Characterizing Heliospheric and Interstellar Interactions Using Pickup Ion Measurements

by

Sarah A. Spitzer

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Climate and Space Sciences and Engineering)
in the University of Michigan
2022

Sarah A. Spitzer

saraylet@umich.edu

ORCID iD: 0000-0001-9607-1492

DEDICATION

To Alon. You are my sunshine and the light of my life. I hope that anyone who reads this thesis will understand the gravity of that.

ACKNOWLEDGMENTS

There have been many important and influential people in my life, and while I can't list them all here, I want to acknowledge the people who helped me get this thesis to where it is today.

First, I want to acknowledge my advisors, Sue Lepri and Jim Raines.

I never took seriously enough the advice and warning that I started to hear more and more as I explored the option of starting in a Ph.D. program that it is critical to find an advisor who works well with you. I met Sue when I took the CLaSP Space Instrumentation course, also dubbed "the balloon class," with her. I was applying to the CLaSP Ph.D. program, and space instrumentation sounded like exactly what I wanted to do, so taking the course as a cognate for my Electrical and Computer Engineering master's degree seemed like a win-win situation. I understand now how lucky I was to have found such a great fit as an advisor and can only say how much I appreciate having had her support throughout the Ph.D. process. Sue is an incredible role model who somehow manages to balance her position teaching, researching, and advising as a professor in the CLaSP department at UM in addition to being incredibly active in committees and outreach, Co-PI for the Solar Orbiter Heavy Ion Sensor, Director of the Space Physics Research Laboratory, and so many other professional qualifications, while still maintaining a work-life balance and finding time for her family and her personal interests. I owe a lot to Sue in my growth as a Ph.D. student, and I am very lucky to have had the opportunity to advance through the program under her expertise and guidance. One of the greatest compliments I have gotten is being compared to Sue, and "when I grow up," I wouldn't mind being a lot like her.

I tell people that I am one of the few people who has been examined by their advisor on a qualifying exam committee through the 2-hr long Ph.D. oral qualifying stage 1 exam, as I was only "adopted" by Jim later, a couple of years into the program. It is usual in CLaSP for Ph.D. students to have two co-advisors, and my relationship with Jim developed so naturally as we started working together on projects, that he was often my first go-to for advice or to work out a problem I was stuck on. By the time the idea came up that we should formalize his role as co-advisor with the department, it felt like it already went without saying and was almost surprising that it wasn't already "on paper." I have always been incredibly impressed with Jim's enthusiasm not only for research, but also for taking on mentorship and teaching roles beyond the requirements of his position. I have not met a single faculty member in the department who is more invested or interested in their students. It is obvious through his involvement how invested Jim is in the department, and through his willingness to support student progress, he became one of the few go-to qualifying examiners for Advanced Fluid Dynamics, and I witnessed his involvement in more qualifying exams than most faculty members in the department. I have always felt that Jim was there when I needed guidance or support, and I have learned so much working with him. Again, I have been incredibly lucky to have found Jim as a Ph.D. advisor.

I also want to acknowledge my mentor, Jason Gilbert. Jason is an incredibly knowledgeable scientist with an impressive expertise in instrumentation. Jason mentored me from the beginning of my time in the Ph.D. program in CLaSP, and I have learned so much from him about spaceflight hardware. During our time overlapping in Michigan, he was always a great resource for research advice and really supported me as I got started in the program. Even at a distance, Jason has always been there when I have needed his help on an instrumentation problem, and his willingness to help me work through some critical phases of my research really enabled me to overcome some seemingly insurmountable barriers and to succeed in solving numerous research problems.

I want to acknowledge all of my collaborators on the papers in progress presented in this work, but first and foremost, I want to acknowledge Eberhard Möbius and Frederic Allegrini. Both Eberhard and Frederic have really gone above and beyond the call of duty as research collaborators, and I count them among my Ph.D. mentors. Eberhard helped me give real traction, significance, and foundation to my research with interstellar pickup ions. He helped me turn a small introductory investigation into a scientific analysis worthy of a Ph.D. thesis, and I was able to learn so much and make so much progress on my collaboration trips to work with him "*in situ*" at the University of New Hampshire. Frederic also went above and beyond, attending virtual weekly advising meetings for the SO-HIS calibration for two years. He has always been willing provide helpful insights and is incredibly knowledgeable about SIMION and the SO-HIS instrument. My work on this project would not be anywhere near what it is today without his guidance and expertise.

I also want to acknowledge Stefano Livi who, as PI of the SO-HIS instrument, has also been instrumental in my ability to work on and complete this project. His support, knowledge, and expertise have been invaluable.

As a Ph.D. student, I had the unique opportunity to mentor five graduate and undergraduate students in directed research. I especially want to acknowledge the work done by Connor Raines, which really facilitated the progress made on the SO-HIS calibration.

All of the collaborators on the projects presented in this thesis provided great insights and really helped me get the work to where it is today. Therefore, I want to acknowledge Sue Lepri, Jim Raines, Jason Gilbert, Eberhard Möbius, Frederic Allegrini, Jonathan Bower, Connor Raines, Stefano Livi, Austin Glass, and Ryan Dewey. I especially want to acknowledge Jonathan Bower for his great insights and long-standing support on my research of the interstellar flow direction.

Of course, I would not be able to complete this program without my Ph.D. committee, so I want to specially thank all of the committee members, Professor Susan Lepri, Dr. Jim Raines, Professor Mark Kushner, Professor Stefano Livi, and Professor Eberhard Möbius.

When I started in graduate school, I first completed a master's degree in Electrical and Computer Engineering before finding my niche in spaceflight hardware and *in situ* space plasma measurements. This would not have been possible without the confidence and support of Professor Amir Mortazawi. Thank you for believing in me and giving me the opportunity to succeed. I also want to acknowledge my ECE program advisor, Professor Jessy Grizzle, for his guidance throughout the master's degree program.

I have been fortunate to have so much support from so many people in the Climate and Space Sciences and Engineering Department. I especially want to acknowledge a number of students for their friendship and support, for really helping me learn the foundational coursework, and for serving with me as leaders in the department on the executive board of the Graduate and Undergraduate Student Organization, namely Ben Alterman, Alicia Petersen, Camilla Harris, Yash Sarkango, Abby Azari, Yeimy Rivera, Ryan Dewey, Tong Shi, Chris Bert, Zach Fair, and Anthony Torres. From outside of CLaSP, I additionally want to thank Eshed Magali and Robert Jarolim for their friendship and support.

Alon. From the moment you entered my life, you gave it meaning and clarity. Everyone who knows us can tell that we were meant for each other, and I really can't describe how you complete me. I believe there just aren't words to describe the support you have and continue to give me, so I won't try, because it wouldn't do you justice. You have made all the difference.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ACRONYMS

ACE	Advanced Composition Explorer
C&DH	Command and Data Handling
CAPS	CAsinni Plasma Spectrometer
CHEM	CHarge-Energy-Mass Spectrometer
CIR	Co-rotating Interaction Region
CME	Coronal Mass Ejection
E/q	Energy Per Charge
EA-IS	Electrostatic Analyzer with Ion Steering
EAR	Energy – Angle Response
EAS	Electron Analyzer System
ESA	Electrostatic Analyzer
FIP	First Ionization Potential
FIPS	Fast Imaging Plasma Spectrometer
FOV	Field Of View
GCR	Galactic Cosmic Ray
GF	Geometric Factor
HCS	Heliospheric Current Sheet
HIS	Heavy Ion Sensor
HVPS	High Voltage Power Supply
IMF	Interplanetary Magnetic Field
IOM	Ion Optical Model

LIC Local Interstellar Cloud

LISM Local Interstellar Medium

LVPS Low Voltage Power Supply

M/q Mass Per Charge

MCP Micro-Channel Plate

PA Potential Array

PAS Proton and Alpha Sensor

PLASTIC PLASma and SupraThermal Ion Composition experiment

PUI Pickup Ion

SO Solar Orbiter

SS Source Surface

SSD Solid State Detector

STEREO Solar Terrestrial Relations Observatory

SULEICA Suprathermal Energy Ionic Charge Analyzer

SWA Solar Wind plasma Analyzer

SWICS Solar Wind Ion Composition Spectrometer

TOF Time Of Flight

VLISM Very Local Interstellar Medium

ABSTRACT

The Heliosphere, orbiting the galactic center of the Milky Way, is a plasma bubble carved out of the Local Interstellar Medium (LISM) by the Sun's dynamic and magnetic influence. Its relative motion creates a steady inflow of neutral particles and dust from the LISM through the Heliosphere, influencing the space environment throughout and impacting the entire Solar System, including the Earth. This thesis addresses the question: How does the Heliosphere — Local Interstellar Medium interaction evolve over time? These Heliosphere — interstellar interactions can be studied *in situ* using measurements of pickup ions (PUIs), ions originating from neutrals, such as interstellar neutrals, ionized through photoionization, electron impact ionization, or charge exchange with the solar wind. This thesis presents measurements of the longitudinal inflow direction, λ_{ISN}, using He$^+$ PUI measurements in the downwind focusing cone. This new method is validated, measuring a flow direction comparable to recent literature studies of 75.37° $\pm$0.43° over the ACE/SWICS data set including focusing cone crossings in the years 1998 through 2010. This method is applied to 3-orbit boxcar averages of combined count distributions with center years 1999 through 2009 to measure the flow direction trend over an 11-year solar cycle. A trend of 0.00° $\pm$0.51° is measured, indicating that no evidence is found of a longitudinal flow direction variation over this 11-year period.

This thesis additionally discusses ongoing efforts to further characterize Heliosphere — interstellar interactions by measuring a more precise yearly flow direction using upwind He$^+$ PUI measurements, as well as a characterization of new spaceflight hardware and challenges associated with current analysis and measurement techniques. Future studies to characterize these interactions will be greatly enhanced with improved measurements implementing new instrument designs and techniques, such as by the Solar Orbiter Heavy Ion Sensor (SO-HIS) launched as a joint ESA/-NASA mission on 9 February, 2020. SO-HIS is an *in situ* time-of-flight triple-coincidence ion mass spectrometer with elevation selection measuring elemental composition and 3D velocity distribution functions. Its higher cadence, specifically at the 30 s and 4 s resolutions, along with the ability to resolve angular measurements, stepping through 16 elevation steps and having an increased continuous azimuthal acceptance of -30° to +66° to include more of the PUI population off the Sun–spacecraft line in the ecliptic, will greatly improve the future available PUI data set for use in studying Heliosphere — interstellar interactions. This thesis presents the cross-calibration of SO-HIS with its SIMION ion optical model to characterize the geometric factor, a measure of the instrument's useful particle intake geometry, under standard solar wind conditions. Validation of the model with laboratory measurements is presented, along with a function of the instrument's geometric factor for elevations from -15° to +18°, ±3°, for standard solar wind energies and azimuths. This characterization will allow the measurements taken by the SO-HIS instrument to be converted to higher level data products useful for studying the *in situ* plasma, including PUIs. Solar Orbiter, with the SO-HIS instrument, having a trajectory up to ∼30° out of the ecliptic in the extended mission, as well as future missions and advancements, will enable further studies of Heliosphere — interstellar interactions, including measuring precise yearly interstellar flow directions and mapping and tracking the 3D structures of the of the interstellar He+ pickup ion distribution in the inner Heliosphere.

CHAPTER 1

Introduction

The Sun orbits the galactic center of the Milky Way, traversing the Very Local Interstellar Medium (VLISM), surrounded by its sphere of influence, the Heliosphere. While not necessarily perfectly spherical in shape, the Heliosphere comprises the region of space affected by the Sun, as a plasma bubble carved out of the VLISM by the continuous stream of solar wind and Interplanetary Magnetic Fields (IMF) from the Sun. While the Heliosphere is created and controlled by the Sun, it is also largely impacted by the VLISM. The Heliosphere and the VLISM interact at the boundary known as the Heliopause, and these interactions influence the composition and dynamics of the space environment throughout the Heliosphere, affecting the whole Solar System, including the planets, such as the Earth.

1.1 The Sun and Heliosphere

1.1.1 The Sun and its layers

The Sun is classified as a G2 v spectral class middle aged average main-sequence star, where the G2 v spectral class means that convection carries the energy generated by nuclear fusion in the Sun's core to the outer layers of the Sun. The Sun has a radius of around 700 Mm, a mass of around 2×10^{30} kg, and an effective temperature of around 5780 K. The magnetic field of the Sun is controlled by the magnetic dynamo, which is responsible for solar activity and controlled by differential rotation paired with stratified convective motions of the solar plasma layers. The differential rotation at the surface is currently about 25 Earth days at the equatorial regions and about 35 Earth days at the poles. (Hansteen (2009a))

The Sun is stratified into layers, as shown in Figure 1.1, starting from the Core and transitioning to the Radiative Zone, the Convective Zone, and the solar atmosphere, which is further stratified into the Photosphere, the Chromosphere, the Transition Region, and the Corona, which expands into the solar wind, creating the Heliosphere.

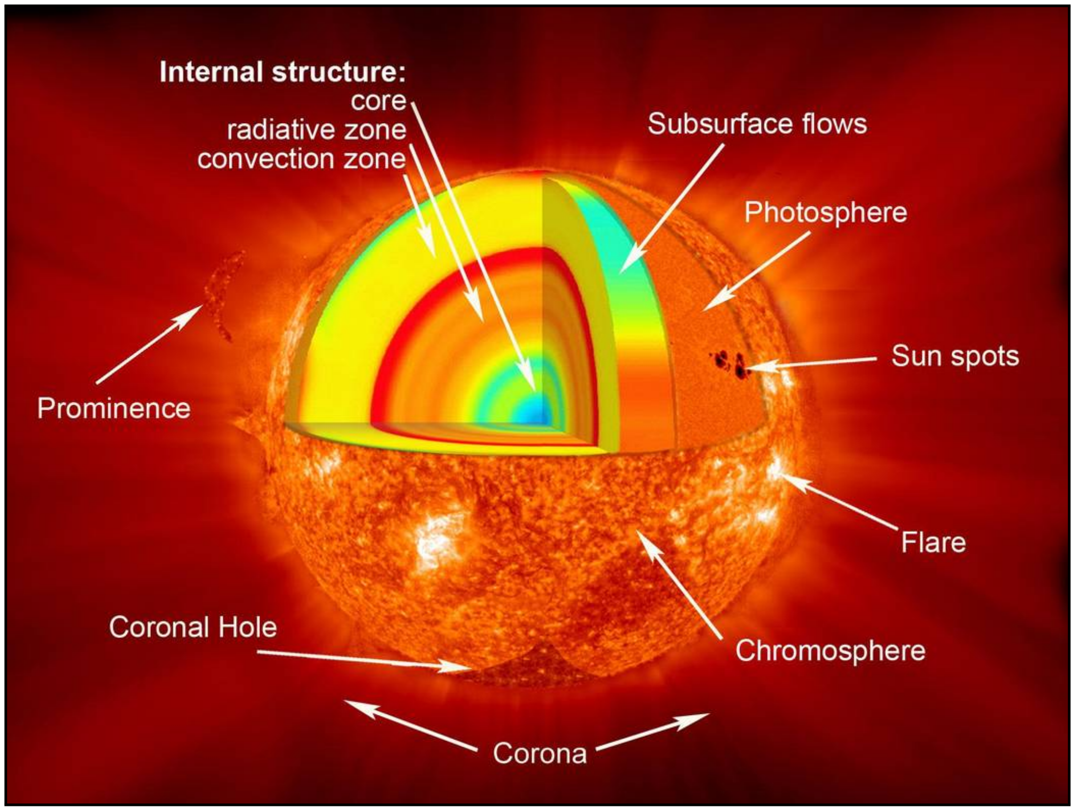


Figure 1.1: Diagram depicting the layers and various features of the Sun. Figure from NASA (2017).

1.1.1.1 From the Core to the Solar Surface

Nuclear reactions occur within the Sun's Core, which is at roughly 15 MK, fusing hydrogen into helium. The Radiative Zone is characterized by energy transport that occurs by radiation transfer or photon diffusion. The Convection Zone, which extends from around 0.7 – 1 Rs (the radius of the Sun's surface), transitions from around 2 MK to around 6000 K. In the Convection Zone, the high temperature gradient drives a buoyancy force which leads to convection of the plasma from the interior to the surface and back down. Hydrogen, helium, and heavier elements eventually get carried out from the Sun through the solar atmosphere into the Heliosphere by the solar wind. (Kivelson & Russell (1995); Hansteen (2009a))

1.1.1.2 The Solar Atmosphere

The Photosphere The Photosphere, the innermost layer of the solar atmosphere, is what we see as the visible surface of the Sun, "photo" coming from the word light. This is the layer from which is emitted the bulk of the electromagnetic radiation leaving the Sun. This layer is relatively shallow, around 500 km in depth, and comprises a relatively cool, thin atmosphere at around 5800 K. Through the photosphere, we see structures on the convective surface of the Sun, including granules, supergranules, sunspots, and faculae. (Kivelson & Russell (1995))

The Chromosphere After the Photosphere, the Chromosphere, “chrome” meaning color, which is dominated by UV emissions, is around 2 – 5 Mm thick and ranges from around 6000 K to around 20,000 K and transitions to the outer solar atmosphere, the Corona, through the Transition Region, a layer around 200 km thick, at which point the plasma density decreases and the temperature increases by around 2 orders of magnitude to around 1 MK or more. (Kivelson & Russell (1995); Hansteen (2009a))

The Corona The Corona, or “crown,” is a low density, ionized plasma emitting highly in the X-Ray band. The Corona, which extends a few hundred Mm beyond the Chromosphere, has a temperature of around 1 MK. The rapid increase in temperature through the Transition Region to the Corona is often referred to as the “coronal heating problem.” While the coronal heating problem has yet to be fully answered, the heating appears to be driven by dissipation of magnetic turbulence and coupling to plasma waves (Alfvén (1942); Hansteen (2009a); Kasper et al. (2021)). The Corona then expands into a supersonic radial flow known as the solar wind, and the steady stream of plasma along with the solar IMF create the region of space affected by Sun, known as the Heliosphere. (Kivelson & Russell (1995))

The Solar Wind Solar wind particles generally stream radially outward in a relatively steady supersonic flow. However, they remain magnetically connected to the region on the Sun from which they originate, as demonstrated in Figure 1.2, creating a spiral-shaped magnetic field. While the magnetic field remains relatively radial out to a region known as the Alfven surface, as it stretches further away from the Sun, it begins to form this spiral pattern of magnetic field lines connected to radially propagating solar wind particles which are magnetically connected to the rotating Sun. This spiral-shaped IMF is known as the Parker Spiral and continues outward from the Sun in 3 dimensions to fill the Heliosphere, until it reaches the Local Interstellar Medium (LISM) at the Heliopause. (Kivelson & Russell (1995); Hansteen (2009b))

1.1.2 Features on the Sun

Granules, on the order of 500 – 1000 km across, are convection cells in which hot convective plasma rises to the surface, spreads and cools, and sinks at the darker boundaries, with a lifetime of around 5 min, while supergranules are structures comprising numerous granules and are on the order of 30 Mm across, with a lifetime on the order of one Earth day. Sunspots appear as darker

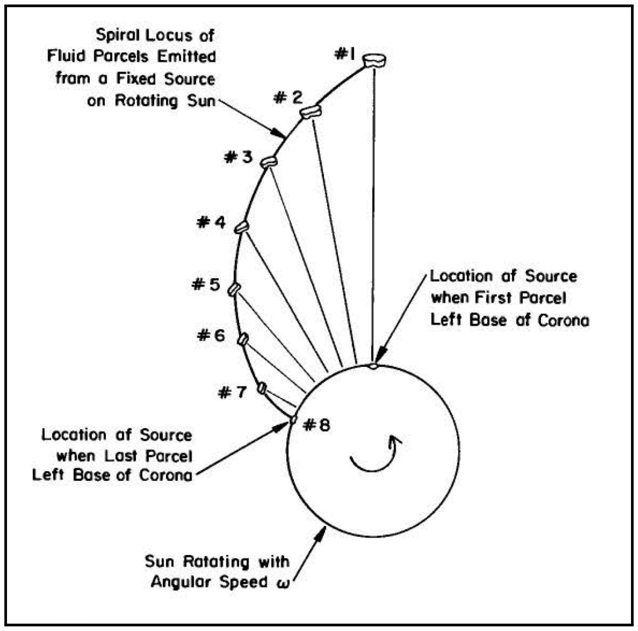


Figure 1.2: Diagram depicting a parcel of plasma leaving the Sun in the radial direction and remaining magnetically connected to the rotating Sun, resulting in a spiral shaped magnetic field. Figure from Kivelson & Russell (1995).

regions, since they are cooler than the surrounding plasma, at around 4000 – 5000 K, and faculae appear as bright spots, hotter than the surrounding plasma. Sunspots usually form in pairs or clusters of oppositely polarized regions, with the opposing pole leading in opposite hemispheres of the Sun, in regions where the convective motions of the plasma are dominated by strong magnetic fields. As the solar cycle progresses and the number of sunspots increase, the sunspots are seen to migrate toward the equatorial region, though individual sunspots only last on the order of a few solar rotations. (Kivelson & Russell (1995); Hansteen (2009a))

1.1.3 The Solar Dynamo

1.1.3.1 The Solar Cycle

The Sun has a solar cycle of around 11 years, during which the Sun progresses from a relatively quiet dipole-like magnetic field in solar minimum to an active more complex magnetic structure including more sunspots and active regions at solar maximum, resulting in higher activity associated with more flares and eruptions. The Sun's magnetic field reverses poles over an $\sim$11-year cycle and completely flips poles from one orientation to another and then back again over an $\sim$22-year cycle, as shown in Figure 1.3.

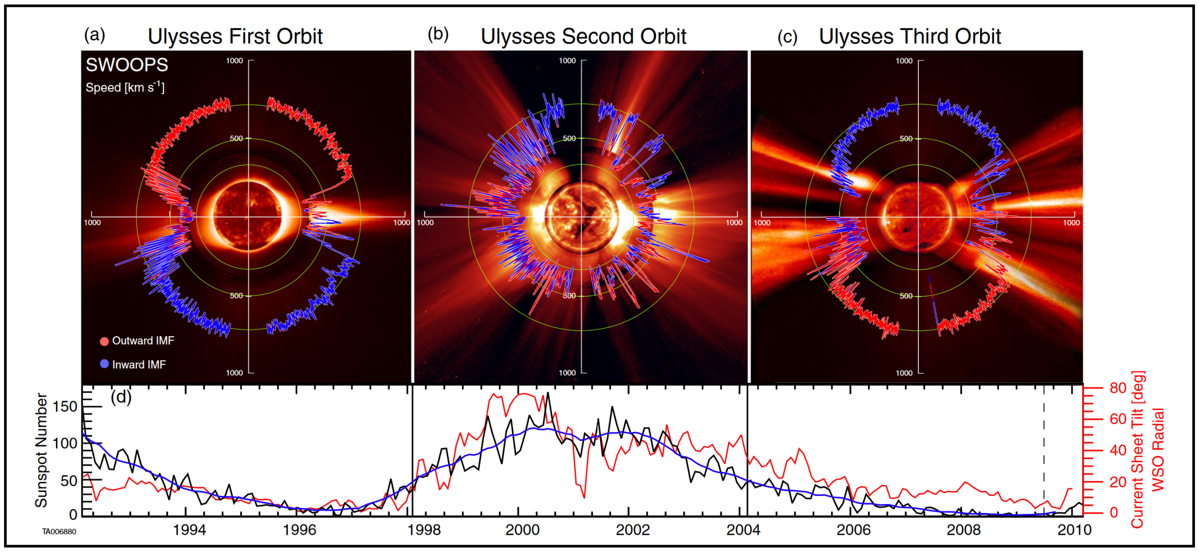


Figure 1.3: Images taken of the Sun by Ulysses at various times during the the solar cycle. Top left: Quiet time solar minimum in original magnetic pole orientation. Top middle: Active solar maximum with flipping magnetic field. Top right: Quiet time solar minimum in reversed magnetic pole orientation. Bottom: Graph of sunspot number demonstrating the solar cycle corresponding to images above. Figure from McComas et al. (2013).

1.1.3.2 The Interplanetary Magnetic Field

The most basic structure of the solar dynamo can be likened to a dipole magnetic field with North and South poles, which flip over and back again on an $\sim$22 year cycle. However, due to the opposite magnetic polarity of the "open" field lines originating from the polar regions of the Sun, as discussed later, a current sheet forms near the Solar equatorial region. Additionally, the Sun's magnetic and rotational axes are offset by $\sim$7°, causing this Heliospheric Current Sheet (HCS) to wave up and down, as often compared to a "ballerina skirt" shape (Alfven (1977)), as shown in Figure 1.4. Therefore, as the Earth orbits the Sun, it passes in and out of HCS into opposite polarity magnetic field regions of the IMF. (Kivelson & Russell (1995); Hansteen (2009b))

Close to the Sun, the magnetic fields near the equatorial regions largely form closed magnetic loops, while the magnetic fields from the poles extend outward into the Heliosphere. While these magnetic fields are still divergence-free, they connect elsewhere to other magnetic structures rather than closing back on themselves, unlike the equatorial fields. These polar magnetic field lines are therefore often referred to as Coronal holes or "open" magnetic field lines, as they are "open" to other regions of the Heliosphere, as shown in Figure 1.5.

