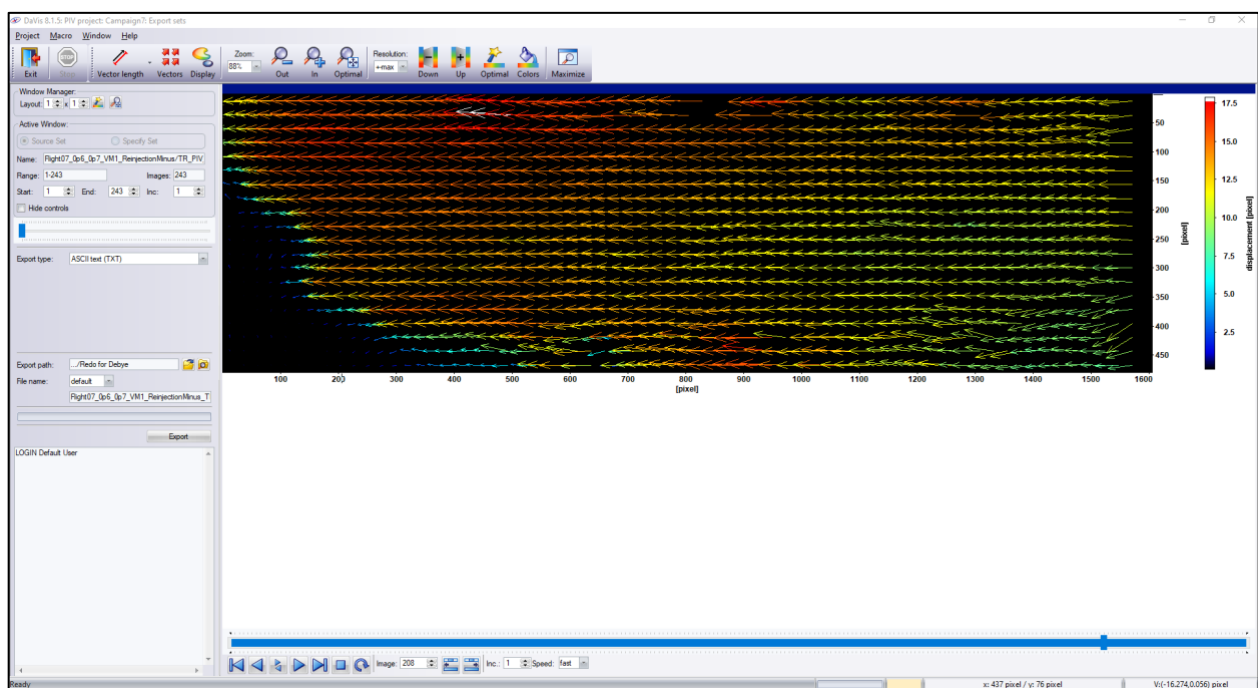
Appendix Figure 7: Vector calculation parameters. This is where the multi-pass settings are located.



Appendix Figure 8: Multi-pass postprocessing.



Appendix Figure 9: Vector postprocessing.



Appendix Figure 10: Export window. The frames are exported as “.txt” files, as (x,y) and (v<sub>x</sub>,v<sub>y</sub>) coordinates. There are also image and video options we use for presentations.

### B.3. MATLAB- Processing Codes

These are the various MATLAB codes, functions, and scripts I use to import, process, and export data. Some are used for the experimental data in Chapter 3, but most are used for YOAK $\mu$ M data in Chapter 4.

#### B.3.1. Camera Sample Data

To Camera Sample the YOAK $\mu$ M data, usually at 70 fps. This uses the instantaneous velocity for the drift velocity, and the average over the datapoints within a frame for the temperature.