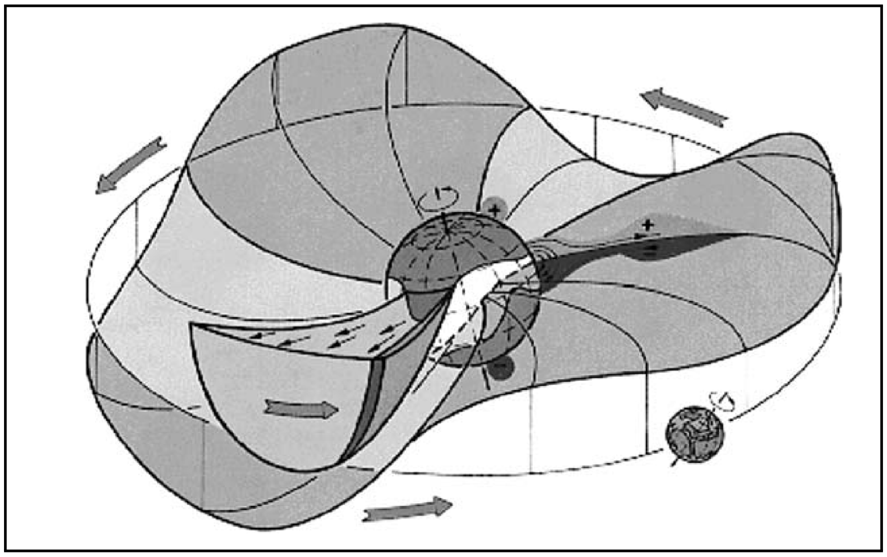


Figure 1.4: Diagram of the Earth orbiting the Sun, passing through the HCS. Figure from Heber & Potgieter (2006), adapted from Schwenn & Marsch (1990).

1.1.4 The Fast and Slow Solar Wind

The solar wind can be largely classified into two major types, which generally originate from these "closed" and "open" regions of magnetic fields on the Sun. The solar wind known as the slow solar wind, which has a bulk flow speed on the order of 300 – 500 km/s, tends to originate from the equatorial regions, while the solar wind knows as the fast solar wind, which has a bulk flow speed on the order of 700 – 900 km/s, tends to originate from the polar regions. In addition to speed, the fast and slow solar winds are also distinguished by other properties characterized by their region of origin on the Sun, and they can be differentiated by their heavy ion compositions and charge-states. While all solar wind is dominated by protons, or H^+, followed by alphas, or He^{2+}, the frozen-in-flux conditions in the solar wind cause these two bulk flows to retain important characterizing features in the major heavy ion species and ions found in their populations. Due to the activity and rotation of the Sun, these fast and slow streams can often interact and sometimes steepen into shocks, especially when the polar regions known as Coronal holes extend down toward the equatorial regions, such as around solar maximum. Additionally, compression and rarefaction regions with higher and lower densities can form within the solar wind as a result of the speed differentials between these flows, as shown in Figure 1.6. When these regions co-rotate with the Sun, they are known as Co-rotating Interaction Regions (CIRs). Therefore, while the Earth orbits the Sun near the equatorial region in the ecliptic plane, the near-Earth space environment is heavily influenced both by the slow solar wind originating from the equatorial regions as well as the fast solar wind originating from the polar regions. (Hansteen (2009b))

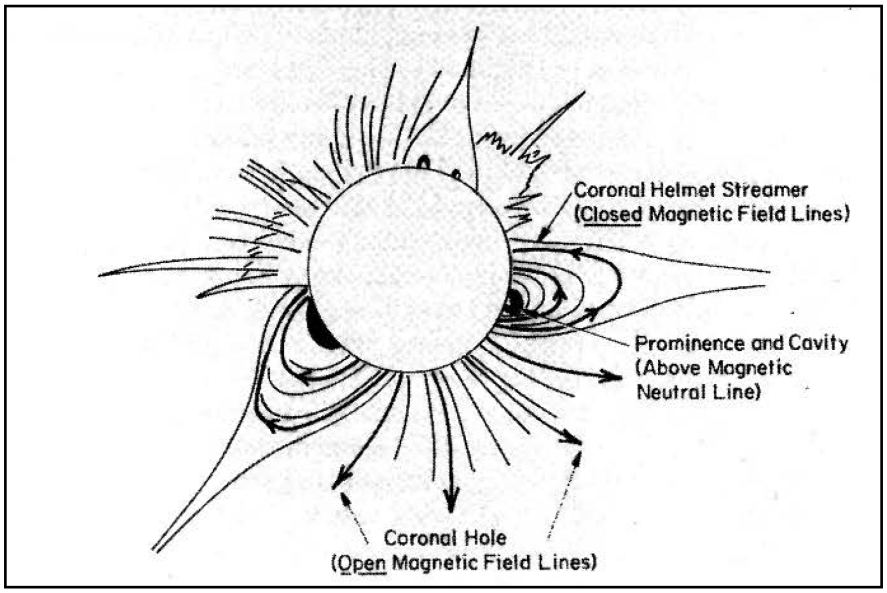


Figure 1.5: Diagram depicting features on the Sun including polar coronal holes. Figure from Kivelson & Russell (1995).

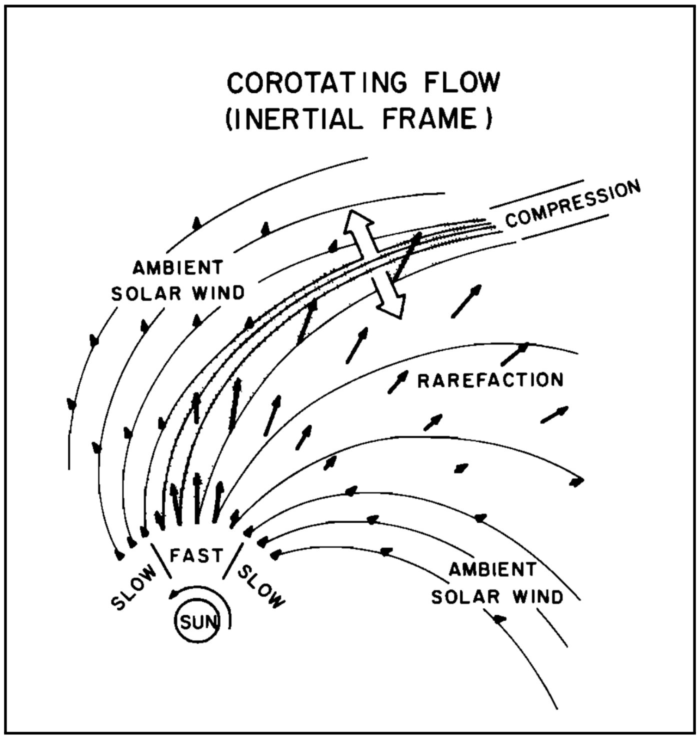


Figure 1.6: Diagram depicting interactions between fast and slow solar wind leading to co-rotating compression and rarefaction regions. Figure from Pizzo (1985).

1.1.5 Solar Transients

In addition to the comparatively steady stream of solar wind being emitted from the Sun, the Sun also produces transients, including solar flares and coronal mass ejections (CMEs). Flares are characterized by sudden bursts of EM radiation, causing a brightening across all wavelengths of the spectra. As they comprise EM radiation, solar flares consist of photonic energy and have a high impact on spacecraft instruments, satellites, and Earth systems such as radio and navigation. CMEs are ejections of plasma and magnetic field from prominences in the solar Corona, as shown in Figure 1.7. They comprise massive parcels of plasma including the leading edge, a rarefied region known as a cavity, and a dense core, and can be ejected at speeds anywhere from below the bulk slow solar wind speed to thousands of km/s. ICMES are able to reach the Earth in a time frame from on the order of a week to around approximately 16 hrs from the time they are released from the Sun, and an average ICME traveling around 550 km/s (Jian et al. (2006)) can reach the Earth in around 3 days. Interplanetary Coronal Mass Ejections, or ICMEs, are CMEs that are ejected Earthward and can lead to geomagnetic storms and other space weather effects, as well as disruptions to the steady solar wind and the magnetic field throughout the Heliosphere. ICMEs can cause interference and damage to satellites and other spacecraft as well as instruments and other electrical equipment both in space and on the ground. The greatest number of CMEs occur when the Sun is most active around solar maximum.

1.1.6 Heliosphere — Interstellar Interactions

The Sun orbits the galactic center of the Milky Way Galaxy, traversing a locally denser region of dust, plasma, and neutral particles known as the Local Interstellar Cloud (LIC) in the LISM. The boundary between the Heliosphere and the LISM is known as the Heliopause. Due to the Solar IMF, most ions in the LISM are diverted around the Heliosphere, with the main exception of highly energetic Galactic Cosmis Rays (GCRs). As previously noted, more solar activity occurs around solar maximum, producing sun spots, faculae, flares, and CMEs. However, it is also the time when the Heliosphere is the most enhanced by the solar wind and provides the greatest amount of shielding from interstellar radiation, such as GCRs. Unaffected by the magnetic boundary at the Heliopause, neutral particles from the LISM are able to flow freely across this boundary, creating the interstellar wind through the Heliosphere. These interstellar neutrals affect the composition in the Heliosphere and can directly interact with the solar wind after becoming ionized through processes such as photoionization, electron impact ionization, and charge exchange with the solar wind. These newly ionized particles, now affected by the Solar magnetic field, are "picked up" and accelerated by the IMF, and therefore known as pickup ions (PUIs) (Fahr (1968); Vasyliunas & Siscoe (1976); Möbius et al. (1985)). They become embedded in the solar wind population,

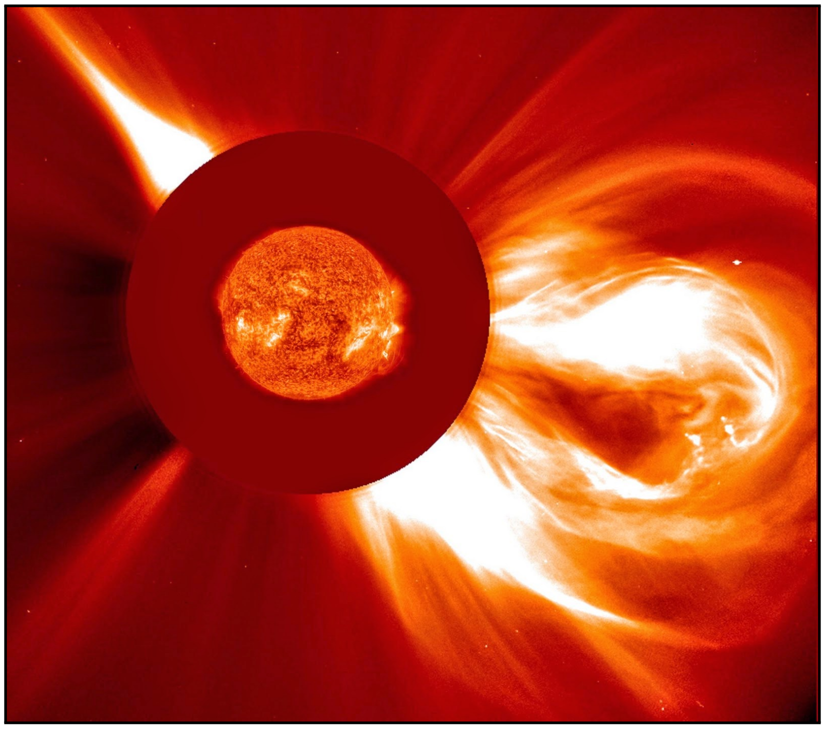


Figure 1.7: Image of a CME emerging from the Sun on the lower right taken by the Solar and Heliospheric Observatory. Figure from NASA (2016).

affecting its dynamics and composition, and can be used to study properties of the Heliosphere — interstellar interactions, as discussed in more detail in Chapters 3 and 4.

1.2 Pickup Ions

PUIs are particles that originate as neutrals but become ionized through a process such as electron impact ionization, photoionization, or charge exchange with the solar wind. Once the particles become ionized, they are picked up, or accelerated by the solar magnetic field and begin to propagate radially outward with the solar wind. PUIs can originate from various sources including solar source, inner sources such as planets, comets, and dust, and from interstellar neutrals. Once the PUIs become embedded in the solar wind, they can be distinguished by their charge-states and signatures in their velocity distributions. (Möbius et al. (1985))

As PUIs originate as neutrals, they tend to have low charge-states, usually +1, though He^{2+} PUIs were also identified by Gloeckler & Geiss (1996). Interstellar PUIs with a lower first ionization potential (FIP) become ionized more quickly, while interstellar PUIs with a higher FIP, such as helium, remain neutral longer and may penetrate farther into the Heliosphere before undergoing ionization. Therefore, only species with a high FIP, such as helium, will be focused into the downwind focusing cone before becoming ionized, while species with a lower FIP will become ionized more quickly and will be found in the upwind crescent, but not the downwind focusing cone. The gravitational focusing cone is an enhancement formed downwind of the Sun from the interstellar flow due to gravitational focusing of neutrals around the Sun, while the crescent is an upwind enhancement formed due to the survival near the Heliopause of in-flowing interstellar neutrals and their higher cutoff energies when flowing antiparallel to the radial solar wind (Sokół et al. (2016)), as discussed in detail in Chapters 3 and 4.

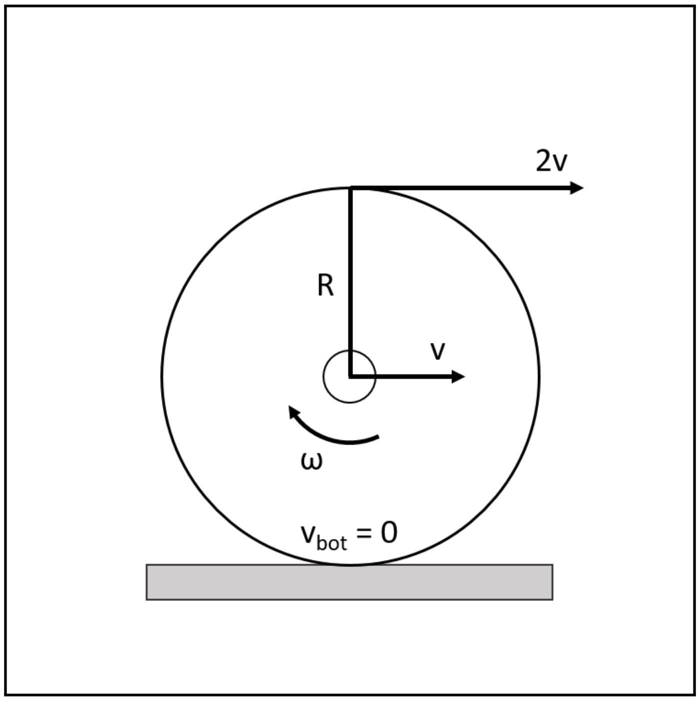


Figure 1.8: Diagram demonstrating that a point on a wheel of radius R and angular velocity ω whose bulk motion is represented by a translational velocity v will have a minimum value of $v_{bot} = 0$ at the bottom of the rotation given no-slip conditions and a maximum of $2v$ at the top of rotation. Figure modeled from Young et al. (2012) and University of New South Wales (2009).

PUIs can be distinguished by their velocity profile in the velocity distribution, as they will have speeds between 0 and 2 times the solar wind speed in the observer frame. This can be demonstrated using the physical motion of a point on a wheel, as shown in Figure 1.8. Given a no-slip condition for the bottom of a wheel with translational bulk velocity v, angular frequency, ω, and radius, R, the velocity v_{top} at the top and v_{bottom} at the bottom are given as:

$$v_{top} = v + \omega R \tag{1.1}$$

$$v_{bottom} = v - \omega R \tag{1.2}$$

Given the no-slip condition, the velocity at the bottom is equal to zero, or $v_{bottom} = 0$. Therefore,

$$\begin{aligned} v_{bottom} &= v - \omega R = 0 \\ &\Rightarrow v = \omega R \end{aligned} \tag{1.3}$$

Substituting back into v_{top} gives:

$$v_{top} = v + \omega R = v + v = 2v \tag{1.4}$$

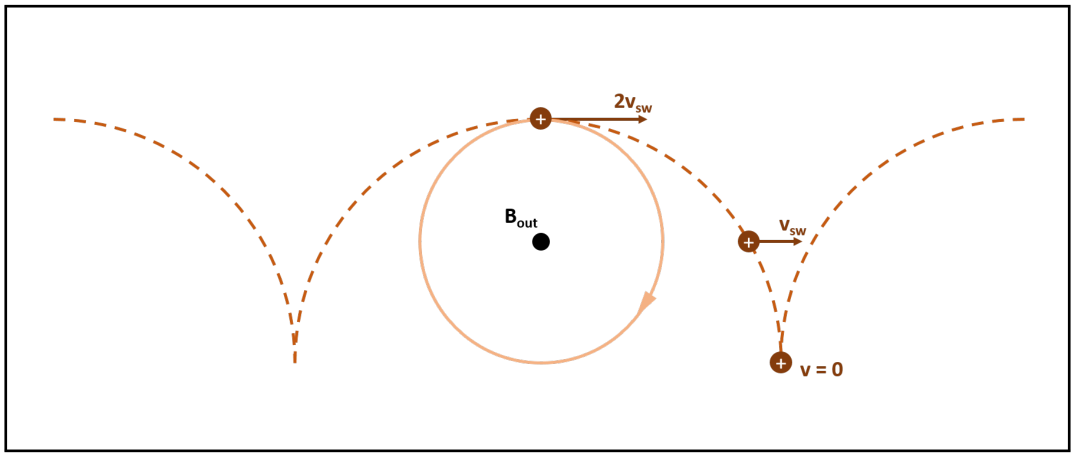


Figure 1.9: Diagram demonstrating PUI motion in a magnetic field, B. A PUI with positive charge gyrates about the magnetic field counter the right-hand rule while propagating with the solar wind, which has a bulk speed of v_{sw}. A PUI injected perpendicular to the magnetic field will have translational speed between $v = 0$ and $v = 2v_{sw}$, or $0 \leq w \leq 2$. Figure modeled from Gilbert (2008).

Similarly, as a PUI injected perpendicularly to the magnetic field propagates with the solar wind of bulk velocity v while gyrating about the magnetic field, it will experience a minimum velocity of $v = 0$ at the bottom of the trajectory and a maximum velocity of $v = 2v_{sw}$ at the top of the

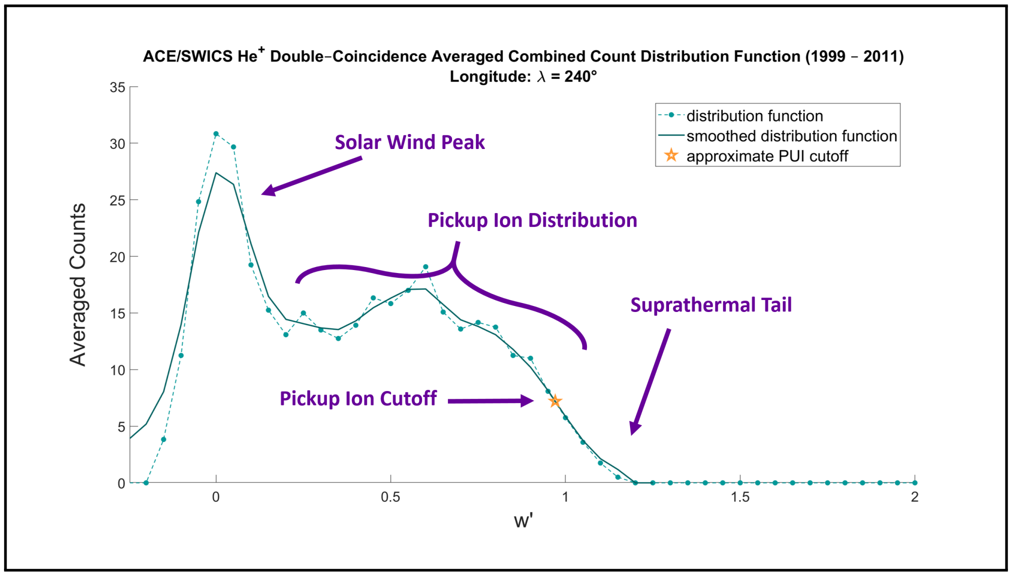


Figure 1.10: ACE/SWICS He$^+$ count distribution for the averaged combined values at ecliptic longitude λ = 240° for the years 1999 – 2011 with regions of the distribution labeled for the solar wind peak, the PUI distribution, the PUI cutoff, and the suprathermal tail.

trajectory, where v_{sw} is the solar wind speed, as shown in Figure 1.9. The particle speed may be represented with relation to the solar wind speed as:

$$w = \frac{v}{v_{sw}} \tag{1.5}$$

$$w' = \frac{v}{v_{sw}} - 1 \tag{1.6}$$

where w represents the inertial, or observer frame, and w' represents the solar wind frame (Gloeckler & Geiss (1998); Möbius et al. (2015a)).

The PUI velocity will therefore be in the range $0 \le w \le 2$ or $-1 \le w' \le 1$. This causes the PUI population to be represented by a distinct portion of the distribution, as shown in figure 1.10. Solar source ions may be found in the solar wind peak, and ions accelerated by other processes, such as shocks, are additionally found in the suprathermal tail of the distribution (Fisk & Gloeckler (2006)).

When PUIs are injected perpendicularly into the interplanetary magnetic field, they will achieve a maximum speed of up to twice the solar wind speed, as described above. However, if the velocity vector of the original neutral particle is at an angle of less than 90° from the magnetic field vector, the maximum speed will be lower. Therefore, the PUI injection at the PUI cutoff in the distribution may be found at or below $w = 2$ or $w' = 1$.

As neutrals are ionized and injected as PUIs, they begin to gyrate around the magnetic field and are injected into a torus, represented by a green band in Figure 1.11, around the magnetic field vector, represented by a red and black striped arrow in Figure 1.11, in velocity space on a shell with radius v_{sw}, or the solar wind velocity, represented by a red and orange sphere in Figure 1.11. Therefore, when the PUI torus is in the field of view of the instrument, as shown for the STEREO/PLASTIC instrument in Figure 1.11, where the green band representing the torus is shown to lie within the blue cone of the instrument's field of view, it is possible to measure the newly injected PUIs near the PUI cutoff. Due to turbulence and pitch angle scattering, the PUIs eventually fill the entire shell distribution in velocity space. However, it was theorized by Gloeckler et al. (1995); Isenberg (1997); Möbius et al. (1998) and confirmed by Drews et al. (2015) that the isotropization of the PUIs from the torus distribution to the shell distribution is a slow process, and the PUI torus signature is largely retained.

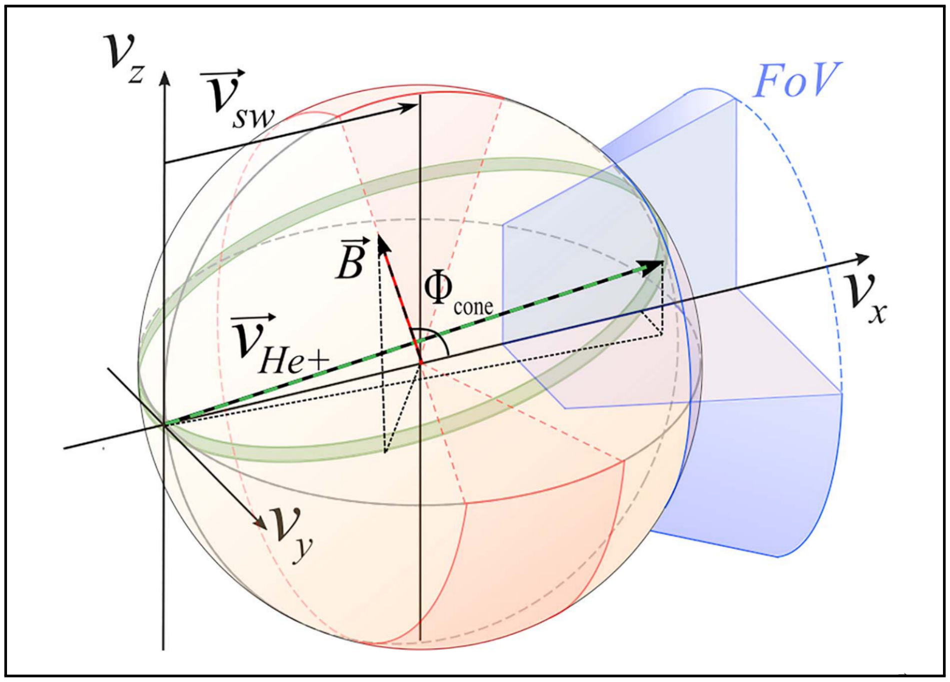


Figure 1.11: The PUI torus, shown as a green band, perpendicular to the magenetic field B and of radius v_{sw}, as seen in the STEREO/PLASTIC instrument field of view (FOV), represented by a blue cone. PUIs will scatter over time into the shell distribution represented in orange and red. Figure from Bower et al. (2019).

The bulk of PUIs in the Heliosphere originate from the interstellar neutral flow across the Heliopause from the LISM, as discussed in Section 1.1.6. As the bulk solar wind flows radially outward from the Sun, interstellar neutrals in the upwind direction between the Sun and the Heliopause have a velocity directly anti-parallel to the solar wind velocity, as shown by the straight black arrow on the righthand side of the upper central panel in Figure 1.12. As those neutrals with higher FIP that are able to penetrate farther into the Heliosphere are focused around the Sun toward the focusing cone on the downwind side, their velocities become increasingly tangential to the solar wind flow, as shown by the curved black arrows originating from the central and bottom right of Figure 1.12. This relative velocity at injection as a function of longitude is demonstrated in Figure 1.12 as well as Figure 1.13, panels a (adapted from Taut et al. (2018)) and b.

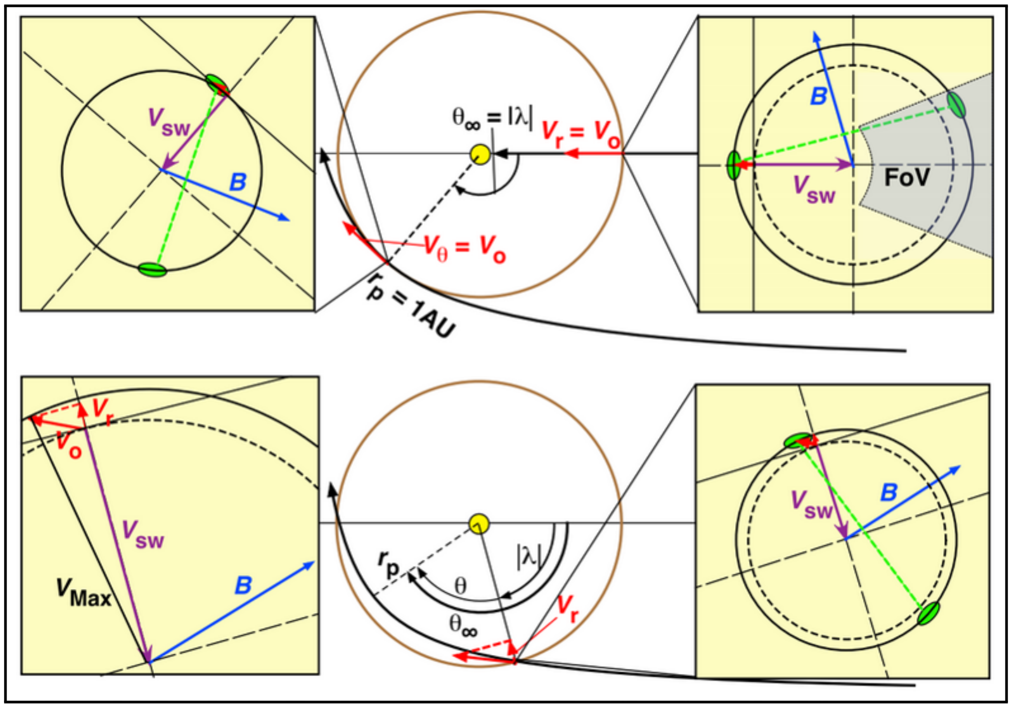


Figure 1.12: A depiction of the relative injection velocities of PUIs under varying solar wind conditions with the PUI torus represented in 2D by the green region perpendicular to the magnetic field, B. Figure from Möbius et al. (2015a).

Three relative injection velocity examples are shown in Figure 1.12 for the case when the focused neutral flow velocity is tangential to the Earth's orbit (upper left), when the flow is anti-parallel to the solar wind on the upwind side (upper right), and a case at a longitudinal angle between the previous two (lower right, with close-up view on lower left). In each case, the gyration of the newly injected PUIs in the 2D projection of the PUI torus in green around the magnetic field, B, is shown relative to the solar wind velocity. Therefore, the relative injection velocity of the newly injected PUIs is dependent upon ecliptic longitude, having a maximum in the upwind direction when the relative velocities of the neutrals and the solar wind are anti-parallel and a minimum in the downwind direction as the relative velocities become increasingly tangential. However, the injection of the PUIs is also dependent on the magnetic field, which is independent of ecliptic longitude. As PUIs are injected perpendicular to the magnetic field, for an instrument with a solar radial look direction, this means that the newly injected PUIs in the torus, projected as a green band in Figure 1.12, will be in the instrument's field of view when the solar wind velocity and magnetic field vectors are relatively perpendicular, as shown in the upper right panel of Figure 1.12.

In Figure 1.13, the velocity distributions of corresponding colors in panel b to panel a demonstrate how the PUI cutoff shifts from a lower velocity on the downwind side to a higher velocity on the upwind side due to the previously mentioned variability in relative velocity angle between the radial solar wind and the incoming neutral flow. This is further shown in Figure 1.13, panels c and d, where 1° velocity distributions of He^+ measured by STEREO A/Plastic are plotted against ecliptic longitude, with a minimum cutoff velocity occurring around 75°, corresponding to the neutral flow direction, and a maximum cutoff velocity occurring around 255°, directly upwind, in ecliptic longitude.

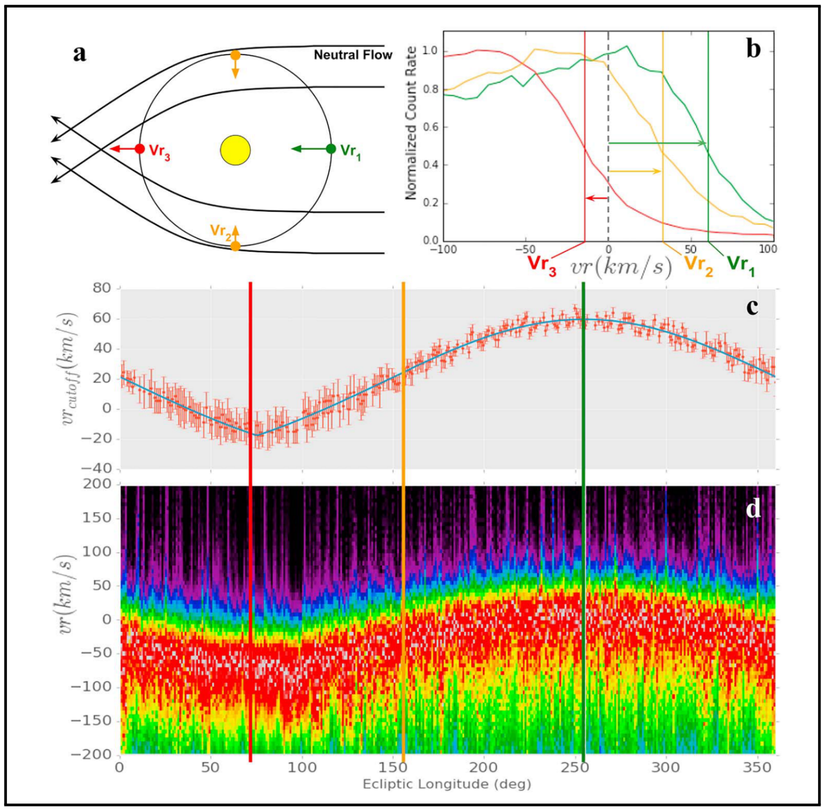


Figure 1.13: Figure demonstrating the shift along the velocity axis of the PUI cutoff with ecliptic longitude due to the relative injection velocity of the neutral flow particles into the PUI population relative to the radially flowing solar wind velocity. Panel a is a diagram of the relative velocities of the neutral flow and the PUIs at varying ecliptic longitudes, panel b shows the shift in the cutoff along the velocity axis, and panels c and d demonstrate the shift in the cutoff over a full 360° orbit in ecliptic longitude. Figure from Bower et al. (2019).

1.3 In Situ Plasma Measurement Techniques

1.3.1 The Evolution of the Time of Flight Ion Mass Spectrometer

Spaceflight ion mass spectrometers have evolved over time from the first flown electrostatic analyzer (ESA) instrument launched on a sounding rocket in 1947, to the modern triple-coincidence TOF ion mass spectrometer with elevation angle selection and azimuthal angular resolution. The original spaceflight mass spectrometers included ESA faraday cups (Reifman & Dow (1949)) and curved plate analyzers (Nagy et al. (1963)). Mariner-2 carried an ESA instrument, the solar plasma experiment (Neugebauer & Snyder (1962)), and the Vela satellites carried ESAs which were used to discover populations of heavy ions in the solar wind, including Vela 3, which was used for the first confirmation of the presence of heavy ions in the solar wind (Bame et al. (1968)). TOF measurements were introduced in the laboratory using the "velocitron" by Cameron & Eggers (1948) and were then incorporated into spaceflight instruments after Gloeckler & Hsieh (1979) introduced the concept of using thinner carbon foils and smaller detectors. This concept was followed by the first spaceflight TOF mass spectrometers, including the CHarge-Energy-Mass Spectrometer (CHEM, (Gloeckler et al. (1985))) and the Suprathermal Energy Ionic Charge Analyzer (SULEICA, (Moebius et al. (1985))) on AMPTE/IRM. These were followed by the Solar Wind Ion Composition Spectrometer (SWICS, (Gloeckler et al. (1992))) on Ulysses, the flight spare for which was also flown on the Advanced Composition Explorer (ACE) spacecraft (Gloeckler et al. (1998)), the PLASma and SupraThermal Ion Composition experiment (PLASTIC, (Galvin et al. (2008))) on the Solar Terrestrial Relations Observatory (STEREO) spacecraft, the CAsinni Plasma Spectrometer (CAPS, (Young et al. (2004))) on Cassini, the Fast Imaging Plasma Spectrometer (FIPS, (Andrews et al. (2007))) on MESSENGER, and the Heavy Ion Sensor (HIS, (Owen et al. (2020))) on Solar Orbiter.

1.3.2 Time of Flight Ion Mass Spectrometers

In situ space plasmas can be sampled using instruments on board spacecraft, such as the Solar Wind Ion Composition Spectrometer on the ACE spacecraft (ACE/SWICS) launched in 1998 and the Heavy Ion Sensor on the Solar Orbiter spacecraft (SO-HIS) launched in 2020. ACE/SWICS and SO-HIS are both *in situ* time-of-flight (TOF) triple-coincidence ion mass spectrometers. These instruments utilize an electrostatic analyzer (ESA) to select ions by energy-per-charge (E/q). The unique capabilities of TOF ion mass spectrometers allow us to separate charge-states of helium and heavier ions using TOF and energy, as described below. By taking these specific measurements of the *in situ* plasma, we are able to measure the specific composition of heavy ions and their charge-states and create 3D velocity distribution functions. These measurements of heavy ions allow us to understand physical processes in the Heliosphere, as they are tracers of mass and charge dependent processes.

These instruments step through a range of E/q bands by setting voltages on the curved concentric ESA plates. By creating an electrostatic field in the absence of a magnetic field, an ion is accelerated parallel, in the case of positive ions, or anti-parallel, in the case of negative ions, to the field by the Lorentz force. An ion moving perpendicular to the field will therefore experience a force perpendicular to its motion and undergo centripetal motion. The radius of curvature of the motion caused by the electrostatic field is proportional to E/q and given, as derived by Gilbert (2008) as:

$$\mathbf{F}_c = m\mathbf{a} = \frac{m(v\sin\theta)^2}{r}\langle -\hat{\mathbf{r}}\rangle \tag{1.7}$$

$$\frac{m(v\sin\theta)^2}{r}\langle -\hat{\mathbf{r}}\rangle = q\mathbf{E} \tag{1.8}$$

$$\Rightarrow r = \frac{m(v\sin\theta)^2}{q|\mathbf{E}|}$$

$$\Rightarrow r = \frac{2}{|\mathbf{E}|}\frac{E}{q}\sin^2\theta \tag{1.9}$$

where θ is the angles between the ion's velocity vector $\mathbf{v}$ and the electric field vector $\mathbf{E}$. An ion whose velocity is parallel to the electric field will be accelerated in a straight line, while the force exerted on the particle becomes increasingly centripetal with increasing θ until only a centripetal force is exerted when the velocity vector is perpendicular to the electric field vector. Therefore, a curved ESA with electric field applied perpendicular to the particle's motion at all times will cause the particle to move in a circular motion and pass through the concentric plates.

After passing through the ESA section, the ions are measured in a TOF chamber using three detections, or coincidences, namely a start timing signal, an energy detection, and a stop timing signal, comprising the triple-coincidence sequence. Together, these measurements yield E/q, energy, and TOF, and can be used to compute mass, charge, and velocity. These instruments provide velocity distribution functions as well as plasma composition information.

The mass (M) in amu, charge (q) in e, mass-per-charge (M/q) in amu/e, incident energy (E_0) in keV, and post-acceleration voltage (V_A) in kV are given by Gilbert et al. (2014) as:

$$(\frac{E_{meas}}{\alpha} = \frac{1}{2}M(\frac{d}{\tau})^2 \tag{1.10}$$

$$(\frac{E}{q} + V_A - \frac{E_{loss}}{q}) = \frac{1}{2}(\frac{M}{q})(\frac{d}{\tau}^2) \tag{1.11}$$

$$q = \frac{E_{meas}/\alpha}{(E/q + V_A - E_{loss}/q)} \tag{1.12}$$

$$E_0 = \frac{q}{E/q} \tag{1.13}$$

where E_{meas} is the energy measured by the solid state detector, α is the nuclear defect in the solid state detector (Ipavich et al. (1978)), d is the distance traveled by the particle through the TOF chamber, τ is the travel time, and E_{loss} is the energy loss experienced by the particle when passing through the carbon foil. These equations have been updated for energy loss considerations from the equations presented by Gloeckler et al. (1998).

1.3.3 The Solar Orbiter Heavy Ion Sensor (SO-HIS)

SO-HIS, shown in Figure 1.14, is a triple-coincidence TOF ion mass spectrometer on board Solar Orbiter, a collaborative mission between ESA and NASA that was launched on 9 February, 2020, as shown in Figure 1.15. SO-HIS is intended to measure ionic and elemental composition and 3D velocity distribution functions of ions in the range He – Fe in the E/q range 0.5 – 80 keV/e with charge-states from He^{+} to Fe^{20+}, and specifically the bulk solar wind in the range 0.5 – 18 keV/e, along with the suprathermal components of the solar wind in He, C, O, and Fe up to at least 60 keV/e, as well as PUIs. SO-HIS selects for E/q by stepping through 64 energy steps, with an energy resolution of 6 – 10%. One unique defining feature of SO-HIS is that it can select for elevation angle with a resolution of $<3.5°$ in addition to E/q and can resolve angular components in both elevation and azimuth. SO-HIS uses 16 steps in elevation in the range $\pm17°$ in elevation to sample the distribution out of the ecliptic and has a continuous azimuthal field of view of -30° to +66°.

SO-HIS and its SIMION ion optical model are discussed in more detail in Chapter 2. In this chapter, the cross calibration of SO-HIS with its ion optical model is presented along with a function of geometric factor in the elevation angle range -15° to +18° $\pm3°$ for standard solar wind conditions. Geometric factor is a feature of the instrument's useful particle intake geometry, representing the effective sampling area of the instrument with considerations for both mechanical and electrostatic effects. The geometric factor can be used to convert instrument counts to physical quantities. Equation 1.14, derived by Stakhiv (2016), shows how to convert between counts and the geometric factor.

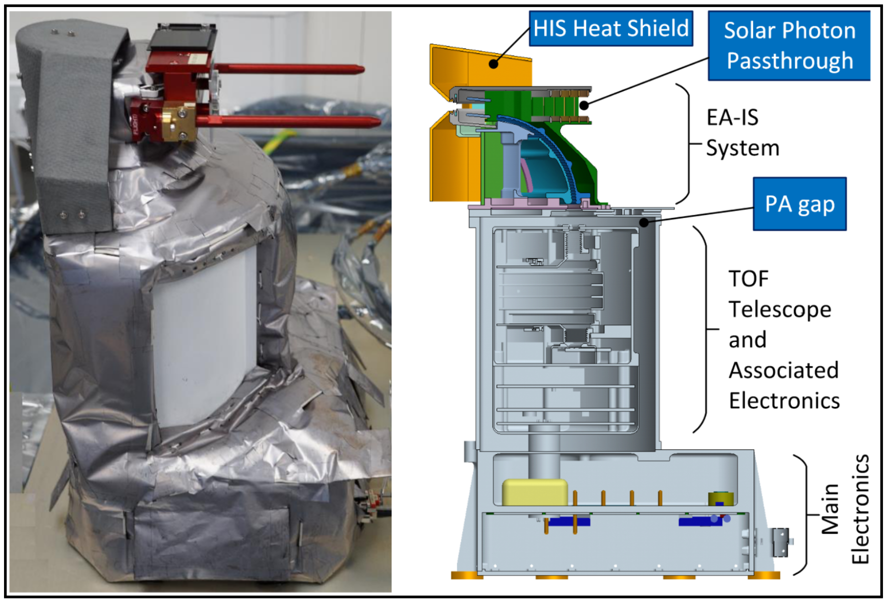


Figure 1.14: Left: Photograph of the SO-HIS instrument. Right: Labeled diagram of the SO-HIS instrument in the corresponding orientation. Figure from Owen et al. (2020).

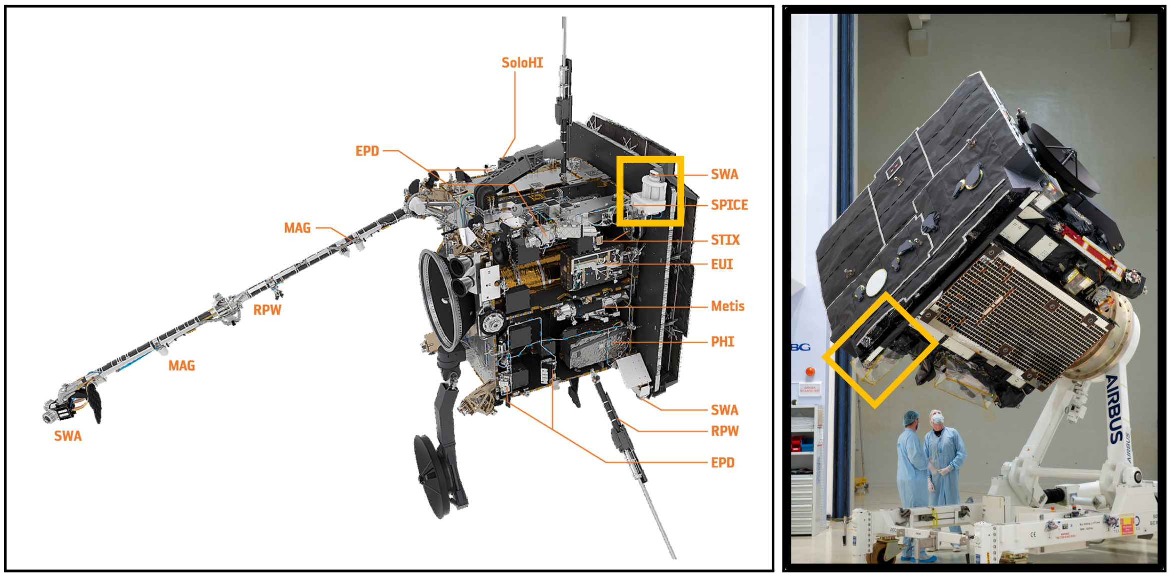
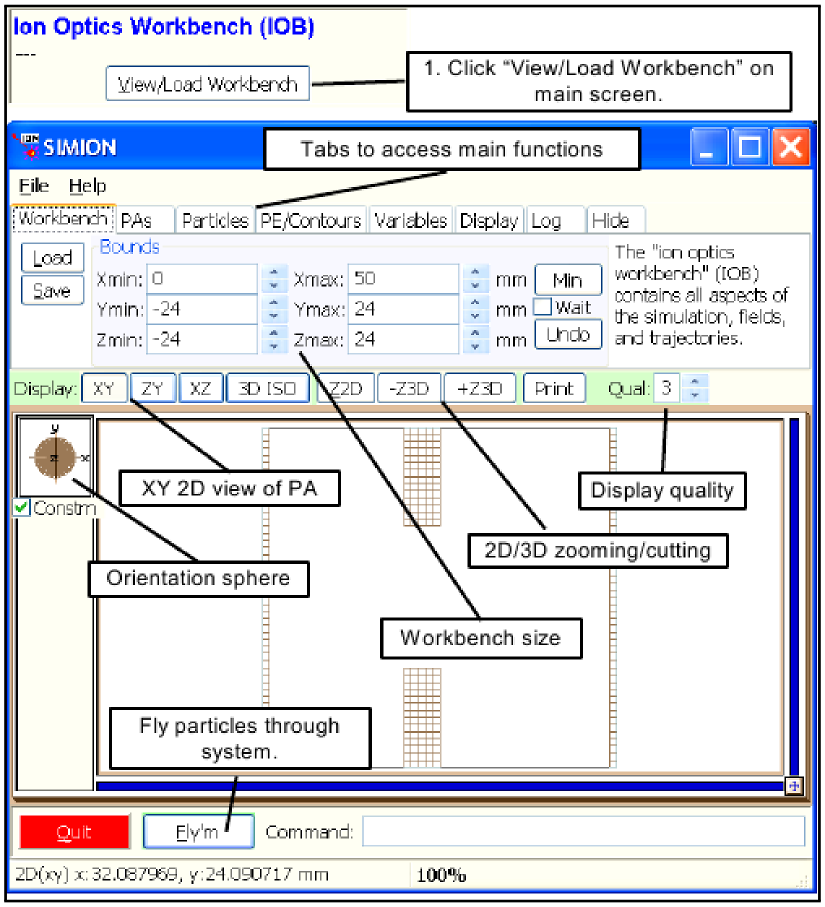


Figure 1.15: Left: Model of the Solar Orbiter spacecraft. SO-HIS is highlighted peeking out of a cutout at the top right corner of the spacecraft heat shield. Figure adapted from Müller et al. (2020). Right: Photograph of the Solar Orbiter spacecraft from a different view. SO-HIS is highlighted peeking out of a cutout at the bottom left of the spacecraft heat shield. Figure adapted from ESA (2019).

$$\Delta C = \frac{1}{2}\,\varepsilon\, G(\mathrm{v},\theta,\phi)\, f(\mathrm{v},\theta,\phi)\, v^4\, \Delta t \tag{1.14}$$

where ΔC is counts, ε is the efficiency of the TOF section, $G(\mathrm{v},\theta,\phi)$ is the geometric factor as a function of velocity, elevation angle, and azimuth angle, $f(\mathrm{v},\theta,\phi)$ is the distribution function as a function of velocity, elevation angle, and azimuth angle, v is velocity, and Δt is time.

1.3.4 Ion Optical Modeling in the SIMION Environment


Ion Optics Workbench (IOB)
View/Load Workbench
1. Click "View/Load Workbench" on main screen.
SIMION
Tabs to access main functions
File Help
Workbench PAs Particles PE/Contours Variables Display Log Hide
Load
Save
Bounds
Min
Wait
Undo
Display: XY ZY XZ 3D ISO Z2D -Z3D +Z3D Print Qual: 3
Constrn
XY 2D view of PA
Display quality
2D/3D zooming/cutting
Orientation sphere
Workbench size
Fly particles through system.
Quit
Fly'm
Command:
100%

Xmin: 0 Xmax: 50 mm
Ymin: -24 Ymax: 24 mm
Zmin: -24 Zmax: 24 mm

The "ion optics workbench" (IOB) contains all aspects of the simulation, fields, and trajectories.

2D(xy) x:32.087969, y:24.090717 mm

Figure 1.16: View displaying certain basic SIMION controls. Figure from the SIMION Version 8.0 User's Manual (Manura & Dahl (2007)).

Spaceflight TOF ion mass spectrometers are complex instruments that depend on the interplay of various instrument sections and components from the instrument aperture to the deflectors, ESA, post-acceleration, carbon foil and electron suppression grid, and the TOF chamber, including additional deflectors, the SSD, and the MCPs in the case of SO-HIS. It is not always possible or practical to solve analytically for the complicated effective combined geometry of interacting electric fields from fringe fields at the aperture and at the edges of the deflectors to the guiding fields in the TOF chamber. Therefore, we create an ion optical model (IOM) in the SIMION computational program version 8.0 (Manura & Dahl (2007)) to characterize the SO-HIS instrument response using Monte Carlo particle simulations. The SIMION program is shown in Figure 1.16, and sample potential arrays are shown in Figure 1.17. The SO-HIS SIMION IOM is shown in Figure 1.18.

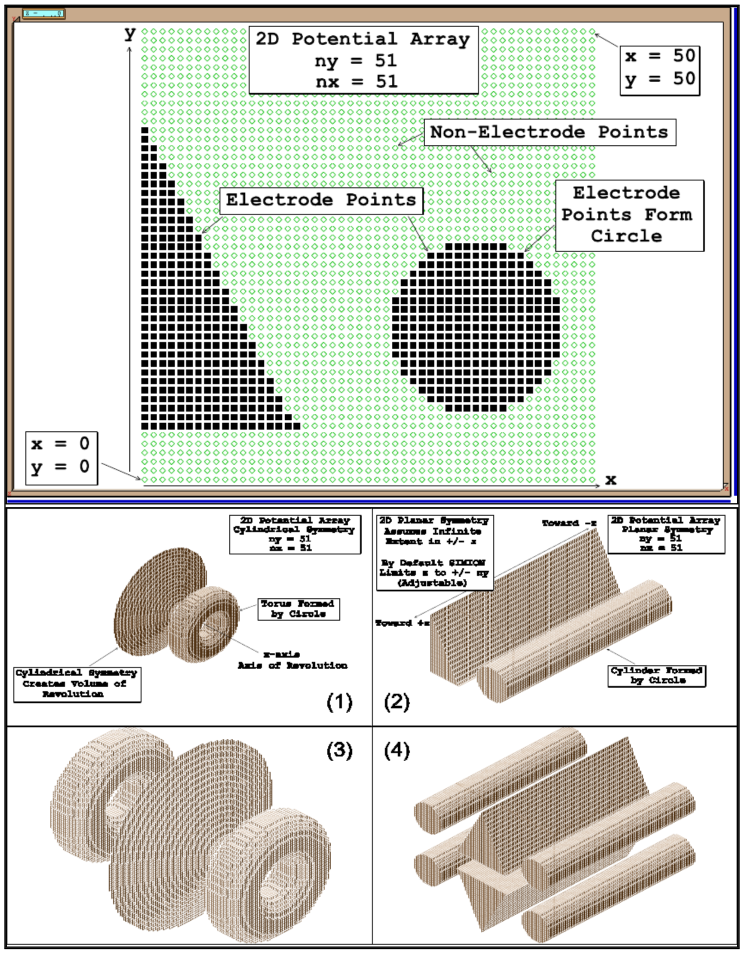


Figure 1.17: Top: Sample of 2D potential array creation in SIMION. Bottom: Sample of 3D potential arrays created using mirroring in SIMON. Figures from the SIMION Version 8.0 User's Manual (Manura & Dahl (2007)).