7/26/22 2:41 PM /Users/lo.../camera\_sample\_data\_both.m 1 of 1

```
function [time_new, pos_new, vel_new_avg, vel_new_inst] = camera_sample_data_both(
    (fps,time,pos,vel)
% samples the yoakum data for a camera frame rate
%
%% Info
data_timestep = time(2,:) - time(1,:);
avg_size = 1 / data_timestep / fps;
temp = 1:avg_size:size(time);
avg_index = round(temp)';

%% Sample
% initialize
pos_new = zeros(length(avg_index),size(pos,2),size(pos,3));
vel_new_avg = zeros(length(avg_index),size(vel,2),size(vel,3));
vel_new_inst = zeros(length(avg_index),size(vel,2),size(vel,3));

% initial step
time_new = time(avg_index);
pos_new = pos(1,:,:);
vel_new_avg = vel(1,:,:);
vel_new_inst = vel(1,:,:);

for i = 2:length(avg_index)
    pos_new(i,:,:)= mean(pos((avg_index(i-1)+1):avg_index(i),,:,:));
    vel_new_inst(i,:,:)= vel(avg_index(i),,:,:);
    vel_new_avg(i,:,:)= mean(vel(avg_index(i-1):avg_index(i),,:,:));
end % for loop

end % function
```

### B.3.2. Extract Data

For cutting off heating segment as a new input file for the settling. Manually pick the index value of the desired Tx value and input it here.

7/26/22 2:45 PM /Users/loriscott/Doc.../extract\_data.m 1 of 1

```
%% Data extractor and write new file
% Use for YOAKUM heating event, Lori PhD data 2022

% this assumes the first column is the time then the following columns are
% the dust particles where each dust particle data order is:
% <x-pos, y-pos, z-pos, x-vel, y-vel, z-vel, charge>

% filename = P0p6_I0p7_N1000_2D_HeatingCutoff.dat % example for mass data
% processing naming scheme

function extract_data(maxindex, filename, time, pos, vel, charge)

% initialize data
i = maxindex; %time value to export
n = size(pos,2); %number of particles
L = zeros(i,n*7+1);

for j = 1:i % loop over time if wanted for plotting
    L(j,1) = time(j);

    for k = 1:n %loop over # particles
        % P = squeeze(pos(j,k,:))';
        % V = squeeze(vel(j,k,:))';
        % C = squeeze(charge(j,k))';
        % L(j) = horzcat(P,V,C); %can use same variable in horzcat

        L(j,((k-1)*7+2:(k-1)*7+4)) = squeeze(pos(j,k,:))';
        L(j,((k-1)*7+5:(k-1)*7+7)) = squeeze(vel(j,k,:))';
        L(j,((k-1)*7+8)) = squeeze(charge(j,k))';
    end %for loop over number particles

end %for loop over time

%write data to file
writematrix(L, filename)
```

### B.3.3. Get Data

For parsing YOAK $\mu$ M .dat file into time/position/velocity/charge MATLAB variables for all particles.

7/26/22 2:46 PM /Users/loriscott/Documen.../get\_data.m 1 of 1

```
%% Generic data splitter (YOAKUM)

% This assumes that the first column is the time then the following columns are
% the dust particles where each dust particle data order is:
% <x-pos, y-pos, z-pos, x-vel, y-vel, z-vel, charge>

function [time, pos, vel, charge] = get_data(file)

% Open the text file
data = importdata( file );

% Get the number of dust particles
num_dust = (size(data,2) - 1) / 7;

% split time off of data
time = data(:,1);
data = data(:,2:end);

% number of time steps
num_time_steps = length(time);

% create empty position and velocity matrices
pos = zeros(num_time_steps, num_dust, 3);
vel = zeros(num_time_steps, num_dust, 3);
charge = zeros(num_time_steps, num_dust, 1);

% populate the position and velocity
for i = 0:num_dust - 1
    % index of the first data point for the ith particle
    j = i * 7 + 1;
    pos(:,i+1,1) = data(:,j + 0);
    pos(:,i+1,2) = data(:,j + 1);
    pos(:,i+1,3) = data(:,j + 2);

    vel(:,i+1,1) = data(:,j + 3);
    vel(:,i+1,2) = data(:,j + 4);
    vel(:,i+1,3) = data(:,j + 5);

    charge(:,i+1,1) = data(:,j+6);
end
```



Alternatively, when the YOAK $\mu$ M datafiles are large, there is `get_data2`, which allows the user to skip steps for efficiency. This is especially useful when using the microsecond timesteps.