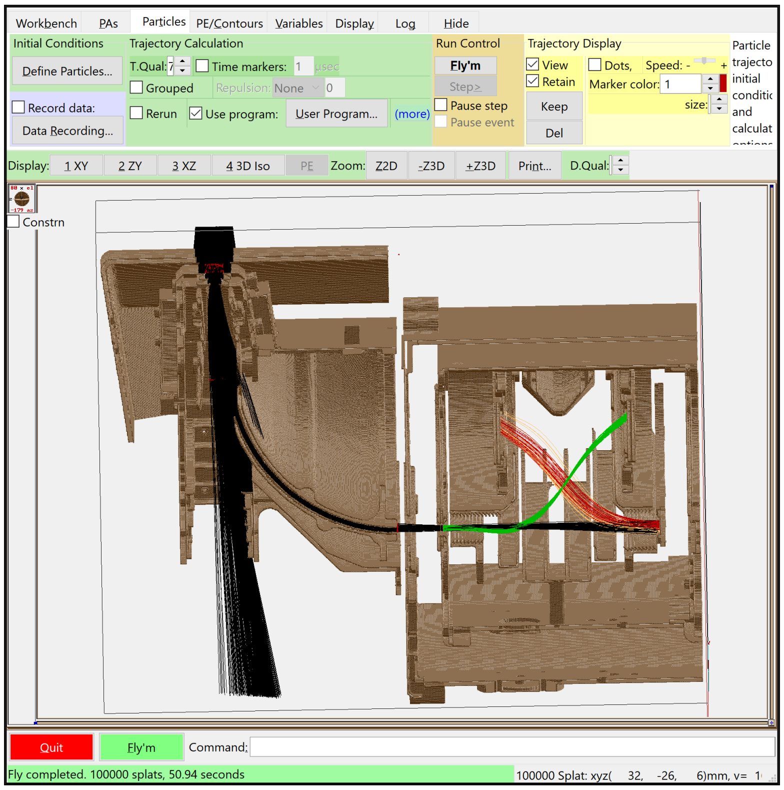
In a Monte Carlo particle simulation, a random set of initial particles are generated given a set of start parameters and instrument configuration, including number and start location of particles and particle energy and angular specifications, as well as instrument parameters such as voltages, which are set based upon desired E/q and angular acceptance. SIMION virtually flies particles through the IOM, tracking and recording their parameters at set locations and conditions, by solving for the electric potentials using the Laplace equation at a given resolution between electrodes, defined by potential arrays. The Laplace equation is given by the SIMION Version 8.0 User's Manual (Manura & Dahl (2007)) as:

$$\nabla^2 V = \nabla \cdot \nabla V = 0 \tag{1.15}$$

$$-\nabla V = -(\partial V/\partial x)i - (\partial V/\partial y)j - (\partial V/\partial z)k = E \tag{1.16}$$

$$\nabla^2 V = \nabla \cdot E = \partial E_x/\partial x + \partial E_y/\partial y + \partial E_z/\partial z = 0 \tag{1.17}$$

This equation constrains electrostatic potential fields to a no space-charge condition and is used by SIMION to compute electrostatic potential fields.



Fly completed. 100000 splats, 50.94 seconds

100000 Splat: xyz(32, -26, 6)mm, v= 1

Figure 1.18: An exploded 3D isometric view of the SO-HIS instrument IOM mostly seen in the z-x plane. Initial particle tracks are shown in black, while start secondary electron tracks are shown in green, stop secondary electron tracks are shown in red, and tracks of particles which miss a detector are shown in yellow. A more detailed description of the SO-HIS SIMION IOM Monte Carlo simulation and a labeled instrument diagram are given in Chapter 2.

1.4 Science Questions

This thesis addresses the major science question: **How does the Heliosphere — Local Interstellar Medium interaction evolve over time?**
This question is broken down into the following sub-questions:

1. **To what precision can a yearly inflow direction of the interstellar wind through the Heliosphere be measured using PUI measurements in the focusing cone and crescent?**

2. **Is there evidence of variation in the longitudinal interstellar neutral inflow direction through the Heliosphere over an 11-year solar cycle?**

   It has taken decades for the Voyager I and II spacecraft to even approach the Heliopause, and the instruments on board are well outside their intended mission operations. While future *in situ* missions such as an Interstellar Probe are proposed to explore the LISM, it is still under debate what trajectory such a mission should take out of the Heliosphere, and it would be intractable to expect an abundance of interstellar measurements made from a multitude of regions in the LISM in the near future. Yet, as the Heliosphere is heavily influenced by interstellar interactions, it is crucial to our understanding to study these interactions using accessible measurements. In the future, we will greatly enhance our understanding of these interactions using a combination of "remote" and *in situ* and observations of the LISM by utlizing both missions like ACE at Earth's Lagrange 1 point and an Interstellar Probe in the LISM. However, we are already able to begin studying properties of the Heliosphere — interstellar interactions by observing features such as the helium focusing cone and crescent using measurements taken within the Heliosphere. Using such measurements, we may both measure and track the longitudinal flow direction, λ_{ISN}, of the interstellar wind through the Heliosphere using the PUI count distributions and signatures of the focusing cone and crescent.

3. **How can new instrument designs and measurement techniques enhance our understanding of the Heliosphere — interstellar interactions?**

   PUIs are a crucial source of information about the plasma in the Heliosphere originating from various sources ranging from the Sun to planets, comets, dust, and the LISM. While these measurements have already greatly enhanced our understanding of Heliospheric plasma interactions, there is more that can be learned from expanding this data set using new measurements in new space environments, such as with SO-HIS and other future missions.

   Future observations from SO-HIS will not only provide an expanded data set that will allow us to study the Heliosphere — interstellar interactions over a longer time period, but will also provide enhanced PUI measurements. The higher time cadence of HIS measurements, with a low cadence Normal Mode of 300 s, Normal Mode of 30 s, and a Burst Mode of 4 s, along with the added feature of angular resolution in elevation, with 16 elevation steps in the range $\pm 17°$ with a resolution of $<3.5°$ and a continuous azimuthal acceptance in the range -30° to +66°, will enable more accurate comparison with solar IMF conditions, allowing a much greater potion of the data to be utilized more accurately when making considerations for PUI transport, which is described in Chapter 3. Additionally, Solar Orbiter's unique orbit into $\sim$2.8 AU and up to $\sim$30° out of the ecliptic in the extended mission will provide PUI measurements from HIS at varying locations and latitudes, enabling future studies of the Heliosphere — interstellar interactions, by mapping the 3D structures of the helium focusing cone and crescent, which are discussed in detail in Chapters 3 and 4.

1.5 Thesis Overview

While the Sun creates and shapes the Heliosphere, affecting the space plasma environment throughout the Solar System, composition and processes of the Heliospheric plasma are also appreciably influenced by the interactions between the Heliosphere and the LISM. This thesis focuses on characterizing these interactions using PUI measurements, namely through calibration of novel new instrumentation and the study of the evolution of the longitudinal inflow direction of the interstellar wind through the Heliosphere. This thesis also presents challenges associated with current PUI data sets and measurement techniques and explores future improvements to analysis and measurement of PUIs.

1.5.1 Cross-Calibration and Performance Assessment of the Solar Orbiter Heavy Ion Sensor with its Ion Optical Model

SO-HIS is an *in situ* TOF triple-coincidence ion mass spectrometer with elevation resolution on board Solar Orbiter, a joint ESA/NASA mission launched on 9 February, 2020 to study how the Heliosphere is created and shaped by the Sun. Solar Orbiter provides a unique opportunity to observed the Sun away from its equator, reaching orbits as close as $\sim$0.28 AU with inclinations out of the ecliptic up to $\sim$30° in its extended mission, enabling new measurements of critical, yet largely unexplored regions of the Heliospheric plasma distribution. SO-HIS plays a crucial role in characterizing the plasma environment by measuring ionic and elemental composition and 3D velocity distribution functions of ions in the energy-per-charge range 0.5 – 80 keV/e, specifically of the bulk solar wind species He – Fe in the energy range 0.5 – 18 keV/e with the charge-states He^{+} to Fe^{20+} and suprathermal components with energies up to at least 60 keV, as well as PUIs. Chapter

2 discusses the cross-calibration of SO-HIS with its ion optical model and presents a function for the spaceflight instrument's geometric factor under standard solar wind conditions.

1.5.2 Determining the Interstellar Wind Longitudinal Inflow Evolution Using Pickup Ions in the Helium Focusing Cone

The interstellar flow through the Heliosphere plays a significant role on the Heliospheric plasma composition and dynamics. Discussion has arisen regarding the question of the evolution of the longitudinal flow direction over time (Frisch et al. (2013, 2015); Lallement & Bertaux (2014); Wood et al. (2015)). Chapter 3 presents measurements of the flow direction, λ_{ISN}, using a new technique combining PUI measurements in the helium focusing cone with the location of the PUI cutoff in the ACE/SWICS He^+ double-coincidence averaged combined count distribution and discusses the flow evolution over an 11-year solar cycle over the years 1999 through 2009.

1.5.3 Implications of Transport and Challenges in Interstellar Flow Measurements Using Pickup Ions

While proving an instrumental data set in the study of Heliosphere — interstellar interactions, PUI measurements present challenges from an analysis and measurement perspective. Considerations must be made for PUI transport as well as conditions due to solar activity, such as flux enhancements and magnetic field disturbances from CMEs and compression regions due to stream-stream interaction regions, which may appreciably limit the accuracy of the relevant data for the determination of the interstellar parameters. Additionally, current measurement techniques do not always provide ideal conditions for sampling the PUI distribution. Chapter 4 discusses challenges in the use of PUI measurements along with ongoing efforts to characterize the evolution of the interstellar flow direction through the Heliosphere using the PUI cutoff method to measure the orientation of the upwind crescent using the ACE/SWICS He^+ double-coincidence data set, as well as future considerations for improved methods in PUI analysis and measurement.

CHAPTER 2

Cross-Calibration and Performance Assessment of the Solar Orbiter Heavy Ion Sensor with its Ion Optical Model

This chapter is being prepared for submission to *The American Institute of Physics: Review of Scientific Instruments* and was written in collaboration with Susan T. Lepri, Jim M. Raines, Frederic Allegrini, Connor P. Raines, Austin N. Glass, Jason A. Gilbert, and Stefano Livi.

2.1 Abstract

Launched on 9 February, 2020 as a collaborative effort between ESA and NASA, Solar Orbiter is an instrumental mission utilizing both remote sensing and *in situ* measurements to study how the Sun creates and shapes the Heliosphere. Solar Orbiter provides a unique opportunity to observe the Sun away from its equator, reaching orbits as close to the Sun as $\sim$2.8 AU at up to around 30° out of the ecliptic in the extended mission. The Solar Orbiter Heavy Ion Sensor (SO-HIS) is an *in situ* triple-coincidence ion mass spectrometer intended to measure ion and elemental composition and 3D velocity distribution functions of ions in the range of He – Fe with charge-states from He^{+} to Fe^{20+} in the full energy-per-charge (E/q) range 0.5 – 80 keV/e. The instrument measures the bulk solar wind in the range 0.5 – 18 keV/e and the suprathermal solar wind components of He, C, O, and Fe up to at least 60 keV/e as well as pickup ions. SO-HIS uses 64 E/q steps with an energy resolution of 6 – 10% and 16 elevation steps in the range $\pm17°$ with an angular resolution of $<3.5°$ and has a continuous azimuthal acceptance in the range -30° to +66°. To determine instrument characterization such as the geometric factor (GF) for any instrument configuration, we create a high resolution ion optical model (IOM) of the SO-HIS instrument from its CAD model using the SIMION software. We present steps to validate SIMION outputs against laboratory measurements as well as our methodology for calculating the GF. We then present a function of the instrument's

GF under standard solar wind conditions. We report here on the first cross-calibration for SO-HIS between laboratory measurements of the flight model and its IOM.

2.2 Introduction

The Solar Orbiter spacecraft was launched on 9 February, 2020 as a collaborative effort between ESA and NASA, enabling fundamental new scientific measurements (Zouganelis et al. (2020)). Solar Orbiter, a mission which provides a unique opportunity to observed the Sun away from its equator, was designed for the overall science mission to study how the Heliosphere is created and shaped by the Sun, providing capabilities to trace features of the solar wind to their origins on the Sun and to observe physical properties that occur in the solar wind. Solar Orbiter's payload comprises instruments to take measurements of the inner Heliosphere and the solar atmosphere and includes both remote sensing and *in situ* instruments.

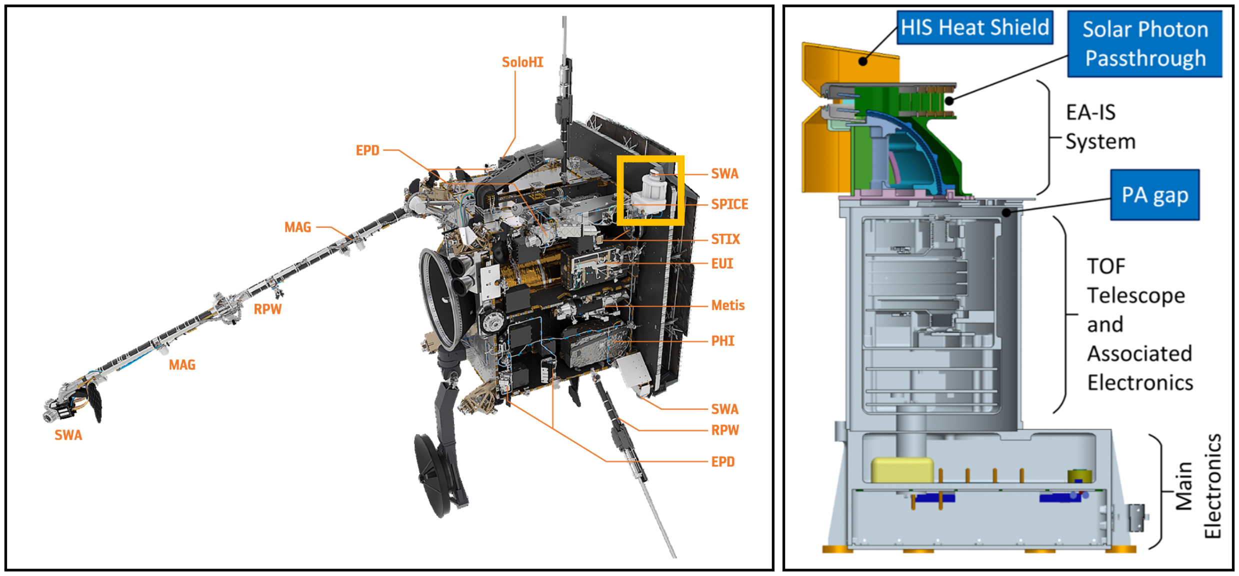


Figure 2.1: Left: Model of the Solar Orbiter spacecraft. SO-HIS is highlighted peeking out of a cutout at the top right corner of the spacecraft heat shield. Figure adapted from Müller et al. (2020). Right: Labeled diagram of the SO-HIS instrument. Figure adapted from Owen et al. (2020).

Solar Orbiter's unique view of the Sun, which will come as close as $\sim$0.28 AU and up to an inclination of tens of degrees, up to $\sim 30^\circ$ out of the ecliptic in the extended mission, allows for novel science opportunities for instruments such as the Solar Orbiter Heavy Ion Sensor (SO-HIS), an *in situ* triple-coincidence ion mass spectrometer, which directly samples the *in situ* plasma (Owen et al. (2020)). SO-HIS, shown in Figure 2.1, comprises 3 major sections, namely the Electrostatic Analyzer with Ion Steering (EA-IS), the time-of-flight (TOF) section, and the Main Electronics

Box. SO-HIS takes direct measurements of particle E/q, Energy, and TOF, which can be used to calculate mass, charge, and velocity, and yields 3D velocity distribution functions. To convert the instrument counts from SO-HIS to physical quantities, it is crucial to understand the characterization of the instrument, namely its geometric factor (GF), which relates the instrument count rate to the particle distribution of the sampled plasma. The GF is a qualifying feature of the instrument's ability to sample a given region and is a feature of the instrument's useful particle intake geometry, essentially an effective collector area considering both mechanical and electrostatic effects. While another critical quantity related to the instrument's characterization is detection efficiency, particularly related to particle transmission and the efficiencies of the solid state detector (SSD) used for particle energy detection and the micro-channel plates (MCPs) used for timing signals, this work focuses on the SO-HIS GF calibration.

The GF of an instrument is affected by the instrument configuration, including variables such as E/q step and elevation acceptance as well as azimuth of the incoming particles, so while an instrument's GF is often summarized by a single value, to achieve the most accurate outputs for scientific data, it is ideal to characterize a function of the instrument's GF for any configuration. Determining the instrument's GF under different configurations requires measurements under known conditions, such as using an adjustable ion beam in a laboratory; however, only limited measurements can be made in a laboratory and are restricted both by laboratory equipment capabilities and allotted calibration time before launch. Therefore, having a properly calibrated ion optical model (IOM), which has been validated using a cross-calibration with laboratory measurements, provides a uniquely useful tool for determining the instrument's characterization, including the GF, under arbitrary conditions for the flight model on board the spacecraft. We present our procedure and conclusions for creating, verifying, and assessing a SO-HIS IOM using the SIMION environment as well as our findings of the GF for standard solar wind conditions.

2.3 Background

2.3.1 The Solar Orbiter Heavy Ion Sensor (SO-HIS)

The Solar Orbiter Heavy Ion Sensor (SO-HIS) is an *in situ* triple-coincidence ion mass spectrometer that is part of the Solar Wind Plasma Analyzer (SWA) Suite onboard Solar Orbiter, along with the Electron Analyzer System (EAS) and the Proton and Alpha Sensor (PAS). SO-HIS directly measures E/q, Energy, and TOF and produces data products including elevation and azimuthal angle, mass, charge, density, velocity, and thermal temperature, providing measurements of the solar wind plasma composition as well as 3D velocity distribution functions, specifically of solar wind heavy ions as well as pickup ions (PUIs) and major suprathermal components of the solar wind.

2.3.1.1 Instrument Measurement Details

SO-HIS directly samples the *in situ* solar wind to measure ionic and elemental composition as well as 3D velocity distribution functions by peeking out through a cutout in the corner of the Solar Orbiter heat shield, as shown in Figure 2.1, and is therefore equipped with an independent heat shield with a slit entrance leading to the instrument aperture. The instrument scans the full E/q range of 0.5 – 80 keV/e in 64 steps with native time resolutions including a low cadence Normal Mode of 300 s, Normal Mode of 30 s, and a Burst Mode of 4 s. SO-HIS is intended to measure ions in the bulk solar wind from He – Fe in the E/q range of 0.5 – 18 keV/e with charge-states from He^{+} to Fe^{20+} as well as suprathermals, specifically the species He, C, O, and Fe, with E/q up to at least 60 keV/e, and PUIs. SO-HIS scans an elevation field of view of about $\pm17°$ out of the ecliptic in 16 steps and has a continuous azimuthal field of view of about -30° to +66° in the ecliptic off the Sun-spacecraft line to account for solar wind aberration from spacecraft motion and for greater sampling of the PUI and suprathermal components in the solar wind. (Owen et al. (2020))

SO-HIS consists of 3 major sections, namely the EA-IS, which guides ions based on elevation and E/q and is grounded to an outer housing, an isolated TOF section floated at the post-acceleration voltage of up to -25 keV, and the supporting electronics in the main electronics box, containing the low voltage power supply (LVPS), the high voltage power supply (HVPS) used for the post-acceleration, and the related circuitry and boards, including the Command and Data Handling Board (C&DH), which runs the flight software, including the main sensor controls.

2.3.1.2 The Electrostatic Analyzer with Ion Steering (EA-IS) Section

As ions enter the instrument through the entrance aperture, they are affected by the electric fields from the voltages on the deflectors and electrostatic analyzer (ESA) plates. Depending on their initial conditions, the ions will either pass directly through the top hat without entering the ESA, will collide with an instrument wall, or will be guided through the ESA to the TOF chamber. The instrument cycles through 64 E/q steps, and if a particle with the correct E/q enters the instrument at a given time step, it will be guided to the the TOF chamber.

The EA-IS comprises a hemispherical top-hat design including the concentric ESA plates used for E/q selection, with an energy resolution of 6 – 10%, and the bottom, top, and top-cap deflectors used for ion steering and elevation selection. The instrument steps through 16 elevation steps with an acceptance of $\pm17°$ above and below the ecliptic and an angular resolution of $<3.5°$. (Owen et al. (2020))

2.3.1.3 The Time of Flight Chamber

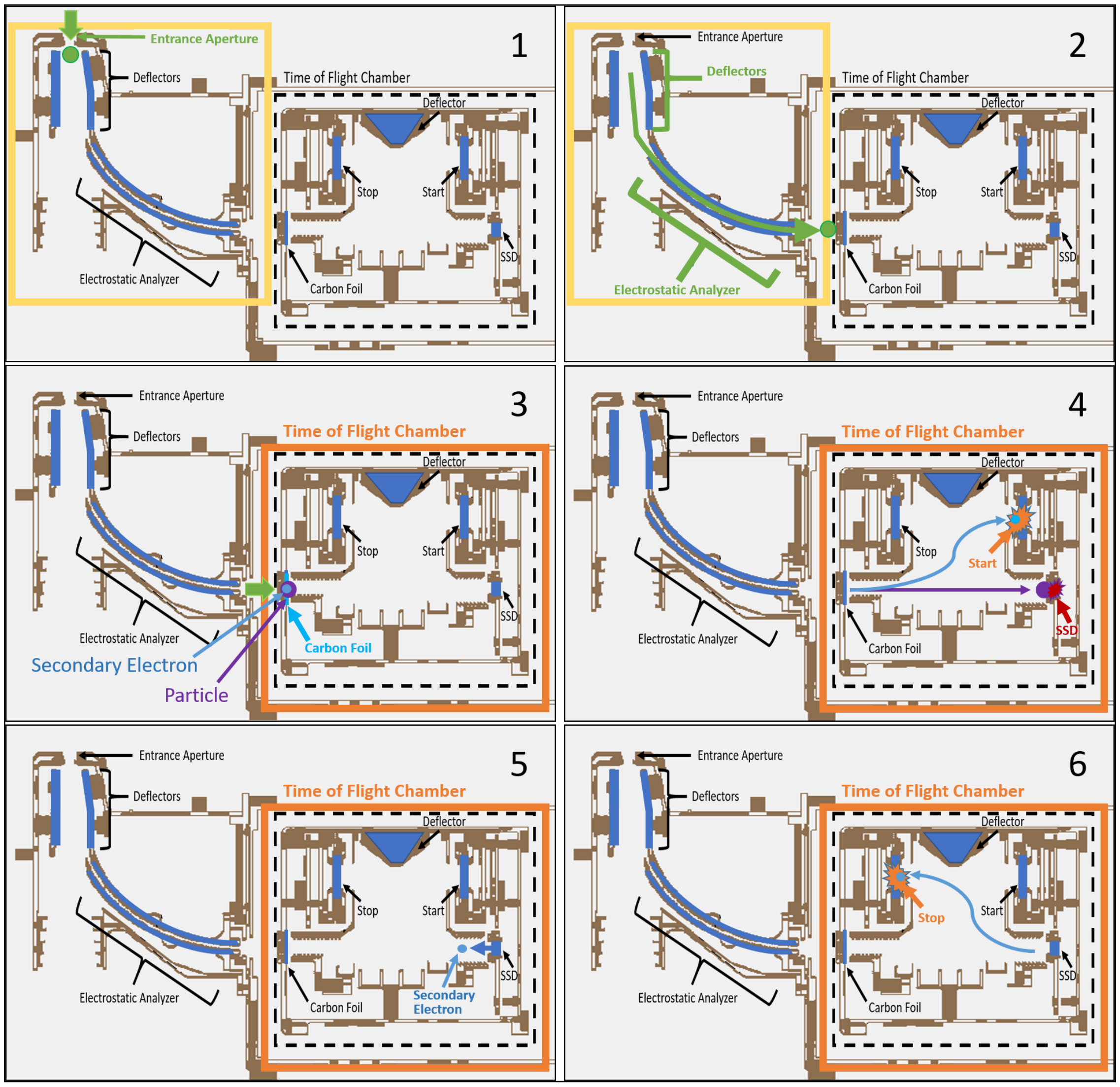


Figure 2.2: A triple-coincidence sequence for the SO-HIS instrument in the SIMION environment. 1. An ion enters the EA-IS Section through the instrument aperture. 2. An ion with the correct E/q is guided by the deflectors and ESA to the TOF section. 3. After being accelerated by a fixed post-acceleration voltage, the ion enters the TOF section through a thin carbon foil, generating a secondary electron. The ion is likely neutralized by the carbon foil. 4. The secondary electron is guided by electric fields in the TOF chamber to the start MCP, where a start timing signal is made. The original particle continues to the solid state energy detector, where a direct energy reading is made. 5. A secondary electron is emitted from the SSD by the particle's impact. 6. The secondary electron is guided by electric fields in the TOF chamber to the stop MCP, where a stop timing signal is made, completing the triple-coincidence sequence.