7/26/22 2:49 PM /Users/loriscott/Docume.../get\_data2.m 1 of 1

```
%% Generic data splitter (YOAKUM)

% This assumes that the first column is the time then the following columns are
% the dust particles where each dust particle data order is:
% <x-pos, y-pos, z-pos, x-vel, y-vel, z-vel, charge>

function [time, pos, vel, charge] = get_data2(file, skipstep)

% Open the text file
data = importdata( file );

% Get the number of dust particles
num_dust = (size(data,2) - 1) / 7;

% dt = data(2,1) - data(1,1);
% skipindex = min(data(:,1)):skipstep*dt:max(data(:,1));
%
% split time off of data
time = data(1:skipstep:end, 1);
data = data(1:skipstep:end, 2:end);

% number of time steps
num_time_steps = length(time);

% create empty position and velocity matrices
pos = zeros(num_time_steps, num_dust, 3);
vel = zeros(num_time_steps, num_dust, 3);
charge = zeros(num_time_steps, num_dust, 1);

% populate the position and velocity
for i = 0:num_dust - 1
    % index of the first data point for the ith particle
    j = i * 7 + 1;
    pos(:,i+1,1) = data(:,j + 0);
    pos(:,i+1,2) = data(:,j + 1);
    pos(:,i+1,3) = data(:,j + 2);

    vel(:,i+1,1) = data(:,j + 3);
    vel(:,i+1,2) = data(:,j + 4);
    vel(:,i+1,3) = data(:,j + 5);

    charge(:,i+1,1) = data(:,j+6);
end
```

### B.3.4. Single Fit Camera 2

The main source of YOAKUM processing- this code imports (from get\_data), camera sample if you so choose (using camera\_sample), fits to a Maxwell-Boltzmann distribution, calculates various values, and plots, for a specified data file. I frequently run, and then use “hold on” command in MATLAB to make the figures show segments altogether. The most common files’ paths are commented out at the top of the file in the import section.

```
7/26/22 2:52 PM /Users/loriscot.../single_fit_camera.m 1 of 4
```

```
%% import and camera sample data

% [time, pos, vel, charge] = get_data
(' /Users/loriscott/Downloads/Pp6_N1000_ThermBM1_Initial2.dat');
% [time, pos, vel, charge] = get_data2
(' /Users/loriscott/Downloads/P0p6_I0p7_N5000_2D_Y1p75_LONG_Inject_mus_updated.dat',
10);

[time, pos, vel, charge] = get_data('/Users/loriscott/Library/CloudStorage/Box-
Box/Dissertation/YOAKUM Files/P0p6_I0p7_N1000_2D_Heating_microdnt_nothem.dat');

disp('Data Imported')

%% Use this area to read in multiple files and combine for one processing
% [t1, pos1, vel1, charge1] = get_data
(' /Users/loriscott/Documents/Codes/YOAKUM/Pp6_Ip7_N1000_ThermBM1_Initial2.dat'); %
crystal
% [t2, pos2, vel2, charge2] = get_data
(' /Users/loriscott/Documents/Codes/YOAKUM/Pp6_Ip7_N1000_ThermBM1_Injection2.dat'); %
% R-
% [t3, pos3, vel3, charge3] = get_data(''); % injection/ R+
% [t4, pos4, vel4, charge4] = get_data(''); % "plasma change"
% [t5, pos5, vel5, charge5] = get_data(''); % PS
%
% % Concatenate- add );% where needed
% time = cat(1, t1, t2(2:end));% t3(2:end), t4(2:end), t5(2:end));
% pos = cat(1, pos1, pos2(2:end,:,:), pos3(2:end,:,:), pos4(2:end,:,:), pos5(2:
end,:,:));
% vel = cat(1, vel1, vel2(2:end,:,:), vel3(2:end,:,:), vel4(2:end,:,:), vel5(2:
end,:,:));
% charge = cat(1, charge1, charge2(2:end,:,:), charge3(2:end,:,:), charge4(2:
end,:,:), charge5(2:end,:,:));

%% Change variable names if not camera sampling
time_new = time;
pos_new = pos;
vel_new = vel;
charge_new = charge;

% [time_new, pos_new, vel_new_a, vel_new_i] = camera_sample_data_both(70,time,pos,
vel);
%
% % clear time pos vel

%% single use v_fit
frames = length(time_new);
num_particles = size(pos_new,2);

avg_all_camera = zeros(frames,3);
sig_all_camera = zeros(frames,3);
skew_all_camera = zeros(frames,3);
kurt_all_camera = zeros(frames,3);

vx_camera = vel_new(:, :,1)';
```

```

vy_camera = vel_new(:,:,2)';
vz_camera = vel_new(:,:,3)';

rdist = zeros(num_particles, num_particles, frames);

for t=1:frames
    %display counter
    % if(mod(d,1000)==0)
    %     disp(['GaussFit location ',num2str(d)])
    % end

    [p,~] = find(vx_camera(:,t)~=0);
    [q,~] = find(vy_camera(p,t)~=0);
    if(~isempty(q))

        [muX,sigX] = normfit(vx_camera(p(q),t));
        avg_all_camera(t,1) = muX;
        sig_all_camera(t,1) = sigX;
        skew_all_camera(t,1) = skewness(vx_camera(p(q),t));
        kurt_all_camera(t,1) = kurtosis(vx_camera(p(q),t));

        [muY,sigY] = normfit(vy_camera(p(q),t));
        avg_all_camera(t,2) = muY;
        sig_all_camera(t,2) = sigY;
        skew_all_camera(t,2) = skewness(vy_camera(p(q),t));
        kurt_all_camera(t,2) = kurtosis(vy_camera(p(q),t));

        [muZ,sigZ] = normfit(vz_camera(p(q),t));
        avg_all_camera(t,3) = muZ;
        sig_all_camera(t,3) = sigZ;
        skew_all_camera(t,3) = skewness(vz_camera(p(q),t));
        kurt_all_camera(t,3) = kurtosis(vz_camera(p(q),t));

    end

    if(exist('rdist','var') ~= 0) %if rdist is defined, run this stuff
        % copied from old OML_Box code!
        %need distance between each of the surface points
        POS1 = repmat(reshape(pos_new(t,:,:), [num_particles,1,3]), [1,num_particles,
1]);
        %each row of pos2 is a repeat of the first one.
        POS2 = repmat(reshape(pos_new(t,:,:), [1,num_particles,3]), [num_particles,
1,1]);
        %find the distance between every pt with all other pts
        rdist(:,:,t) = sqrt(sum((POS1-POS2).^2,3));
    end
end

temperature = 0.5 * 2.93027*10^-14 .* (sig_all_camera).^2 / (1.602*10^-19); % mass
[kg] from mathematica PK4 file- MF, 3.36 diam particles

clear POS1 POS2