Ions guided through the ESA will enter the TOF chamber through a thin carbon foil after being accelerated by a fixed post-acceleration voltage of up to -25 keV to aid in overcoming the energy detection threshold of the SSD. As an ion passes through the carbon foil, it generates secondary electrons, which are guided by the electric fields in the TOF chamber to the start MCP, where they trigger a start timing signal. The ion experiences charge exchange in the carbon foil and continues to the energy detecting SSD. The charge-state of the particle as it exits the carbon foil is affected by the composition of the foil (Phillips (1955)), though especially for low energy ions, such as PUIs, the charge exchange processes become adiabatic near the foil surface, and the majority of ions are neutralized (Allegrini et al. (2016) and references therein). For example, in the energy range used in our study, $\sim$60% of hydrogen atoms exit the foil as neutral hydrogen, $\sim$40% as H^+, and a very small fraction as H^-. When the particle impacts the SSD, an energy reading is made, and secondary electrons are again emitted. These secondary electrons are guided by the electric fields in the TOF chamber to the stop MCP, where they trigger a stop timing signal, completing the sequence, as shown in Figure 2.2. These three detections, or coincidences, namely the start, stop, and energy detections, provide the three signals needed for a triple-coincidence detection. This full sequence provides E/q, Energy, and TOF, which can be used to compute mass, charge, and velocity.

In the flight model, if the particle does not overcome the detection threshold of the SSD, no energy reading will be made, resulting instead in a double-coincidence TOF detection. Triple-coincidence detections are therefore a subset of double-coincidence detections. In our SIMION simulation, we treat all particles that land on the SSD and cause both start and stop timing signals as physical triple-coincidence detections for simulation purposes. Additionally, a single secondary electron is simulated in each instance.

2.3.2 Calculation of Geometric Factor

The GF is calculated according to Equation 2.1 derived from Collinson et al. (2012):

$$G = \frac{C\,\Delta Y_{SS}\Delta Z_{SS}\,\bar{E}\cos^2\bar{\theta}\,\Delta E\,\Delta\theta\,\Delta\phi}{N\,E_0^2} \tag{2.1}$$

where G is the GF, C is the number of particles which penetrate past the carbon foil, $\Delta Y_{SS}\Delta Z_{SS}$ is the source surface area for initial ions, $\bar{E}$ is the mean initial ion energy, $\bar{\theta}$ is the mean initial ion elevation, ΔE is the initial ion energy width, $\Delta\theta$ is the initial ion elevation width, $\Delta\phi$ is the azimuthal angular width taken as $\pm3°$ = 6°, N is the number of initial ions, and E_0 is the initial ion peak energy.

As the laboratory GF values are calculated using the total flux just after the carbon foil, we compute the GF from a SIMION run as a summation of GFs over a 10x10 grid in energy and elevation for particles which successfully pass through the carbon foil. Each sub-grid unit in energy and elevation is correlated to the relevant sub-grid of particles generated at the source surface. This is achieved by identifying all of the particles which penetrate the carbon foil and determining a minimum and maximum bound for their initial energies and elevation angles. We then subdivide those bounds into a 10x10 grid in energy and elevation, and use each combination of energy and elevation range in those subdivisions as one grid cell in the GF calculation for a total of 100 values, as:

$$G_{Run} = \sum_{E,elv} G_{E,elv} \tag{2.2}$$

where G_{Run} is the GF calculated for a given run, E is the current energy sub-grid range, elv is the current elevation sub-grid range, and $G_{E,elv}$ is the GF calculated for the current energy-elevation sub-grid range. All parameters in a sub-grid calculation relate to the post-carbon-foil measurements and initial ions corresponding to the current energy-elevation range.

For each sub-grid energy-elevation range, we take the total number of ions, N, to be all initial ions generated at the source surface in the current energy-elevation range, and the number of post-carbon-foil ions, C, to be all ions whose initial energies and elevations are in the current energy-elevation range and which penetrate past the carbon foil. We additionally scale the number of post-carbon-foil ions by the transmissions of the electron suppression grid and carbon foil, each with a transmission rate of 73%, as discussed later. We add all 100 GFs calculated according to Equation 2.1 at each combination of energy and elevation ranges to compute the overall GF for a given SIMION run according to Equation 2.2.

2.4 Calibration of the Solar Orbiter Heavy Ion Sensor with its Ion Optical Model

2.4.1 Methodology

To achieve the ultimate goal of mapping the GF to the flight model, we first represent the SO-HIS instrument with an IOM and associated Monte Carlo particle simulation. We then determine the desired characteristics, such as the GF, from the model and simulations. We verify the model against laboratory measurements, as it is critical to ensure that the model accurately represents the flight model. The SIMION IOM and labeled schematic are shown in Figure 2.3.

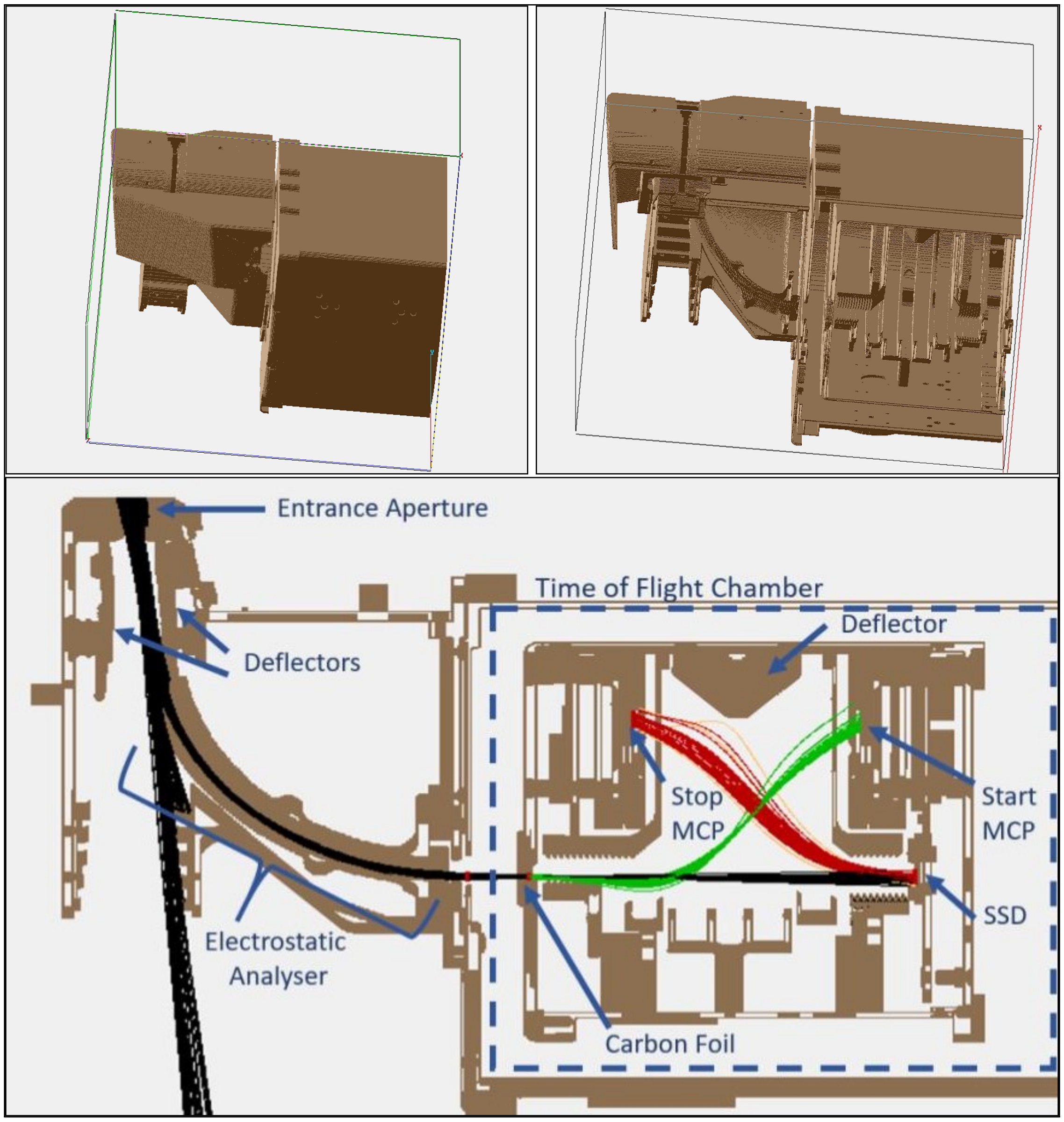


Figure 2.3: Top: SO-HIS model in 3D-isometric view in SIMION (Left) with exploded view (Right). Heat shield aperture can be seen in the top left of each image and TOF chamber can be seen on the right. Bottom: Labeled view of a 2D slice of the SO-HIS model in SIMION in a similar orientation to the 3D-isometric view above with particle trajectories displayed. In the online full color version, initial particle trajectories are black, start electron tracks are green, stop electron tracks are red, and tracks of electrons that miss a detector are yellow.

We examine factors that may affect the IOM instrument response and cause deviations from the flight model response, including model resolution and manufacturing and integration effects on the hardware, among others. To characterize the effect of these differences, we find a mapping between the instrument responses. Once the IOM and simulations are determined to accurately represent the flight model, simulations are performed for numerous configurations, and a function is fitted to the instrument response. We may then determine the GF of the flight model under arbitrary configurations and make determinations of the flight model characterization from the ground. We perform this process for values which represent commonly occurring states in the solar wind.

Itemized steps of our procedure are given as follows:

1. Create a SIMION IOM and Monte Carlo simulation using spaceflight CAD drawings.
   (a) Create a tool to assess and verify SIMION scans.
   (b) Identify factors that could cause the IOM to deviate from the flight model and ensure these factors are addressed such that the IOM accurately represents the instrument.
   (c) Determine that the IOM maps to laboratory measurements in lab configuration.
2. Write analysis code to calculate the simulated GF.
3. Determine a function for the GF of the flight model for standard solar wind conditions.

2.4.2 Ion Optical Model and Monte Carlo Simulation

We are able to analyze the instrument response by creating an IOM using the SIMION environment from the spaceflight CAD drawings, in which a potential array (PA) file is created for all surfaces that will be held at a given voltage, including those with adjustable voltages, such as the deflectors, top cap, and ESA plates. We orient our coordinate axes such that the origin is aligned with the ESA's axis of symmetry, the x-axis is pointing out of the instrument aperture, the z-axis is pointing down the TOF chamber, and the y-axis completes the right-handed coordinate system. We remove all insulators from the model, as they will not affect the electric fields in the instrument. We create zero-thickness PAs for the thin CF and electron suppression grid at the entrance of the TOF chamber. These zero-thickness PAs allow particles to pass through within the SIMION simulation environment while still affecting the electric fields. We implement their thickness and effects on scattering and transmission within the Monte Carlo simulation and post-processing analysis code based on effects simulated using the TRIM software. For scattering effects in the Monte Carlo simulation, we take the thickness of the CF to be $1.1\mu g/cm^2$, which is the upper bound of the value $1.0\mu g/cm^2 \pm 10\%$ measured at Southwest Research Institute in San Antonio, TX. We take the upper bound of this value for simulation, as a mono-layer of water is generally expected to be present on the surface of the foil. Effects of a foil of this thickness as modeled in the TRIM software are then simulated in the Monte Carlo code. The 73% transmission rate of each of the CF and suppression grid are accounted for in the post-processing analysis code during the GF calculation.

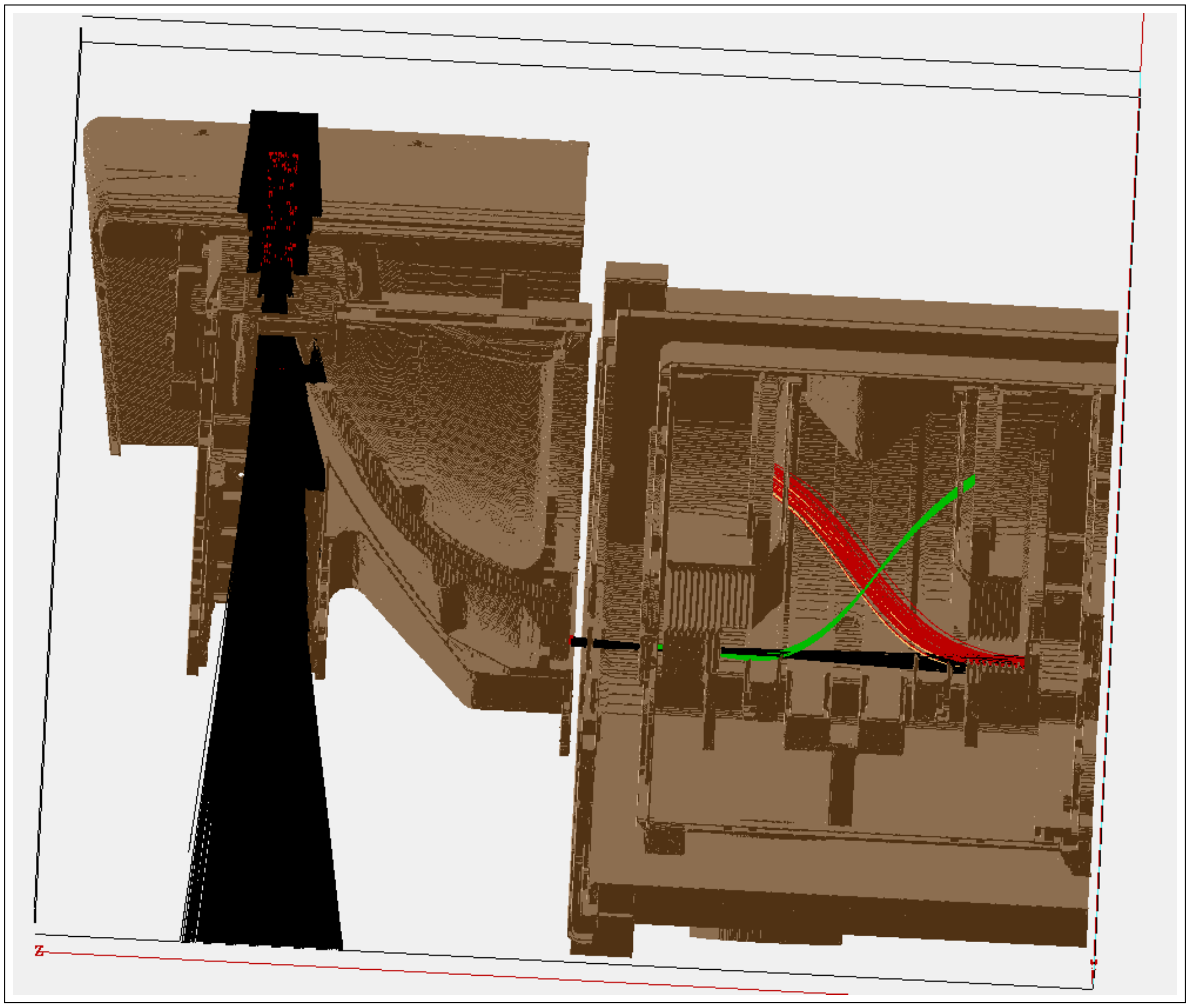


Figure 2.4: A SIMION run with the SO-HIS instrument viewed in exploded 3D-isometric view. Particles start at a flat SS outside the heat shield aperture. Initial particle trajectories are black, start electron tracks are green, stop electron tracks are red, and tracks of electrons that miss a detector are yellow.

Runs in the SIMION environment constitute Monte Carlo simulations in which a particular E/q step and elevation acceptance are selected. We adjust the voltages according to the values used in laboratory testing, as determined by testing performed at SWRI and IRAP. Particles are generated on a flat source surface (SS) in front of the instrument aperture, similar to that described by Collinson et al. (2012). The desired number of particles are randomly but evenly generated at this surface, which is sized to fully illuminate the aperture given the desired particle angles in azimuth and elevation. The initial particle velocities from this SS are determined by the E/q step and the input elevation and azimuth ranges. We place the SS at a distance of 50mm from the axis of symmetry along the x-axis. We find no significant effects of fringe fields outside the instrument aperture, and at a location of 50mm, we are able to simulate all accepted elevation and azimuth angles within the simulation environment while completely covering the accepting regions of the instrument aperture without losing excessive particles to splats that occur when the particles are unable to enter the instrument aperture, where, in a SIMION run, a particle is considered to have ”splatted” if it impacts any surface or a simulation boundary. A sample SIMION run is shown in Figure 2.4.

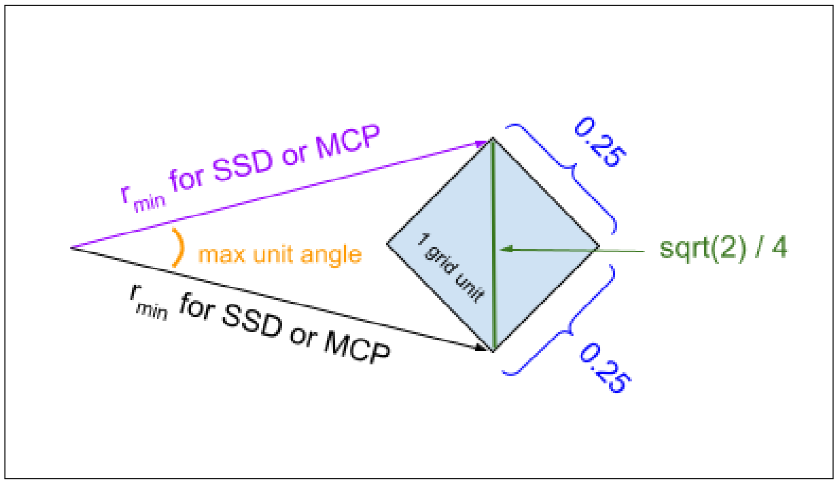


Figure 2.5: Diagram for estimation of error in polar coordinates for 1 SIMION grid unit (0.25 mm/gu).

A series of up to eight marks are written to an output file for each particle sequence when: 1. A particle is generated at the SS, 2. A particle crosses the instrument aperture, 3. A particle exits the ESA, 4. A particle reaches the CF, 5. A particle passes the CF, 6. A particle strikes the SSD, 7. A particle strikes the Start MCP, and 8. A particle strikes the Stop MCP or at any final splat of a particle sequence, which may be on a detector as listed above, on an instrument wall, or at a simulation boundary. Not all particle sequences will generate all eight marks, as most will splat on an instrument wall or simulation boundary before completing the full sequence, and some may encounter some detectors while missing others. A full triple-coincidence sequence will include eight marks made at generation, instrument aperture, ESA exit, pre-CF, post-CF, SSD, Start MCP, and Stop MCP. Marks written to the output file include the particle’s ion number, time of flight, mass, charge, position and velocity components in Cartesian coordinates, elevation and azimuth angles, and energy. To determine whether a mark should be generated at a particular time step, the particle position is compared to the known locations of the desired marks either in Cartesian or cylindrical coordinates as appropriate, with a tolerance of one grid unit. In Cartesian coordinates, one grid unit is taken as the simulation resolution of 0.25 mm/gu, while one grid unit in radial or angular coordinates is calculated as follows the maximized value for an isosceles triangle over one grid unit, as shown in Figure 2.5. The radial tolerance is therefore taken as the maximum distance across one grid unit, or $r_{tol} = \sqrt{2}/4$, while the angular tolerance is taken as $\theta_{tol} = 2\,\mathrm{arcsin}\,(\sqrt{2}/8)/r_{min}$, where r_{min} is the minimum radial boundary of a given region, such as an MCP or SSD, in order to calculate the maximum tolerance.

2.4.3 Cross-Calibration

2.4.3.1 Identifying Factors That Could Lead to Discrepancies Between Laboratory and IOM Values

Each of the IOM and the flight model are physically different systems and necessarily have differences in makeup and response. This is true of any backup or spare system, including flight spares and engineering models. We deal with these differences by understanding and minimizing their effects and finding a function to map between the measured IOM response and the measured laboratory values. For our standard solar wind case, we find the mapping from the SIMON model values to the measured laboratory values to be well represented by a factor of $\sim$4, as discussed later in our results. Factors that we account for when comparing between the SO-HIS flight model and its IOM include the following:

Model resolution. The IOM has resolution limitations due to modeling software capabilities in SIMION and due to simulation complexity and computational resources. A relatively efficient IOM is required to make reasonable simulations with reasonable timing. Such differences resulting from limitations in model resolution include the inability to accurately represent the fine scalloping on the ESA plates. These effects should be accounted for in the GF mapping.

Analyzer constant. As the IOM is limited by simulation resolution, the analyzer constant, k, is determined by simulation resolution rather than laboratory testing. These effects should be accounted for in the GF mapping.

Charge buildup on surfaces. In the flight instrument, charged particles may accumulate on surfaces and affect the electric potentials, which affect incoming particles. This effect cannot be simulated in SIMION, but should be accounted for in the GF mapping.

Ion source. In the lab, the instrument was mounted on a plate and tilted in elevation and rotated in azimuth under a fixed ion beam that fully covered the entrance aperture. Due to simulation environment constraints in SIMION, the IOM must remain stationary. Additionally, the ion SS cannot be placed tangent to an arc around the aperture, but must rather be represented by a flat plane perpendicular to the aperture. Also, due to simulation computational constraints, the SS should be placed relatively close to the aperture. To achieve the same effects of the chamber ion beam, we ensure that the beam fully illuminates the aperture, is not appreciably affected by aperture fringe fields, and has ions originating at the desired angles such that they are uniformly distributed, as would be the case given an instrument being tilted and rotated within an ion beam.

Insulators. All surfaces in SIMION are treated as potential arrays; therefore, all insulators that exist in the flight model must be removed in the IOM. This difference should be trivial, as ions that should have splatted on these surfaces had they been included, will splat elsewhere on an instrument wall or simulation boundary. Only ions that make it through the ESA and the CF into the TOF chamber and onto one of the detectors will affect the measured instrument response.

Chamber effects. The laboratory vacuum chamber is a physically different setup from the simulated environment in SIMION and may have factors not taken into account, such as water molecules in the chamber. These effects should be negligible or accounted for in the GF mapping.

Electron suppression grid and carbon foil transmissions. The grid and CF at the entrance to the TOF chamber reduce the number of ions that successfully enter the TOF section. We are able to account for this by using a laboratory measured transmission of 73% for each.

Fringe behaviors at higher azimuths (above ~45°). There may be issues related to pileup at higher azimuths due to a necessity to move the instrument wall very close to the start MCP that may cause a difference in response between the SIMION IOM and the flight model. For our purposes in the standard solar wind case, these azimuths are not relevant, so we neglect them for this study.

SSD active area. The SSD comprises specific dead zones between cells and near spokes, which are not explicitly accounted for in the Monte Carlo simulation. We determine that these effects are minimal in the regions of the standard solar wind used for this study; however, this effect could be modeled in future studies.

2.4.3.2 Visualization Tool for Scan Verification

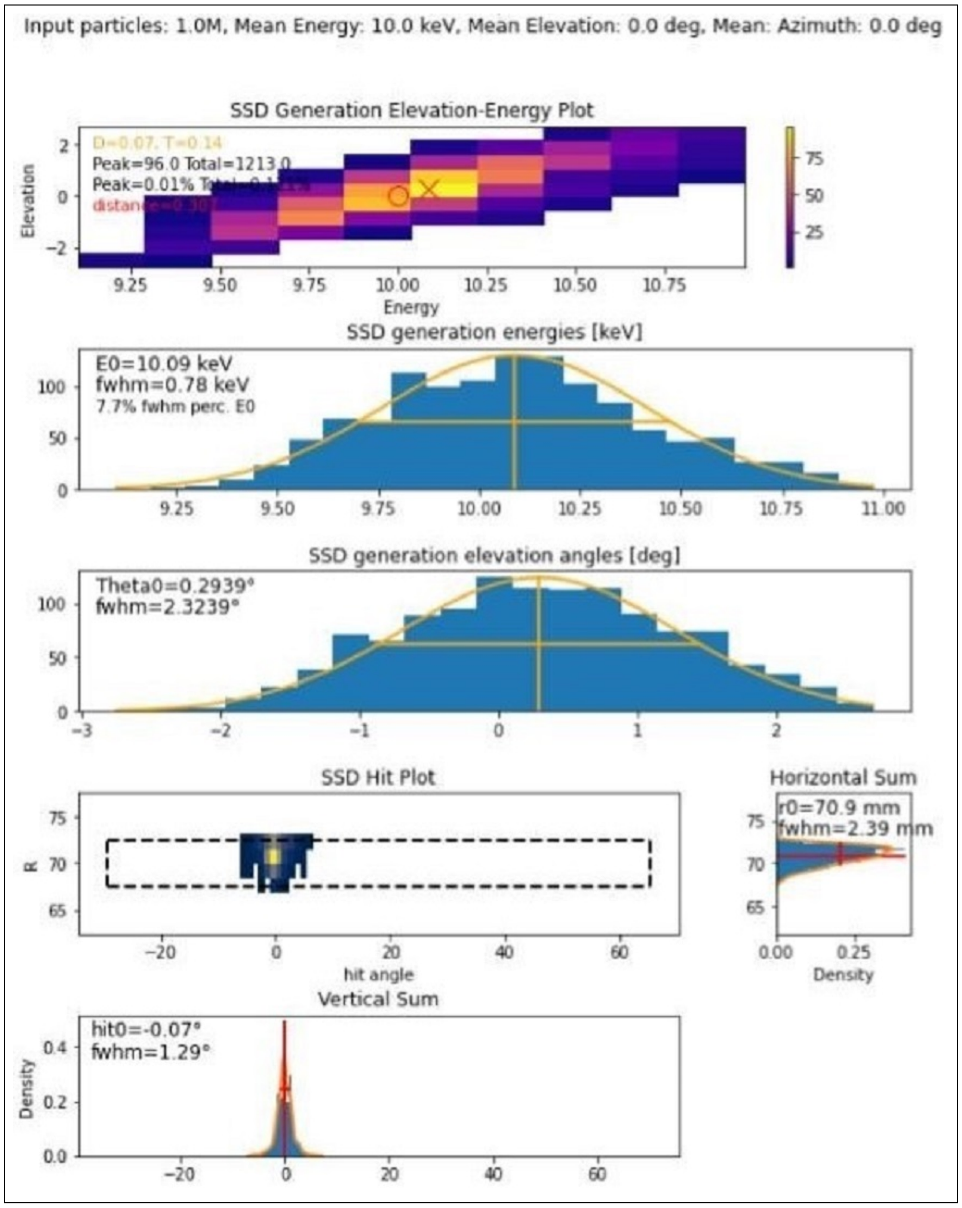


Figure 2.6: A panel plot used in this characterization procedure to verify SIMION runs (10 keV, 0° elevation in spaceflight/IOM coordinates). The first panel contains the 2D histogram EAR, the second and third panels contain 1D histograms in energy and elevation only, and the fourth panel contains the SSD spot with horizontally (Right) and vertically (Bottom) summed counts. Orientations of EAR panels differ due to differences in positive and negative angle definitions between the laboratory and flight model / IOM coordinate systems. These differences are accounted for in all measurement comparisons.

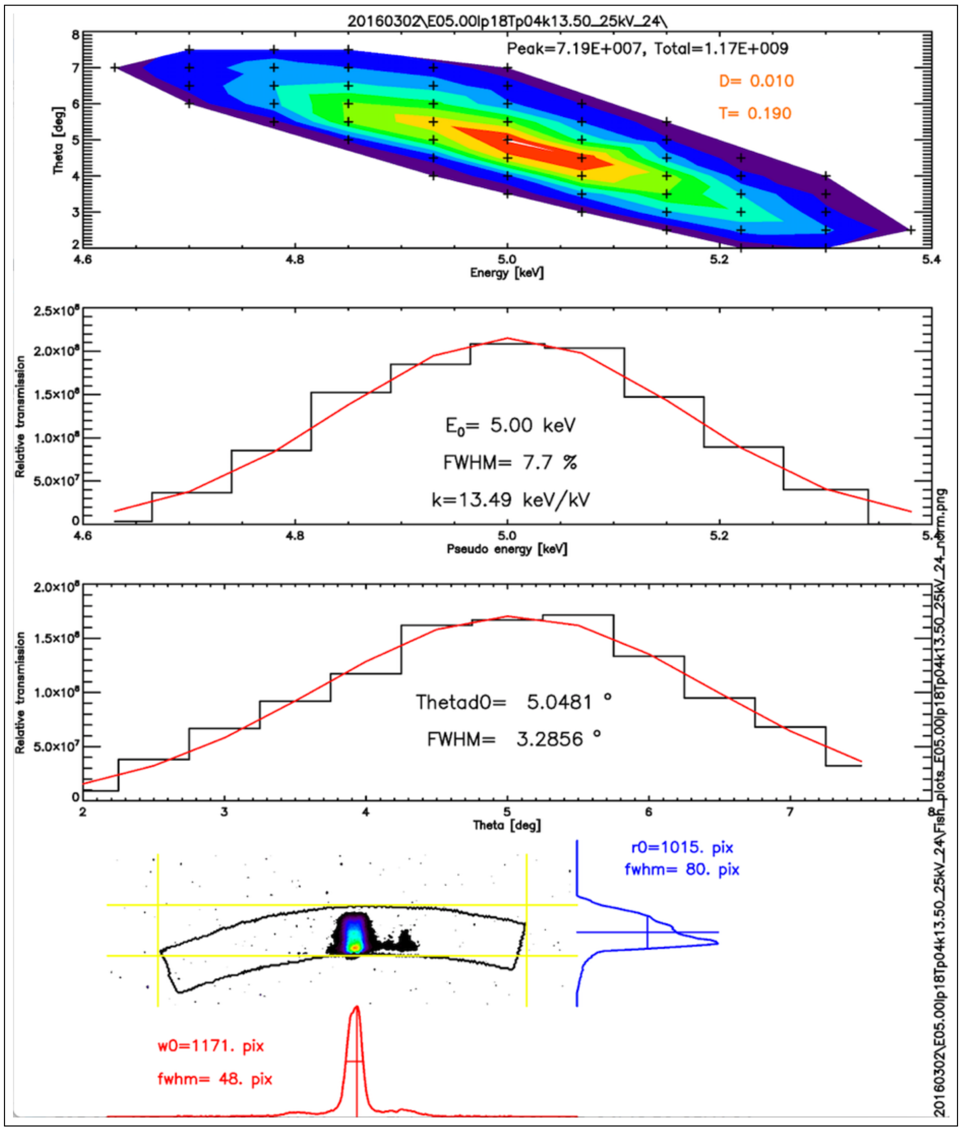


Figure 2.7: A panel plot used in the lab at SWRI during the SO-HIS voltage optimization campaign (5 keV, -5° elevation in spaceflight/IOM coordinates).

SIMION runs are used to compare the instrument response of the IOM with that of the flight model. Runs are performed with 2.5M particles for H^+ in a uniformly distributed beam of E/q range $\pm 15\%$ and azimuth range $\pm 0.1°$. We perform each run as a scan over a range in elevation of $\pm 7°$ to ensure full Energy – Angle Response (EAR) coverage. To verify the quality of a scan, we create a panel plot, as shown in Figure 2.6 similar to that used in the laboratory voltage optimization, as shown in Figure 2.7. This plot allows us to verify that our IOM scans meet the qualitative standards of a laboratory scan and in fact capture the full EAR. The first panel of this plot visualizes the EAR as a 2D histogram of triple-coincidence counts on initial elevation against initial energy. We identify the location of the histogram peak and the percentage of SSD strike counts to fall within the peak, allowing us to gauge particle transmission. On the next two panels, we visualize the histogram in energy and elevation only, measuring the full-width-half-max, which allows us to gauge E/q and elevation acceptance. On the last grouping of panels, we analyze the “spot” created by strikes on the SSD by creating a 2D histogram of counts in a region representing the physical surface of the SSD. Since the SSD is curved, the vertical and horizontal axes are radial distance and hit angle along the SSD in polar coordinates. Qualitatively, this spot should be well localized and contained on the SSD for optimal detection. The Horizontal Sum and Vertical Sum histogram plots visualize the location and spread of the spot along each axis. For the laboratory voltage optimization procedure, the quality of a scan is assessed by verifying that where possible, transmission is relatively maximized, a desired range of E/q acceptance is achieved, the elevation acceptance is within a desired threshold, and the SSD spot is relatively localized and contained.

2.5 Results

Over SIMION scans comprising 24 configurations of energy, elevation, and azimuth, we calculate the GF of the IOM to compare with laboratory measurements. We interpolate laboratory values for azimuths of 5° and 10° to get a representation of flight model values near solar wind azimuths in the instrument’s field of view (FOV) and to ensure that we are checking for dependencies in E/q, elevation, and azimuth in the IOM to lab value comparison. We then evaluate the ratios of the SIMION values to the laboratory values to determine if we find a significant dependence on E/q, elevation, or azimuth. We compare the mean and median and explore the option of using 2D interpolation. We find that the ratio is very consistent with a value of 4 (SIMION / laboratory). We use this value rather than a function or interpolation scheme in order to not introduce artifacts from under-constrained data. Table 2.1 in Appendix 2.8 summarizes our results for the GF and the SIMION / laboratory ratio. The mean and median of all ratios both result in a value of $\sim$4.

Helium and oxygen are important heavy ion components of the solar wind. We therefore complete additional SIMION runs to measure IOM GFs for standard solar wind speeds of 438 km/s and

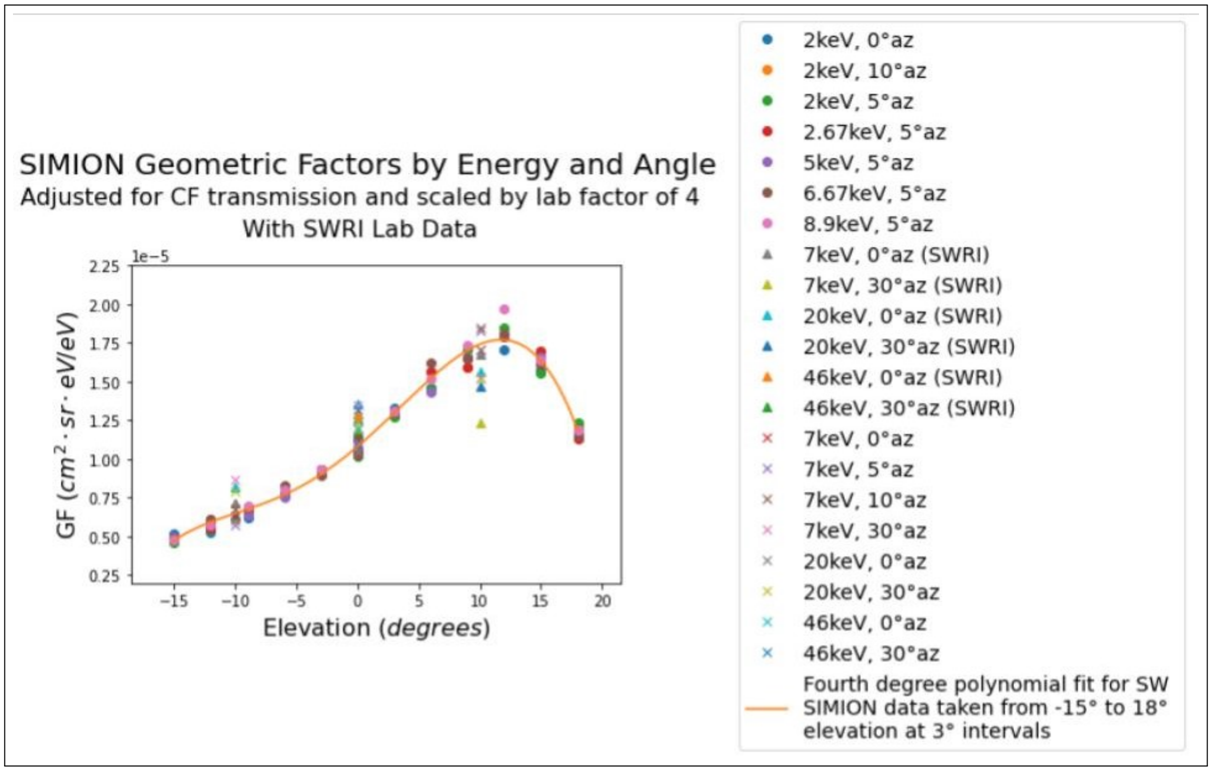


Figure 2.8: IOM and laboratory GF measurements with a fourth degree polynomial fit to the SIMION values measured at 3° elevation intervals from -15° to 18° for standard solar wind conditions.

800 km/s corresponding to energies for He^{2+} and O^{6+} (2 keV, 6.67 keV, 2.67 keV, and 8.90 keV) as well as for 5 keV at the average measured solar wind azimuth of 5° for the SO-HIS instrument taken at the instrument bin resolution of 3° elevation from -15° to 18°. We compute the energies for He^{2+} and O^{6+}, derived from Gloeckler et al. (1998), by:

$$E/q \approx (v_{sw}/438)^2 \times m/q \tag{2.3}$$

where E/q is energy-per-charge, v_{sw} is the solar wind speed, and m/q is the mass-per-charge. We take a standard slow solar wind speed of 438 km/s to set protons at an energy of 1 keV.

SO-HIS is designed for a full elevation range of -20° to 20° given a ±17° range with ∼3° bin resolution. However, under standard solar wind conditions, we do not expect measurements in the very low elevation range near -20°. Since we expect potential edge effects in that low elevation range and since it is not needed for our characterization under standard solar wind conditions, we do not include -18° in our SIMION runs. We compare values at 2 keV for azimuths of 0°, 5°, and

10° and determine that we do not see an azimuthal dependence within a reasonable range of solar wind azimuths.

In Figure 2.8 we plot together all SIMION GF values adjusted by the scaling factor of 4 for the measurements taken at the 3° elevation resolution as well as those values that correspond to lab measurements along with all laboratory GF values for azimuths less than 45°. We perform a fit on the SIMION values measured at the 3° elevation resolution resulting in Equation 2.4. We use a fourth-degree polynomial to capture all visible features of the data. The measured GF values used in the fourth-degree polynomial fit are reported in Table 2.2 through Table 2.6 in Appendix 2.8.

$$\begin{aligned} G_{fit} = \quad & (-1.010 \times 10^{-10})\theta_{el}^4 + (-1.275 \times 10^{-9})\theta_{el}^3 \\ & +(2.154 \times 10^{-8})\theta_{el}^2 + (6.732 \times 10^{-7})\theta_{el} \\ & +(1.079 \times 10^{-5}) \end{aligned} \tag{2.4}$$

where θ_{el} is elevation angle in degrees, the fitting errors of the coefficients from highest to lowest power are: A: 7.467×10^{-12}, B: 7.801×10^{-11}, C: 2.013×10^{-9}, D: 1.427×10^{-8}, E: 1.007×10^{-7}, the fit R^2 value is 0.988, and the fit χ^2 value is 1.483×10^{-6} with a p value of 1.

2.6 Conclusions

We present a cross-calibration of SO-HIS with its IOM. SO-HIS is an *in situ* triple-coincidence ion mass spectrometer intended to measure elemental composition and 3D velocity distribution functions of ions in the range He – Fe in the energy-per-charge (E/q) range 0.5 – 80 keV/e with charge-states from He^{+} to Fe^{20+}. SO-HIS measures the bulk solar wind in the range 0.5 – 18 keV/e and the suprathermal solar wind components of He, C, O, and Fe up to at least 60 keV/e, as well as PUIs. SO-HIS uses 64 E/q steps with an energy resolution of 6 – 10% and 16 elevation steps in the range $\pm 17°$ with an angular resolution of $<3.5°$ and has a continuous azimuthal acceptance in the range -30° to +66°.

The SO-HIS IOM is created and validated along with its associated Monte Carlo particle simulation using the SIMION environment and calibrated to the flight model. We discuss the parameters for our SIMION runs and our verification procedures as well as compare between our results and the laboratory measurements. We present factors that may relate to potential differences in our IOM results from those measured in the lab for the flight model, leading to the necessity to determine a mapping between the two, as well as our steps taken to ensure reasonable comparison between the laboratory data and the SIMION outputs. We discuss our methodology for calculating the GF and determining a mapping of a factor of 4 between laboratory and SIMION values. We enumerate our specific run parameters and our measured values both for the determination of the IOM to laboratory measurements scaling factor as well as for the fitting of the GF function. We finally present a function in Equation 2.4 for determining the instrument's GF under standard solar wind conditions. We report here on the first cross-calibration for SO-HIS between laboratory measurements of the flight model and the IOM.

Subsequent studies are planned to extend this calibration in the future to include the entire SO-HIS acceptance beyond the standard solar wind conditions. Additionally, we plan to explore discrepancies of azimuthal response at larger azimuthal angles and more accurately represent the SSD active area.

2.7 Acknowledgments

We thank Mark Ferris, David Heirtzler, Keiichi Ogasawara, Toni Galvin, and the SO-HIS instrument team for providing instrument drawings and specifications.

This work is supported by NASA SO-HIS grant: (NASA) NNG10EK25C (SWRI) A99201MO. This work has additionally been supported by the Michigan Plasma Sciences & Engineering Graduate Fellowship, The Michigan Space Grant Consortium Graduate Fellowship, the American Geophysical Union Student Travel Grant, the University of Michigan Rackham Student Travel Grant, the University of Michigan Diversity, Equity, and Inclusion Grant in the Department of Climate and Space Sciences & Engineering, and the University of Michigan College of Engineering Office of Student Affairs Student Travel Grant.

2.8 Appendix: Reported Values

Table 2.1: Geometric Factors (GFs) reported in $cm^2 \cdot sr \cdot eV/eV$ measured using the IOM compared with laboratory measurements to determine a mapping or ratio from IOM to spaceflight values. IOM GFs are calculated using a 73% transmission rate from each of the electron suppression grid and carbon foil. The ratio is taken as IOM GF / SWRI GF. Azimuths marked with an asterisk use laboratory values that have been interpolated from measured values for a representative azimuth closer to the measured standard solar wind in the instrument FOV.

E/q	Elevation	Azimuth	SWRI GF	IOM GF	Ratio
20 keV	-10°	0°	8.15×10^{-6}	2.41×10^{-5}	2.96
20 keV	0°	0°	1.36×10^{-5}	4.17×10^{-5}	3.08
20 keV	10°	0°	1.56×10^{-5}	6.82×10^{-5}	4.37
20 keV	-10°	5° *	7.86×10^{-6}	2.71×10^{-5}	3.45
20 keV	0°	5° *	1.31×10^{-5}	3.93×10^{-5}	3.00
20 keV	10°	5° *	1.55×10^{-5}	6.86×10^{-5}	4.43
20 keV	-10°	10° *	7.57×10^{-6}	2.69×10^{-5}	3.56
20 keV	0°	10° *	1.27×10^{-5}	4.28×10^{-5}	3.37
20 keV	10°	10° *	1.53×10^{-5}	7.44×10^{-5}	4.85
20 keV	-10°	30°	6.40×10^{-6}	3.16×10^{-5}	4.94
20 keV	0°	30°	1.10×10^{-5}	4.96×10^{-5}	4.51
20 keV	10°	30°	1.47×10^{-5}	6.07×10^{-5}	4.13
7 keV	-10°	0°	7.17×10^{-6}	6.29×10^{-5}	3.51
7 keV	0°	0°	1.26×10^{-5}	1.13×10^{-5}	3.60
7 keV	10°	0°	7.17×10^{-6}	1.70×10^{-5}	4.07
7 keV	-10°	5° *	7.00×10^{-6}	2.29×10^{-5}	3.28
7 keV	0°	5° *	1.25×10^{-5}	4.40×10^{-5}	3.51
7 keV	10°	5° *	8.03×10^{-6}	7.32×10^{-5}	4.57
7 keV	-10°	10° *	6.83×10^{-6}	2.71×10^{-5}	3.97
7 keV	0°	10° *	1.24×10^{-5}	4.98×10^{-5}	4.02
7 keV	10°	10° *	8.90×10^{-6}	7.37×10^{-5}	4.82
7 keV	-10°	30°	6.16×10^{-6}	3.45×10^{-5}	5.59
7 keV	0°	30°	1.20×10^{-5}	5.39×10^{-5}	4.50
7 keV	10°	30°	1.24×10^{-5}	6.09×10^{-5}	4.92

Table 2.2: Geometric Factors (GFs) reported in $cm^2 \cdot sr \cdot eV/eV$ measured using the IOM at 3° elevation intervals from -15° to 18°. Reported values are used for creating the fourth-degree polynomial fit. IOM GFs are calculated using a 73% transmission rate from each of the electron suppression grid and carbon foil and then scaled down by a factor of 4. Values are reported in this table for an E/q of 2 keV.

E/q	Elevation	Azimuth	Scaled IOM GF
2 keV	-15°	0°	5.2125×10^{-6}
2 keV	-12°	0°	5.2975×10^{-6}
2 keV	-9°	0°	6.22×10^{-6}
2 keV	-6°	0°	7.915×10^{-6}
2 keV	-3°	0°	9.2975×10^{-6}
2 keV	0°	0°	1.1385×10^{-5}
2 keV	3°	0°	1.32975×10^{-5}
2 keV	6°	0°	1.45925×10^{-5}
2 keV	9°	0°	1.65225×10^{-5}
2 keV	12°	0°	1.70725×10^{-5}
2 keV	15°	0°	1.57075×10^{-5}
2 keV	18°	0°	1.18775×10^{-5}
2 keV	-15°	10°	4.865×10^{-6}
2 keV	-12°	10°	5.5575×10^{-6}
2 keV	-9°	10°	6.5825×10^{-6}
2 keV	-6°	10°	7.875×10^{-6}
2 keV	-3°	10°	9.2775×10^{-6}
2 keV	0°	10°	1.0545×10^{-5}
2 keV	3°	10°	1.308×10^{-5}
2 keV	6°	10°	1.56275×10^{-5}
2 keV	9°	10°	1.58925×10^{-5}
2 keV	12°	10°	1.792×10^{-5}
2 keV	15°	10°	1.692×10^{-5}
2 keV	18°	10°	1.214×10^{-5}
2 keV	-15°	5°	4.5725×10^{-6}
2 keV	-12°	5°	5.4475×10^{-6}
2 keV	-9°	5°	6.83×10^{-6}
2 keV	-6°	5°	7.735×10^{-6}
2 keV	-3°	5°	9.015×10^{-6}
2 keV	0°	5°	1.02075×10^{-5}
2 keV	3°	5°	1.27075×10^{-5}
2 keV	6°	5°	1.44675×10^{-5}
2 keV	9°	5°	1.7025×10^{-5}
2 keV	12°	5°	1.848×10^{-5}
2 keV	15°	5°	1.553×10^{-5}
2 keV	18°	5°	1.23×10^{-5}

Table 2.3: Geometric Factors (GFs) reported in $cm^2 \cdot sr \cdot eV/eV$ measured using the IOM at 3° elevation intervals from -15° to 18°. Reported values are used for creating the fourth-degree polynomial fit. IOM GFs are calculated using a 73% transmission rate from each of the electron suppression grid and carbon foil and then scaled down by a factor of 4. Values are reported in this table for an E/q of 2.67 keV.

E/q	Elevation	Azimuth	Scaled IOM GF
2.67 keV	-15°	5°	4.865×10^{-6}
2.67 keV	-12°	5°	5.5575×10^{-6}
2.67 keV	-9°	5°	6.5825×10^{-6}
2.67 keV	-6°	5°	7.875×10^{-6}
2.67 keV	-3°	5°	9.2775×10^{-6}
2.67 keV	0°	5°	1.0545×10^{-5}
2.67 keV	3°	5°	1.308×10^{-5}
2.67 keV	6°	5°	1.56275×10^{-5}
2.67 keV	9°	5°	1.58925×10^{-5}
2.67 keV	12°	5°	1.792×10^{-5}
2.67 keV	15°	5°	1.692×10^{-5}
2.67 keV	18°	5°	1.1255×10^{-5}

Table 2.4: Geometric Factors (GFs) reported in $cm^2 \cdot sr \cdot eV/eV$ measured using the IOM at 3° elevation intervals from -15° to 18°. Reported values are used for creating the fourth-degree polynomial fit. IOM GFs are calculated using a 73% transmission rate from each of the electron suppression grid and carbon foil and then scaled down by a factor of 4. Values are reported in this table for an E/q of 5 keV.

E/q	Elevation	Azimuth	Scaled IOM GF
5 keV	-15°	5°	4.95×10^{-6}
5 keV	-12°	5°	5.92×10^{-6}
5 keV	-9°	5°	6.3925×10^{-6}
5 keV	-6°	5°	7.515×10^{-6}
5 keV	-3°	5°	9.245×10^{-6}
5 keV	0°	5°	1.0395×10^{-5}
5 keV	3°	5°	1.303×10^{-5}
5 keV	6°	5°	1.42675×10^{-5}
5 keV	9°	5°	1.659×10^{-5}
5 keV	12°	5°	1.8×10^{-5}
5 keV	15°	5°	1.65625×10^{-5}
5 keV	18°	5°	1.15675×10^{-5}

Table 2.5: Geometric Factors (GFs) reported in $cm^2 \cdot sr \cdot eV/eV$ measured using the IOM at 3° elevation intervals from -15° to 18°. Reported values are used for creating the fourth-degree polynomial fit. IOM GFs are calculated using a 73% transmission rate from each of the electron suppression grid and carbon foil and then scaled down by a factor of 4. Values are reported in this table for an E/q of 6.67 keV.

E/q	Elevation	Azimuth	Scaled IOM GF
6.67 keV	-15°	5°	4.855×10^{-6}
6.67 keV	-12°	5°	6.1125×10^{-6}
6.67 keV	-9°	5°	6.85×10^{-6}
6.67 keV	-6°	5°	8.245×10^{-6}
6.67 keV	-3°	5°	8.9425×10^{-6}
6.67 keV	0°	5°	1.02475×10^{-5}
6.67 keV	3°	5°	1.30125×10^{-5}
6.67 keV	6°	5°	1.6215×10^{-5}
6.67 keV	9°	5°	1.65025×10^{-5}
6.67 keV	12°	5°	1.8055×10^{-5}
6.67 keV	15°	5°	1.60775×10^{-5}
6.67 keV	18°	5°	1.147×10^{-5}

Table 2.6: Geometric Factors (GFs) reported in $cm^2 \cdot sr \cdot eV/eV$ measured using the IOM at 3° elevation intervals from -15° to 18°. Reported values are used for creating the fourth-degree polynomial fit. IOM GFs are calculated using a 73% transmission rate from each of the electron suppression grid and carbon foil and then scaled down by a factor of 4. Values are reported in this table for an E/q of 8.9 keV.