```

```
%% Energy Calcs

PE = zeros(length(time_new),1);
KE = zeros(length(time_new),1);
lambda = 1.45E-3; %m
q_dust = 6000 * 1.602E-19; % C; 6000 * e-
mass = 2.93027*10^-14; % [kg]

for i = 1:length(time_new)
%   PE(i) = sum( triu( (q_dust^2 .* exp(-rdist(:,:,i)./lambda) ./ (4 * pi * 8.85E-12
%   E-12 .* rdist(:,:,i))) ,1),'all');
    KE(i) = sum( 1/2 .* mass .* sum(vel_new(i,:).^2,3) , 'all');
end

% PE = PE ./ 1.602E-19; % J to eV
% KE = KE ./ 1.602E-19; % J to eV

%% Plots
disp('Starting Plots')

figure(1) %position plots
clf
cm = colormap(jet(t));
hold on
for t = 1:frames
plot(pos_new(t,:,1),pos_new(t,:,2),'.','Color', cm(t,:))
end
xlabel('x position (m)')
ylabel('y position (m)')
c = colorbar;
c.Label.String = 'Time (s)';
numticks = numel(c.Ticks);
timeskips = floor(numel(time_new)/numticks);
c.TickLabels = num2str(time_new(1:timeskips:end));
hold off
% xlim([-0.03 .03])
% ylim([-0.03 .03])

figure(2) % x mean plot
plot(time_new,avg_all_camera(:,1))
xlabel('time (s)')
ylabel('vx mean (m/s)')

figure(3) %std dev plot
plot(time_new,temperature(:,1))
xlabel('time (s)')
ylabel('T_x (eV)')

figure(4) % y mean plot
plot(time_new, avg_all_camera(:,2))
xlabel('time (s)')
ylabel('vy mean (m/s)')
```

```
figure(5) % std dev plot
plot(time_new,temperature(:,2))
xlabel('time (s)')
ylabel('T_y (eV)')

figure(6) % energies plot
yyaxis left
plot(time_new, PE)
hold on
plot(time_new, KE)
plot(time_new, PE+KE)
ylabel('Calc Energy (eV)')

yyaxis right
plot(time_new, temperature(:,1))
plot(time_new, temperature(:,2))

xlabel('time (s)')
ylabel('Dust Energy (eV)')
legend('PE Total', 'KE Total', 'Calc total', 'Dust Tx', 'Dust Ty')

hold off

figure(8) % temp comparison
plot(time_new,temperature(:,1),'b')
hold on
plot(time_new,temperature(:,2),'g')
plot(time_new,temperature(:,3),'r')
hold off
xlabel('time [s]')
legend('Tx [eV]', 'Ty [eV]', 'Tz [eV]')

if(exist('rdist','var') ~= 0)
figure(9) % avg dist/ g(r) evolution plot
plot(time_new,squeeze(mean(rdist,[1 2])))
title('avg. particle dist. evolution')
xlabel('Time [s]')
ylabel('Avg. distance [m]')
end
```