E/q	Elevation	Azimuth	Scaled IOM GF
8.9 keV	-15°	5°	4.765×10^{-6}
8.9 keV	-12°	5°	5.7425×10^{-6}
8.9 keV	-9°	5°	6.9975×10^{-6}
8.9 keV	-6°	5°	8.0125×10^{-6}
8.9 keV	-3°	5°	9.335×10^{-6}
8.9 keV	0°	5°	1.11175×10^{-5}
8.9 keV	3°	5°	1.307×10^{-5}
8.9 keV	6°	5°	1.5155×10^{-5}
8.9 keV	9°	5°	1.73575×10^{-5}
8.9 keV	12°	5°	1.96825×10^{-5}
8.9 keV	15°	5°	1.63975×10^{-5}
8.9 keV	18°	5°	1.18225×10^{-5}

CHAPTER 3

Determining the Interstellar Wind Longitudinal Inflow Evolution Using Pickup Ions in the Helium Focusing Cone

This chapter is being prepared for submission to *The Astrophysical Journal* and was written in collaboration with Susan T. Lepri, Jim M. Raines, Jason A. Gilbert, Eberhard Möbius, Jonathan Bower, and Ryan M. Dewey.

3.1 Abstract

We determine the flow direction, λ_{ISN}, and trend of the interstellar wind through the Heliosphere over an 11-year solar cycle using ACE/SWICS He^+ double-coincidence pickup ion (PUI) measurements in the helium focusing cone from the data set spanning 13 full orbits with focusing cone crossings occurring between the years 1998 and 2010. We determine a flow direction for each 3-orbit boxcar centered on the years 1999 through 2009 and find the trend of the resulting best-fit line. We account for solar transients by removing measurements associated with CMEs. To ensure the PUI torus is in view, we restrict measurements to times when the solar wind velocity and IMF vectors are relatively perpendicular. To account for varying IMF angles, we transform the distribution into the solar wind frame. To account for general solar cycle activity, we use a boxcar averaging technique. Lastly, to ensure that we are analyzing newly injected interstellar PUIs largely unaffected by transport effects, we find the distribution value at the cutoff location and limit the search region to a range of $0.7 \leq w' \leq 1.5$, where $w' = v/v_{sw} - 1$, and v_{sw} is the solar wind velocity. We determine the trend in the flow direction over the 11 full 3-orbit boxcars to be 0.00° $\pm 0.51°$ / yr, indicating that we do not measure a varying flow direction over this time period. We then repeat the focusing cone analysis for the combined data set and determine an overall flow direction of 75.37° $\pm 0.43°$.

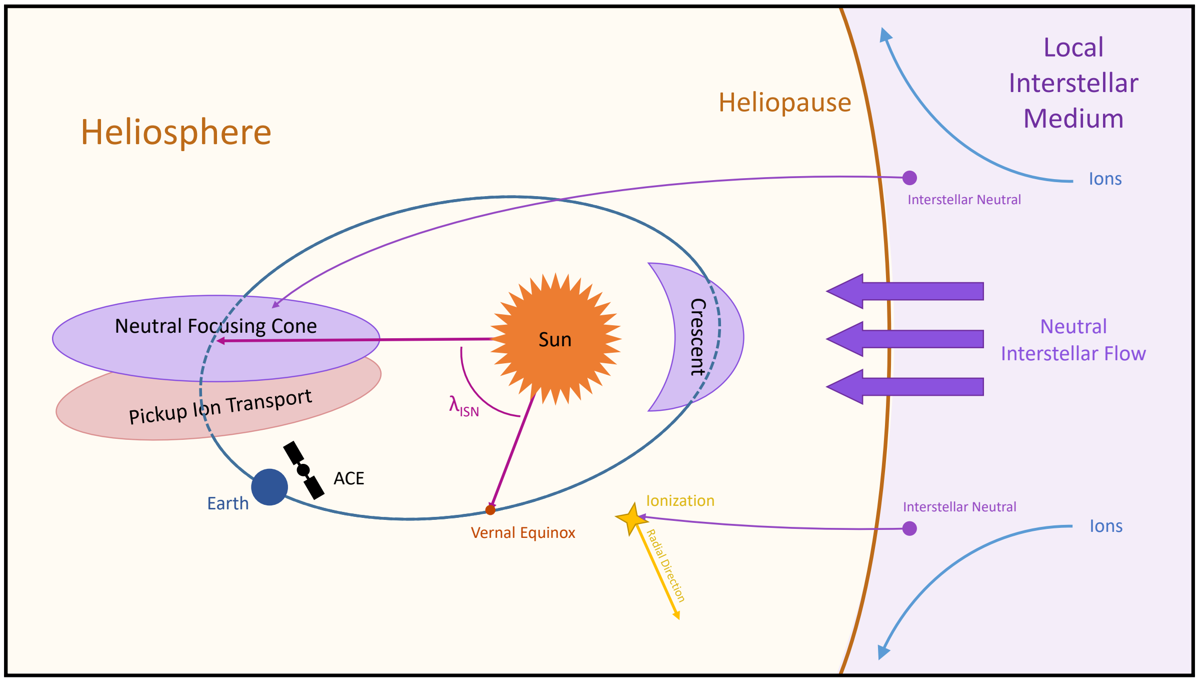


Figure 3.1: Diagram of Heliosphere – Interstellar interactions.

3.2 Introduction

The Heliosphere is the Sun's region of influence, carved out of the Local Interstellar Medium (LISM) by the solar wind and IMF. The plasma within the Heliosphere interacts with the Earth and the rest of the Solar System, and while the bulk of the plasma originates from the Sun, additional sources include planets, comets, dust, and the LISM. A steady stream of interstellar source plasma in the Heliosphere originates from the inflow of interstellar neutrals due to the relative motion of the Sun and the LISM. The neutrals flow freely across the Heliopause, or the boundary between the Heliosphere and the LISM, unaffected by the Solar magnetic fields, as shown in Figure 3.1. These neutrals may then become ionized through photoionization, electron impact ionization, and charge exchange with the solar wind. The newly ionized particles are then affected by the Solar magnetic fields and become accelerated, or picked up, by the solar wind. These ionized particles, therefore, are known as pickup ions (PUIs) and can reach speeds up to two times that of the solar wind, forming a critical portion of the energetic particle population in the Heliospheric plasma distribution (Möbius et al. (1985)). Due to the pickup process and their gyration about the Solar magnetic field lines, the PUIs form signatures distinct from the solar wind peak within the distribution in velocity space, allowing them to be analyzed within the distribution function.

Measurements of the interstellar flow parameters were first made using ultraviolet back scatter techniques by Bertaux & Blamont (1971); Thomas & Krassa (1971); Weller & Meier (1974), and the first *in situ* measurements of PUIs were made by Möbius et al. (1985). Properties of the interstellar wind, and therefore the interaction between the Heliosphere and the LISM, such as the longitudinal inflow direction, λ_{ISN}, can be measured using *in situ* particle data and measurements of the solar wind distribution at 1 AU (Möbius et al. (1985); Moebius et al. (1995)). The flow direction, λ_{ISN}, directly correlates to the orientation of the helium gravitational focusing cone and crescent, as shown in Figure 3.1. The focusing cone is a structure formed by the gravitational focusing of interstellar neutral helium on the downwind side of the Sun, while the crescent is an enhancement formed due to the survival upwind of interstellar neutrals closer to the LISM source beyond the Heliopause and the higher upwind PUI cutoff energy resulting from the anti-parallel neutral to solar wind flow (Sokół et al. (2016)). While species with a low first ionization potential (FIP) are enhanced in the crescent on the upwind side of the Sun but are not found in the cone due to fast ionization, species such as helium with a high FIP can be found in both the upwind crescent and the cone on the downwind side (Moebius et al. (1995); Gloeckler et al. (2004); Drews et al. (2012); Gershman et al. (2014)).

Enhancements in the helium density in the cone and crescent regions are caused by the in-flow of neutral atoms. However, once these neutrals become ionized, they create measurable signatures of the helium focusing cone and crescent in the PUI data. It has been shown by Möbius et al. (2015a, 2016a,b); Taut et al. (2018); Bower et al. (2019) that λ_{ISN} can be measured indirectly by determining the upwind flow orientation using the PUI cutoff method. This method utilizes the variation of the cutoff in the PUI distribution function due to the dependence on ecliptic longitude of the PUI injection speed into the solar wind. This dependence is a result of the variation in the relative direction of the neutral interstellar wind particles to the solar wind with longitude, as described by Möbius et al. (1999). Other studies, such as by Gloeckler et al. (2004), Drews et al. (2012), and Gershman et al. (2013), have measured the flow direction from the orientation of the helium focusing cone. A more complete list of measured λ_{ISN} values reported in the literature are enumerated in Appendix 3.10.

In response to a debate regarding a possible decades-long variation in the flow direction, λ_{ISN}, (Frisch et al. (2013, 2015); Lallement & Bertaux (2014); Wood et al. (2015)), we use PUI measurements in the helium focusing cone to determine the flow direction, λ_{ISN}, over an 11-year solar cycle using ACE/SWICS He^+ double-coincidence PUI data. After finding no significant shift in λ_{ISN} within this data set, we determine a single flow direction on the combined data set. We note that we achieve comparable flow direction results to Taut et al. (2018) and Bower et al. (2019), though this method utilizes a fitting of the focusing cone using distribution values at the cutoff, while Taut et al. (2018) and Bower et al. (2019) directly fit the cutoff location.

3.3 Background: Pickup Ion Transport Considerations

Effects of the solar IMF on the PUIs lead to PUI transport, as reported by Möbius et al. (1996), Chalov & Fahr (1999), Chalov (2014), and Quinn et al. (2016), causing a spread and shift in the focused ions as viewed in the PUI data. This results in the center of the PUI cone to be offset relative to that of the neutrals, which corresponds directly to the interstellar wind longitudinal inflow direction, as discussed in Section 3.2. As described by Möbius et al. (2016a), these transport effects are minimized by finding the correct cutoff location in the distribution as a function of w', analyzed in the solar wind frame to account for varying IMF directions. While $w = v/v_{sw}$ represents the inertial, or observer frame (Gloeckler & Geiss (1998)), where v_{sw} is the solar wind speed, the solar wind frame can be represented by $w' = v/v_{sw} - 1$ (Möbius et al. (2015a)). To further minimize the effects of PUI transport on our measurements, we select only counts measured during times when the solar wind velocity vector is approximately perpendicular (90° ±20°) to the solar IMF vector to ensure the PUI torus is in the instrument's field of view, as described in Möbius et al. (2015a) and Bower et al. (2019), due to the findings reported by Chalov & Fahr (1998) and Drews et al. (2015), which indicate that these conditions are favorable to having the ring-beam within the field of view. This method helps to ensure that we are primarily analyzing PUIs that have been newly injected into the solar wind and therefore have not yet undergone significant transport effects. Additionally, we ensure that we are looking exclusively at the region of the distribution known to correspond most directly with interstellar PUIs by restricting the cutoff search region of the distribution to $0.7 \leq w' \leq 1.5$ as in Möbius et al. (2015a). This approach allows us to ensure that we are avoiding the regions of the distribution known to be dominated by the solar wind and by suprathermals, as well as avoiding inner source PUIs, which can generally lie in a w' range of up to 0.5, as reported in Gloeckler & Geiss (1998) and Drews et al. (2012). By finding the PUI cutoff point in this region rather than using a fixed value in w', we utilize the most pristine PUIs from very close to their production point, which further minimizes transport effects, as discussed in Quinn et al. (2016), while accounting for the shift in the cutoff in w' at varying longitudes resulting from the relative velocity of the radial solar wind and the injected PUIs, as reported by Möbius et al. (1999).

3.4 Background: Data Sources and Analysis

3.4.1 The ACE / Solar Wind Ion Composition Spectrometer (ACE/SWICS)

The Solar Wind Ion Composition Spectrometer (SWICS) on-board the ACE spacecraft is a triple-coincidence *in situ* ion mass spectrometer that samples the solar wind plasma with a look direction in the solar radial direction (Stone et al. (1998); Gloeckler et al. (1998)). Ions entering the instrument aperture are selected for energy-per-charge (E/q) according to instrument step by applying set voltages to the electrostatic analyzer (ESA) plates. Ions of the correct E/q at a given step will pass from the ESA into a time-of-flight (TOF) chamber through a thin carbon foil after being accelerated by a fixed post-acceleration voltage for more efficient detection and to improve the instrument resolution in TOF. Secondary electrons are generated from the carbon foil and guided by electric fields within the TOF chamber to a start signal micro-channel plate (MCP), while the original ion, likely neutralized by the carbon foil in the case of He^+ (Allegrini et al. (2016)), continues to a solid state detector (SSD), triggering an energy reading. Once the neutralized particle strikes the SSD, secondary electrons are again emitted and guided by the electric fields in the TOF chamber to a stop MCP, completing the sequence and providing a time-of-flight for the particle across a known distance through the chamber. In this way, it is possible to measure the particle's E/q, Energy, and TOF and to calculate velocity, charge, and mass. The three detections, or coincidences, with the start and stop MCP and the energy SSD comprise a complete triple-coincidence detection. If the particle is not sufficiently energetic to overcome the minimum detection threshold of the energy SSD, such as is often the case with low charge-state ions, such as PUIs, an energy reading may not be made, but the start and stop timing coincidences may still be recorded. Such a sequence is therefore considered a double-coincidence detection, and while it does not provide a direct energy reading, it does provide useful and sufficient measurements for the purposes of this study. This is especially true for He^+, having a mass per charge of 4, which is distinguishable from other ions in the solar wind in the E/q range of the PUI distribution. We therefore include all double-coincidence measurements in our data set, which includes the sub-set of triple-coincidence measurements.

3.4.2 Data Sources

We obtain data necessary for the PUI analysis from the following sources:

Solar wind velocity and IMF magnetic field vectors in RTN coordinates The solar wind velocity and IMF directional components are obtained in the RTN coordinate system for comparing perpendicularity through ACE/MAG and ACE/SWEPAM data from the ACE Science Center and linearly interpolated to the mid-point of the ACE/SWICS 12-minute data collection measurement times.

Solar wind speed derived from proton speeds The solar wind speed is taken as the proton speed measured by ACE/SWICS obtained from the Ace Science Center and interpolated to the required measurement resolution.

ICME start and end times for the ACE spacecraft ICME start and end times are obtained from the Richardson & Cane ICME database for the ACE spacecraft (Cane & Richardson (2003); Richardson & Cane (2010)).

Earth ecliptic longitudes Earth ecliptic longitudes are obtained from NASA HelioWeb (2017) and linearly interpolated to the required 12-minute resolution.

ACE/SWICS He$^+$ count distribution functions The ACE/SWICS instrument collects data into 60 set E/q bins (Gloeckler et al. (1998); Gilbert et al. (2014)). However, we neglect the outer two bins due to measurement uncertainty, such as from the ESA ramp-up, which causes a large uncertainty in E/q. Since the PUI distribution is not found at the highest or lowest energy bins measured by ACE/SWICS, restricting the distribution to the 58 center bins does not affect our results.

Count distribution functions are binned from native ACE/SWICS He$^+$ double-coincidence measurements using a technique according to the following process. All counts identified as He$^+$ using forward modeling are selected and accumulated for a given 12-minute time window in each of the center 58 E/q bins. The modeled value is taken as the E/q–TOF track for He$^+$ with an acceptance range of $\pm 5\%$, as shown in Figure 3.2. Additionally, protons are removed by using a cutoff filter in TOF at 85% the proton speed, which only removes measurements at speeds slower than the PUI portion of the distribution, and therefore does not affect our analysis. The instrument noise signatures shown in Figure 3.2 are removed by selecting TOFs above 75 ns, which only removes measurements in the suprathermal range, and therefore does not affect the PUI portions of our distributions and our analysis (Gilbert et al. (2014)).

3.5 Methodology

Accumulating distribution functions into a data set spanning 13 complete solar orbits and focusing cone crossings which occur between the years 1998 and 2010, we measure 11 values for the interstellar wind inflow direction, λ_{ISN}. We measure the flow direction over individual 3-orbit (3-year) boxcar averages centered on orbits with focusing cone crossings occurring between 1999 and 2009. We then fit those values to a best fit trend line. We additionally measure a unified flow direction for the data set spanning all 13 complete orbits.

We measure flow directions by analyzing combined averaged count distribution functions of ACE/SWICS He$^+$ double-coincidence pickup ion measurements for the count value at the PUI cutoff. The cutoff is found as a function of ecliptic longitude using a method first reported by Möbius et al. (2015a) as described in our methodology. We then fit the cutoff count values as a

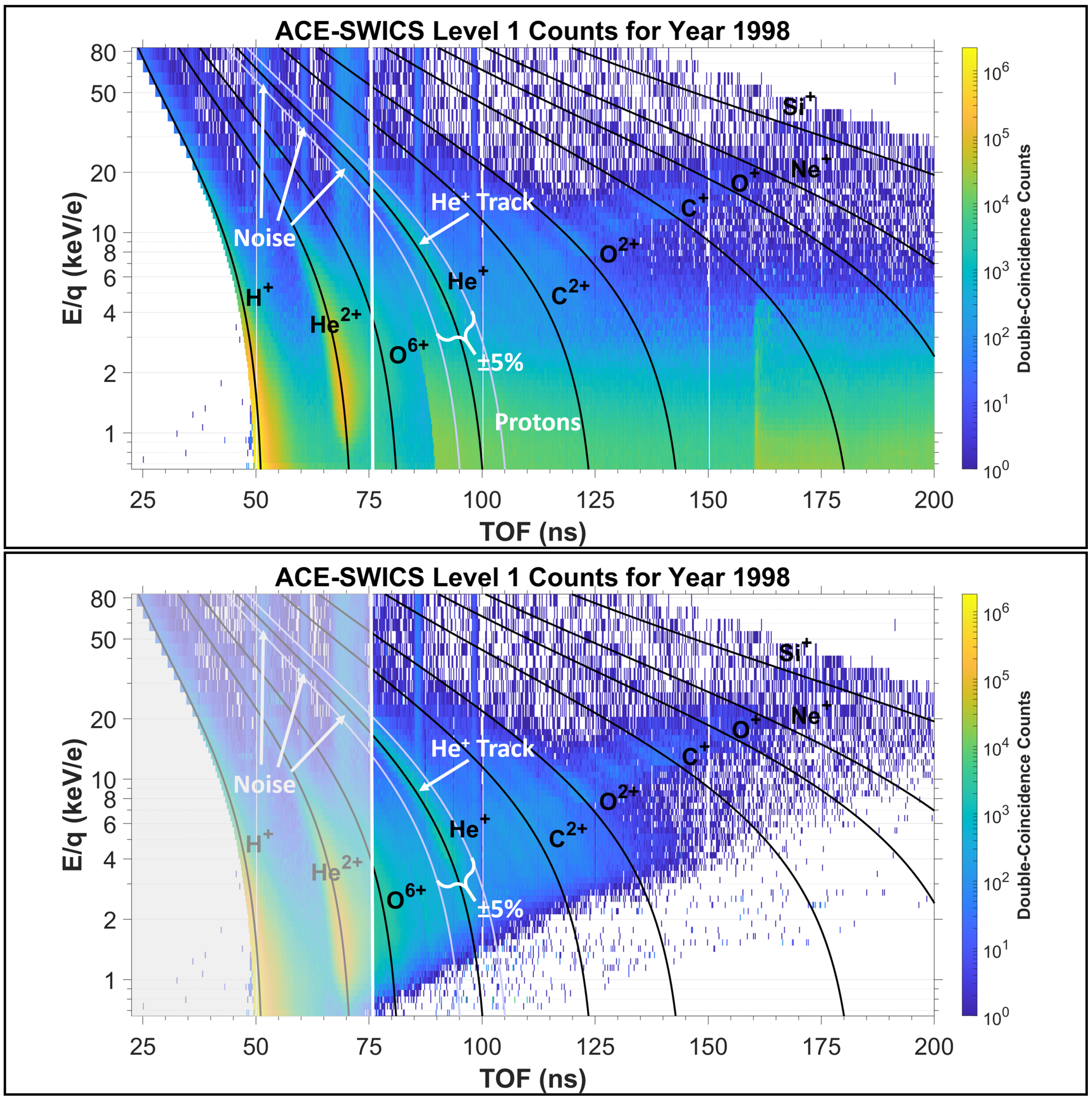


Figure 3.2: 2D histogram plots of double-coincidence counts plotted in E/q against TOF. The He$^+$ track is marked along with the surrounding region of margin ±5%. Vertical stripes associated with instrument noise are identified. In the count distribution accumulation, all TOF values for He$^+$ are neglected below 75 ns. Additionally, as shown in the bottom figure, protons are removed by using a filter in TOF of 85% the proton speed.

function of ecliptic longitude to a kappa function whose peak location is taken as the flow direction, λ_{ISN}. We take various measures to account for solar transients, solar cycle variability, and PUI transport. We report here on our detailed methodology.

3.5.1 Binning Native Resolution ACE/SWICS He$^+$ Count Distributions

The full data set of ACE/SWICS He$^+$ double-coincidence measurements from 1998 through 2011 is accumulated into count distributions at the instrument native 12-minute resolution integrated over all instrument sectors. Only those measurements taken at times when the solar wind velocity vector is approximately perpendicular to the solar IMF are selected, as previously described.

For each 12-minute count distribution function, we convert the 58 ACE/SWICS center E/q bins to particle velocity, v, in km/s similarly to Equation 1 from Gloeckler et al. (1998) as:

$$v = \sqrt{\frac{2e(E/q)}{1000(m/q)(m_u)}} \tag{3.1}$$

in the spacecraft frame, where e is the elemental charge, E/q is energy per charge, m/q is mass per charge, and m_u is the atomic mass unit. We then normalize the ion speed, v, to the solar wind speed, v_{sw}, as $w = v/v_{sw}$ (Gloeckler & Geiss (1998)). Similarly to Möbius et al. (2015a, 2016a,b) and Bower et al. (2019), we transform the PUI distribution into the solar wind frame according to:

$$w' = \frac{v}{v_{sw}} - 1 \tag{3.2}$$

Since ACE/SWICS is located at the Earth L1 position, measurement times are mapped to Earth ecliptic longitudes.

3.5.2 Measuring the Flow Direction from Distribution Values at the Cutoff in the Pickup Ion Distributions

3.5.2.1 Binning 3-orbit Boxcar Averaged 1° Count Distributions

Utilizing a method similar to that outlined in Möbius et al. (2015a) and Bower et al. (2019), we locate the PUI cutoff locations in ACE/SWICS He$^+$ double-coincidence count distribution functions as described below. Figure 3.3 shows a 2D histogram of 1° normalized averaged count distribution functions against ecliptic longitude for the example 3-orbit boxcar centered on the focusing cone crossing occurring in 2009 as well as for all orbits.

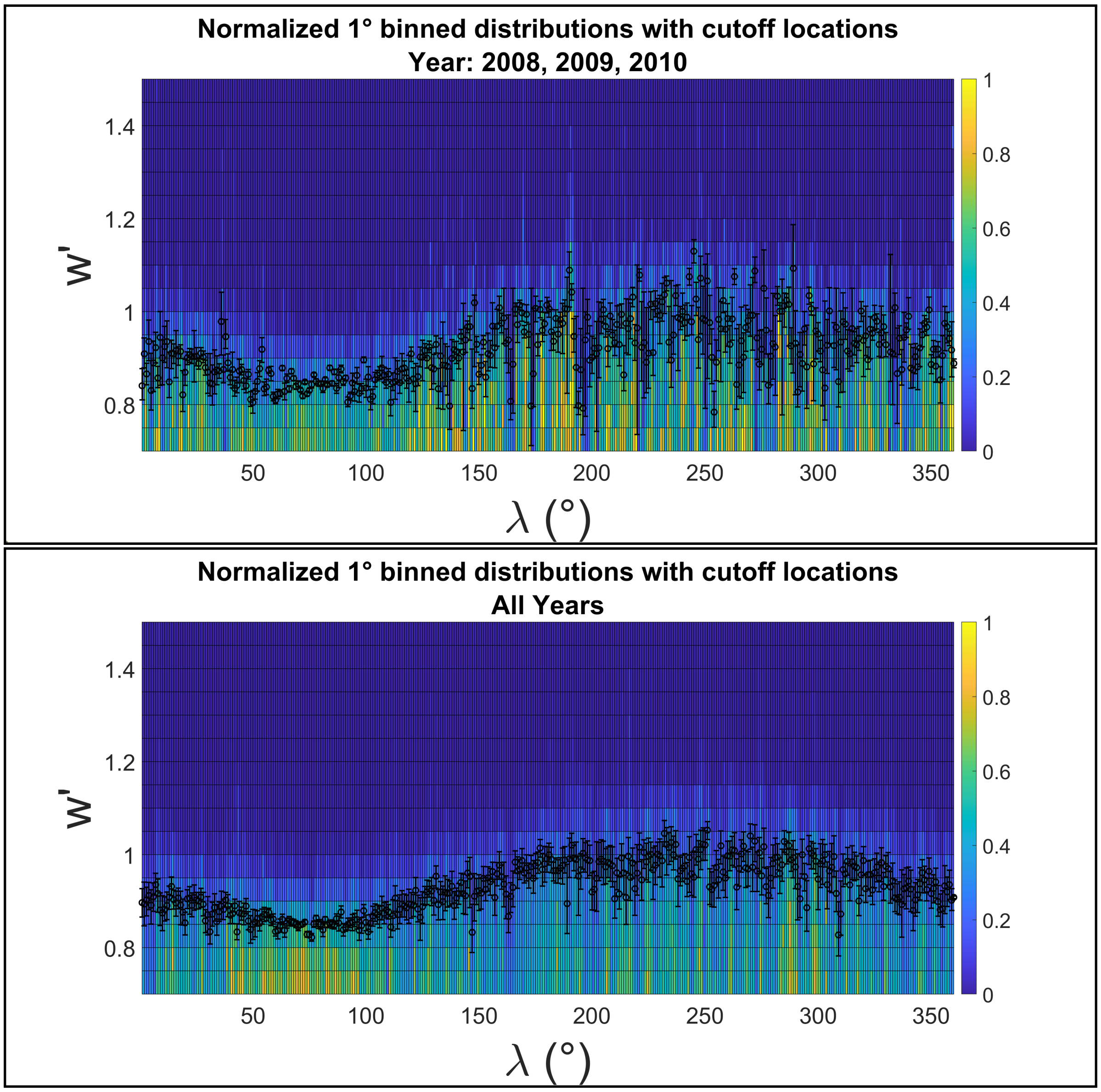


Figure 3.3: Cutoff locations with error bars in w' plotted as black circles over a 2D histogram of normalized averaged count distribution functions per 1° bin in ecliptic longitude for the 3-orbit boxcar centered on the focusing cone crossing occurring in 2009 as well as for all combined orbits. Averaged count distribution functions are normalized per 1° bin here for visualization purposes only to show the cutoff location with respect to each distribution function.