## B.4. Mathematica- PK-4 All Calculations File

This code uses values from the RSI paper on PK-4 [62] and calculates various conditions in the experiment. This is used for the plasma conditions listed in Section 2.1.2, the experimental results based on screening length in Section 3.5, and influencing the MD Simulation parameters of Chapter 4.

### Variable Initializations

```
in[1]= (* PK-4 Variables *)
p = 0.6 * 100; (* mBar * mB to Pa conversion *)
i = 0.7; (* [mA] *)
element = neon;
rd = (3.34 * 10^-6)/2; (* [m] *)

e0 = 8.854 * 10^-12; (* [Nm^2/C^2] *)
kb = 1.380 * 10^-23; (* [J/K] *)
qe = 1.6 * 10^-19; (* [C] *)
amu2kg = 1.66054 * 10^-27; (* kg/amu *)

(* calculations *)
rho = 1.510 * 10^3; (* for MF [kg/m^3] *)
md = 4/3 * Pi * rho * rd^3 (* [kg] *)
q = rd * 10^6 * 2 * 2000 * -1.6 * 10^-19; (* [C],
assuming 2000 e- per um diameter of dust *)
pcount = IntegerPart[p/20];
(* to get fit values from table in RSI calculations *)
(* 0.2 = 1, 0.4 = 2, 0.6 = 3 so /20 *)
neon = 20.18; (* neon mass in amu *)
(* neon = 16.026; ??? *)
argon = 39.948; (* argon mass in amu *)
fitvalues = Dataset[{
  <|"a" -> 1.92, "b" -> -.38, "c" -> 7.13, "d" -> 1.23, "f" -> 1.97, "g" -> 0.14|>,
  <|"a" -> 2.75, "b" -> -.42, "c" -> 7.06, "d" -> 0.75, "f" -> 2.11, "g" -> 0.072|>, <|
  "a" -> 3.15, "b" -> -.34, "c" -> 6.98, "d" -> 0.77, "f" -> 2.07, "g" -> 0.098|>};];

Out[10]= 2.94588 * 10^-14

In[16]= rd * 10^6 * 2 * 2000
Out[16]= 6680.
```

Printed by Wolfram Mathematica Student Edition

## RSI Calculations

```
in[17]= ne = fitvalues[pcount, "a"] * i + fitvalues[pcount, "b"] * i^2 (* [10^8 cm^-3] *)
      te = fitvalues[pcount, "c"] + fitvalues[pcount, "d"] * i^-1 (* [eV] *)
      efield = (fitvalues[pcount, "f"] + fitvalues[pcount, "g"] * i^-2) * 100 (* [V/m] *)
```

```
Out[17]= 2.0384
```

```
Out[18]= 8.08
```

```
Out[19]= 227.
```

## Debye Length

```
in[20]= λD = Sqrt[ε0 * (te * qe) / ((ne * 10^8 * 100^3) * qe^2)] (* [m] *) (* electron *)
      (* eV * C / eV          ne [cm^-3 * (cm/m)^3] *)
```

```
Out[20]= 0.00148105
```

## Thermal Drag

$$F_{\text{epstein}} = \frac{-8}{3} \sqrt{(2\pi)} r_d^2 m_N N_N v_{T_N} v_{\text{rel}} = \frac{-8}{3} \sqrt{(2\pi)} r_d^2 m_N \frac{P}{k_B T} v_{T_N} v_{\text{rel}}$$

$$\text{and } F_{\text{drag}} = -\gamma m_d v_{\text{rel}}$$

$$\text{so } \gamma = \frac{8}{3} \sqrt{(2\pi)} r_d^2 m_N \frac{P}{k_B T} v_{T_N} / m_d$$

Now for the debate about  $v_{T_N}$  :

```
in[21]= (*if v_{T_N} = sqrt(k_B T / m_N), where T is Dust temp, assume room temp *)
      vtn1 = Sqrt[((1/40) * qe) / (element * amu2kg)];
      (*1/40 eV * ev to C, amu * amu to kg *)
      γ1D = (8/3 * Sqrt[2 Pi] * rd^2 * p / vtn1) / md (* [m^2 * kg/m/s^2 * s/m /kg = 1/s] *)
```

```
Out[22]= 109.896
```

```
in[23]= (*if v_{T_N} = sqrt(8k_B T / π m), gas avg speed (3D) *)
      vtn3 = Sqrt[8 * ((1/40) * qe) / (3 * element * amu2kg)];
      (*1/40 eV * ev to C, amu * amu to kg *)
      γ3D1 = (32/3 * rd^2 * p / vtn1) / md (* [m^2 * kg/m/s^2 * s/m /kg = 1/s] *)
      γ3D2 = (64/3 * Sqrt[2/Pi] * rd^2 * p / vtn3) / md
```

```
Out[24]= 175.369
```

```
Out[25]= 171.372
```

## Epstein Timescale

```
In[39]= pstime = Integrate[-8/3 * Sqrt[2 Pi] rd^2 p v / (vtn1 * md), {v, .015, .000}]
pstime * 71.4 (*frames*)
```

```
Out[39]= 0.0123633
```

```
Out[40]= 0.882742
```

```
In[41]= 1/0.01236335867207524^
```

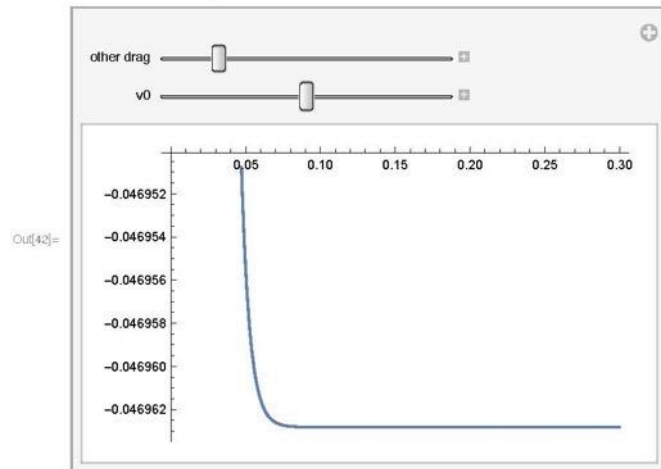
```
Out[41]= 80.8843
```

## Other Drag Calculation

$F = F_{\text{drag}} + F_{\text{electric}} + F_{\text{other}} + F_{\text{thermal}}$  (ignoring for this ODE)

$a = -\gamma v + qE/m + \alpha v$

```
In[42]= Manipulate[Plot[Evaluate[
  v[t] /. DSolve[{v'[t] == -(γ3D1 + α) * v[t] + q * efield / md, v[0] == v0}, v[t], t]],
  {t, 0, 0.3}], {{α, 0, "other drag"}, -100, 500}, {{v0, 0}, -2, 2]
```



```
In[43]=
```

```
termvel = -0.02;
```

```
Solve[α2 * termvel == -γ3D1 * termvel + q * efield / md, α2]
```

```
Out[44]= {{α2 → 236.422}}
```