We bin PUI count distributions, as described and utilized by Möbius et al. (2015a, 2016a,b), into 1° bins in ecliptic longitude for a given data subset, then appky a 3-orbit moving boxcar average. The moving boxcar method allows us to average out effects of transients, similarly to Drews et al. (2012), who implement a sliding average within their data set and achieve flow direction calculations from a set of 4 STEREO A/PLASTIC orbits, using subsets with 2 to 4 orbits for calculations in their second method. Using this method, we maintain any overall trend information in the full data set. Additionally, when combining orbits in a given boxcar window, we average the distributions to further reduce effects of solar activity, such as transients, from individual years on the combined data set.

We report our measurements for a given orbit according to the year in which the focusing cone crossing occurs and label each 3-orbit boxcar either with the 3 corresponding years or with the associated center year. For a given orbit, however, it may be noted that the portion of the orbit during which ACE traverses larger ecliptic longitudes, including the crescent crossing, occur in the calendar year following the associated focusing cone crossing, which takes place late in the labeled year.

3.5.2.2 Locating the Value at the Cutoff in the Pickup Ion Distributions

We combine and average 1° binned count distribution functions for 3-orbit boxcars and analyze them as a function of w', neglecting data points which fall within an ICME time window, as they introduce flux enhancements and magnetic field distortions which may influence the signatures we are looking for and incorrectly bias our measurements. We first smooth each 1° averaged count distribution using a 5-point window Gaussian smoothing algorithm, neglecting the end of the distribution where no counts are recorded, and then restrict the w' axis to $0.7 \leq w' \leq 1.5$ as previously described. We then designate the w'-cutoff to lie at the linearly interpolated half-maximum of the count distribution, as in Bower et al. (2019), shown in Figure 3.4.

3.5.2.3 Measuring the Flow Direction from a Kappa Fit of He^+ Counts as a Function of Ecliptic Longitude

The focusing cone signature can be seen as an enhancement in the PUI flux, represented by the value of the distribution measured at the cutoff as a function of ecliptic longitude, as seen in Figure 3.5. We then fit these data using a kappa distribution, as indicated by Swaczyna et al. (2019), to determine the peak location, λ_0, corresponding to λ_{ISN}.

The kappa fit to the measured distribution values at the cutoff for each 1° averaged count distribution is demonstrated in Figure 3.5. The kappa functional form is given as:

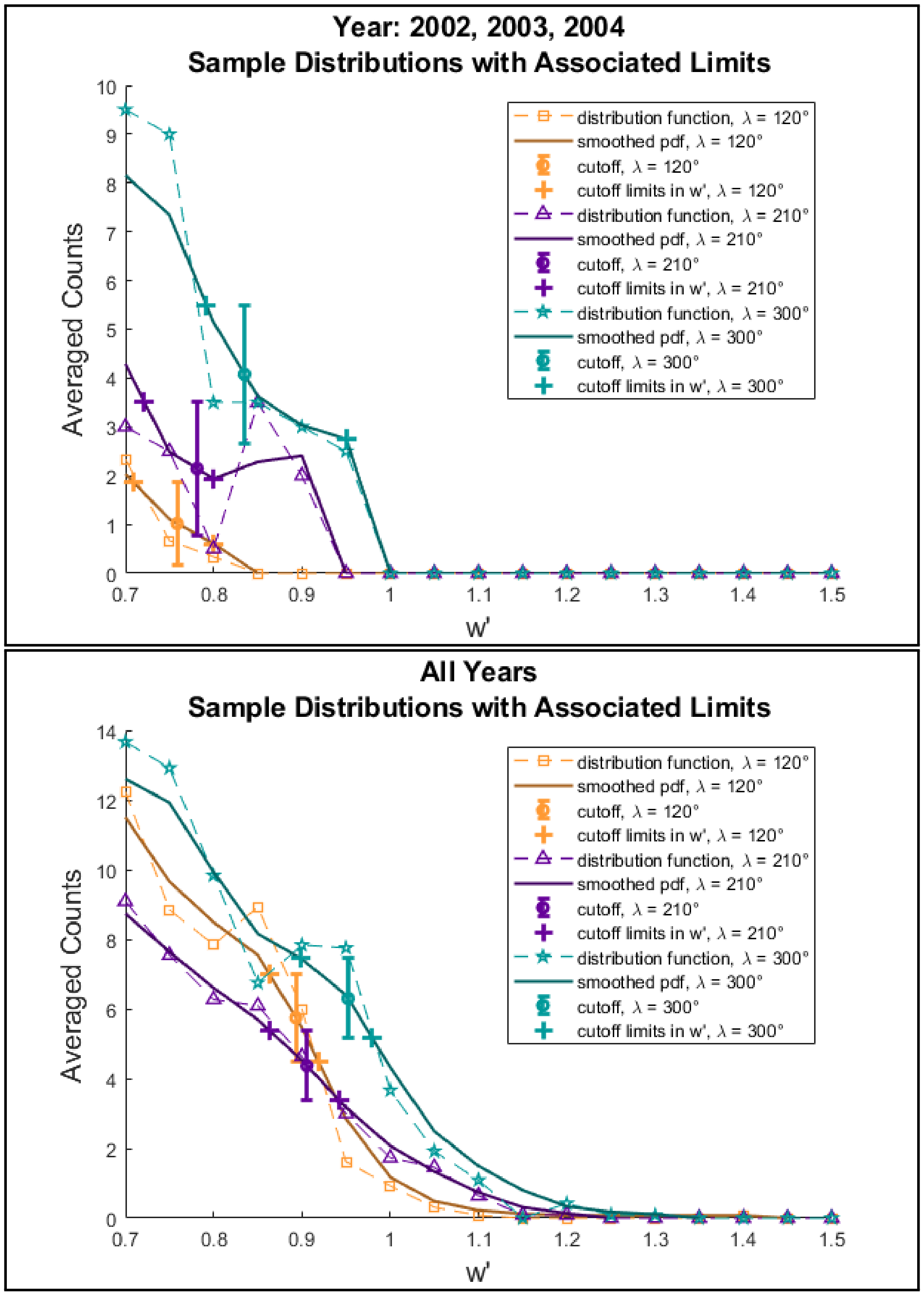


Figure 3.4: 3-orbit boxcar averaged count distribution functions centered on the cone crossing occurring in 2003 as well as all-orbit averaged count distributions plotted against w' in the range $0.7 \leq w' \leq 1.5$ for ecliptic longitudes $\lambda = 120°$ (orange squares), $210°$ (purple triangles), and $300°$ (teal stars) along with the corresponding smoothed distributions, the distribution value at the cutoff with statistical errors, and the errors projected as limits in w' in the corresponding color. The cutoff point can be seen to shift with ecliptic longitude.

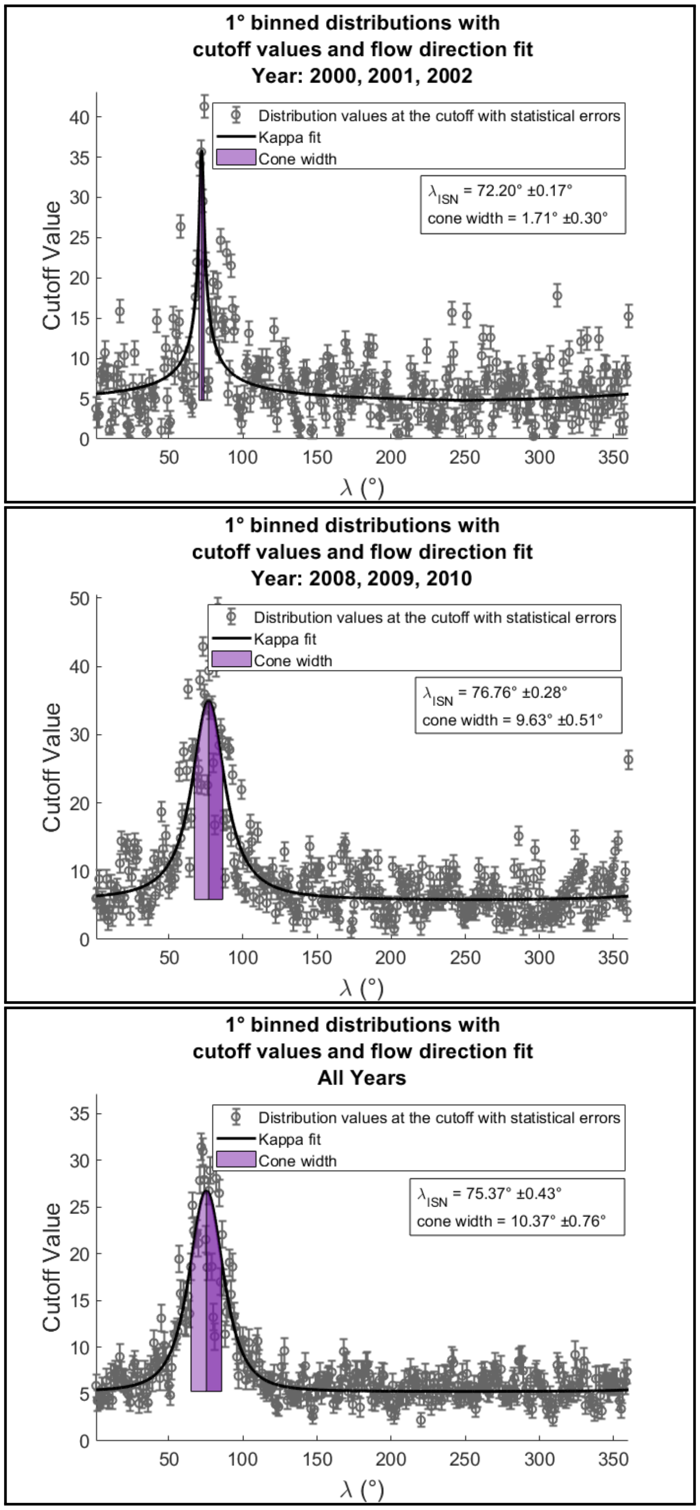


Figure 3.5: Averaged count values at the PUI cutoff per 1° bin in ecliptic longitude plotted with statistical errors and kappa fit for years 2001 and 2009 as well as the aggregate data set. The cone width, given by the parameter x_1 from the kappa fit, is shown to the right and left of the cone center in the purple shaded regions. The cone center with fitted errors corresponding to λ_{ISN} along with the cone width parameter and fitted errors are reported on the plots.

$$f_\kappa(\lambda) = y_1(1 + \frac{(\lambda - \lambda_0)^2}{2x_1^2 x_2})^{-x_2} + y_2 \tag{3.3}$$

where λ_0 is the fitted parameter for λ_{ISN}, y_1 is the kappa height, y_2 is the offset along the counts axis, x_1, is the upper kappa width, and x_2 is related to the lower kappa width.

3.5.3 Determining the Flow Direction Trend Over Time

Employing the kappa fitting function over the distribution values as a function of ecliptic longitude, we determine a value and uncertainties for λ_{ISN} for each 3-orbit boxcar and plot the 11 results together by boxcar center year in Figure 3.6. We then perform a linear fit with weights taken as the inverse square of the statistical measurement fit errors to determine the overall trend across the data set. The error of the slope is reported as the error computed by the fitting algorithm.

3.5.4 Measuring an Aggregate Data Set Flow Direction

For the combined data set including cone crossings from 1998 through 2010, we repeat the method described above to calculate a single combined flow direction from the full data set for all complete orbits. We average across the combined distributions to reduce solar cycle effects that would bias our result by giving a higher influence to orbits with higher overall count rates.

3.6 Results

3.6.1 Individual Flow Direction Measurements and Trend

Our full data set spans from 1998 through 2011, comprising 13 complete orbits including traversals of the focusing cone during the years 1998 through 2010. Using a 3-orbit boxcar, we are able to measure 11 individual flow directions, λ_{ISN}, providing us trend information to track λ_{ISN} over an 11-year solar cycle, as shown in Figure 3.6. We report the individual flow direction measurements in Table 3.1. The linear fit to longitude over time yields a slope of 0.00° $\pm$0.51° / yr with a χ^2 fit p value of 1, indicating that we do not detect a varying trend. Some fluctuations can be seen beyond the reported statistical fit error. These fluctuations are discussed in more detail in Section 3.7.2.

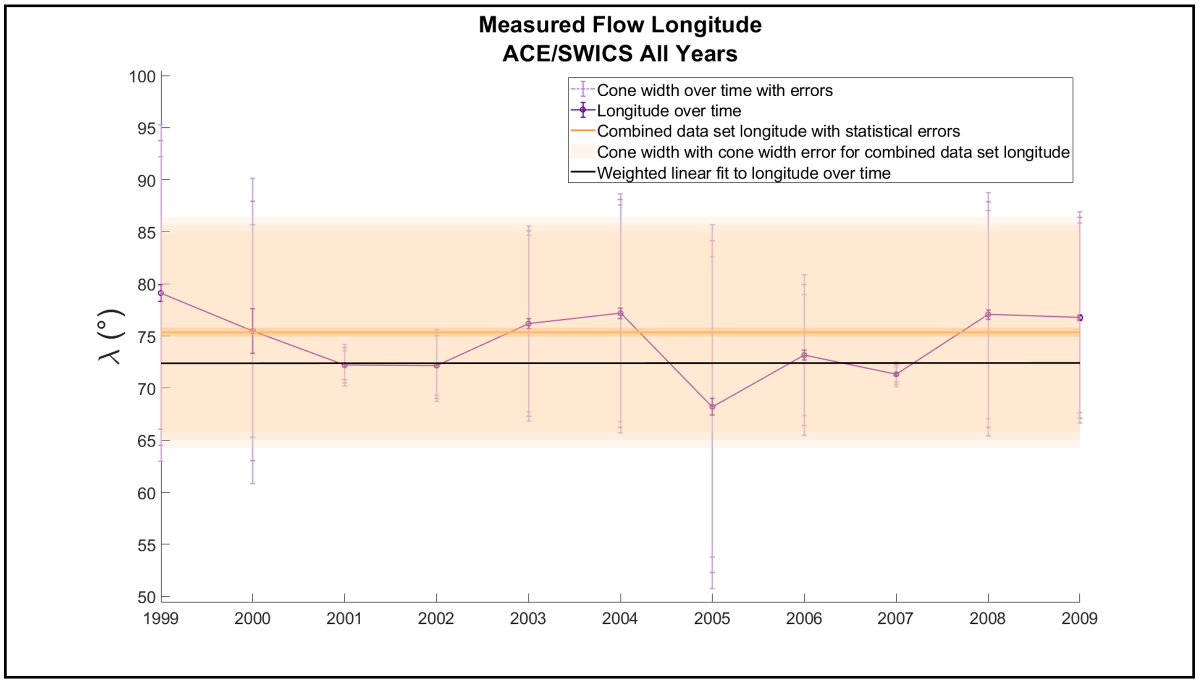


Figure 3.6: λ_{ISN} values are plotted with statistical fit errors as dark purple circles against 3-orbit boxcar center focusing cone crossing year along with cone widths and their associated errors from the fitted x_1 value plotted in light purple. The combined data set λ_{ISN} value with statistical fit error is plotted as a dark orange line with a dark orange shaded region. The overall measured flow direction, λ_{ISN}, is reported as 75.37° ±0.43°. The combined cone width from the fitted x_1 value with fitted errors is plotted as a lighter orange shaded region, extending from ∼65° to ∼85° in ecliptic longitude. The linear fit of the individual yearly λ_{ISN} values using weights derived from the statistical fit errors is plotted as a black line and yields a slope of 0.00° ±0.51° / yr.

3.6.2 Unified Flow Direction

We apply the 1° binning process for the combined data set with all 13 orbits, and by repeating the process described above, we measure a flow direction, λ_{ISN}, of 75.37° ±0.43°. This value is plotted as the flow longitude for the combined data set in Figure 3.6. In Appendix 3.10, we compare our final results to other values published in the literature in Table 3.2 and plot the these literature values by year published along with our own results in Figure 3.7. Using our new method combining the PUI cutoff location with a fit to the focusing cone orientation, we achieve a value consistent with recent literature results. We discuss these results in the context of the literature in more detail in Section 3.7.3.

Table 3.1: λ_{ISN} measurements by orbit yielding an overall linear fit with a slope of 0.00° ±0.51° / yr with a χ^2 fit p value of 1.

Year	λ_{ISN}	Variance	Fit Value (R^2)
1999	79.15°	±0.78°	0.96
2000	75.47°	±2.14°	0.42
2001	72.20°	±0.17°	0.96
2002	72.17°	±0.18°	0.96
2003	76.19°	±0.47°	0.96
2004	77.18°	±0.52°	0.96
2005	68.21°	±0.81°	0.97
2006	73.17°	±0.50°	0.97
2007	71.33°	±0.10°	0.97
2008	77.07°	±0.46°	0.98
2009	76.76°	±0.28°	0.98
Combined Data Set	75.37°	±0.43°	0.97

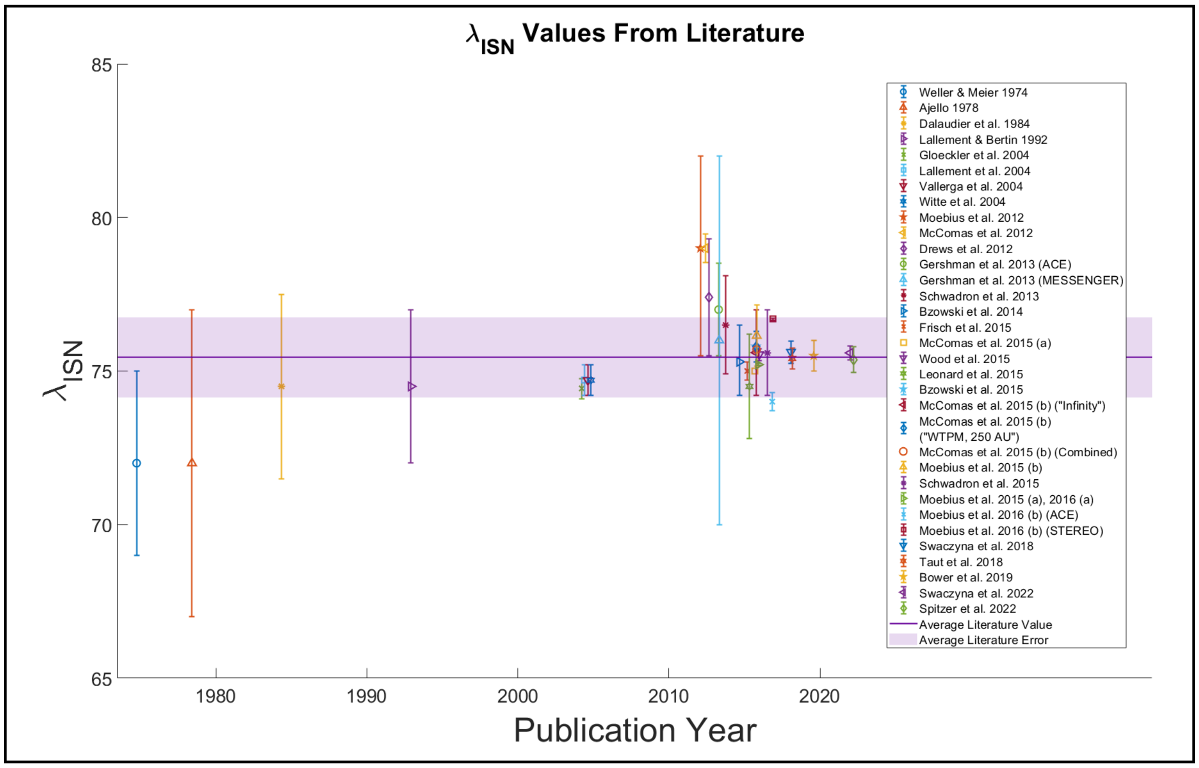


Figure 3.7: Measured unified flow direction of λ_{ISN} = 75.37° ±0.43° plotted together with values from the literature as reported in Table 3.2 in Appendix 3.10. λ_{ISN} values are plotted against year of publication along with the average for all plotted values and errors.

3.7 Discussion

3.7.1 Error Estimation

The statistical error of each data point in the count distribution is estimated using the Poisson statistic, or the square of the number of counts, $\varepsilon_{dist} = \sqrt{n}$, where n is the number of counts. A data point with 0 counts is considered a non-measurement and disregarded.

Orbits combined into averaged count distribution functions are assigned errors as the mean of the individual statistical errors taken in quadrature such that the combined error for a point i in the distribution is given as:

$$\varepsilon_{combined,i} = \frac{1}{N_{orbits}} \sqrt{\sum_{j=1}^{N_{orbits}} (\sqrt{n_{i,j}})^2} = \frac{1}{N_{orbits}} \sqrt{\sum_{j=1}^{N_{orbits}} n_{i,j}} \tag{3.4}$$

where N_{orbits} is the number of orbits being combined and $n_{i,j}$ is the number of counts in the ith measurement in the count distribution of the jth orbit.

The error of the count distribution at the cutoff is calculated by linear interpolation of the errors between the two associated points if the value at the cutoff is found using linear interpolation.

In the kappa fitting function, we use weights as one over the squared error of a value measured at the cutoff for a given longitude. The error for the cone center is then reported as the error computed in the fitting algorithm.

When smoothing each distribution function before locating the PUI cutoff, we utilize a 5-point Gaussian smoothing algorithm so that the center point has the highest weighting, with less influence from points farther away. For smoothed distribution points, we take the error as the quadrature errors of the points used in the smoothing for any given point, weighted by the same factor used in the smoothing algorithm. For a moving Gaussian smoothing algorithm of window size N_{window}, the quadrature error taken at each smoothed point is given as:

$$\varepsilon_{smoothed,i} = \sqrt{\sum_{j=1}^{N_{window}} (G_j \varepsilon_j)^2} \tag{3.5}$$

where ε_j is the statistical error at a given point and G_j is the Gaussian weighting factor for a given point within the window.

3.7.2 Fluctuations in Individual Reported Values

Considering the fluctuations that vary beyond the reported statistical errors for the individual measurements, we note that there are likely effects of solar activity, such as compression regions, on the focusing cone beyond those explicitly accounted for in our method, which include the removal of ICMEs and the application of a 3-orbit boxcar, and note that the reported error is given solely as the statistical measurement error.

As noted by Möbius et al. (2010), these measurements can be affected by compression regions, which can lead to seeming local enhancements of the PUI fluxes in the focusing cone, the region containing the highest interstellar helium density. Compression regions, such as stream-stream interaction regions, can lead to even higher local densities that cause the focusing cone to appear more peaked or fragmented, either reducing the apparent width, such as seen for center year 2001 in Figure 3.5, or widening it. In the example for center year 2001, the enhancement, shown in Figure 3.8, likely makes the focusing cone appear more peaked with a narrower width and a cone center that is pulled to a lower longitude, resulting in a lower measured value for λ_{ISN}. While we do not explicitly quantify this systematic error or remove these effects which strongly increase the variations between individual measurements in this study, we see that these effects average out over our data set which spans a full 11-year solar cycle.

Solar activity affects the plasma flow within the Heliosphere, while the solar IMF additionally affects PUIs and their transport. While these effects do not impact the inflow direction, λ_{ISN}, of the neutral interstellar wind, they do play a role in the shape and location of the focusing cone, which we use to measure the flow direction. Therefore, while we do not completely account for all of these effects directly in our reported measurement error, we plot the value x_1 from our kappa fit as a representation of focusing cone width to visualize this additional source of error, which may account for some of the fluctuations. Variations in these cone widths are demonstrated in the kappa fit for the 3-orbit center years of 2001 and 2009 as well as for all 3-orbit center years and the combined data set in Figure 3.6.

While we do see effects from features such as compression regions, which may occur more frequently during solar maximum years, we do not see a specific solar cycle dependent trend in this data set, with the maximum of Solar Cycle 23 occurring around the year 2001 and the minimum of Solar Cycle 24 occurring around the year 2008. Also, we do see that the fluctuation effects average out when the entire data set over the 11-year solar cycle is used.

3.7.3 Comparison with Literature Values

The first measurements for the inflow direction λ_{ISN} were inferred early on using ultraviolet back scatter techniques by by Bertaux & Blamont (1971); Thomas & Krassa (1971); Weller & Meier

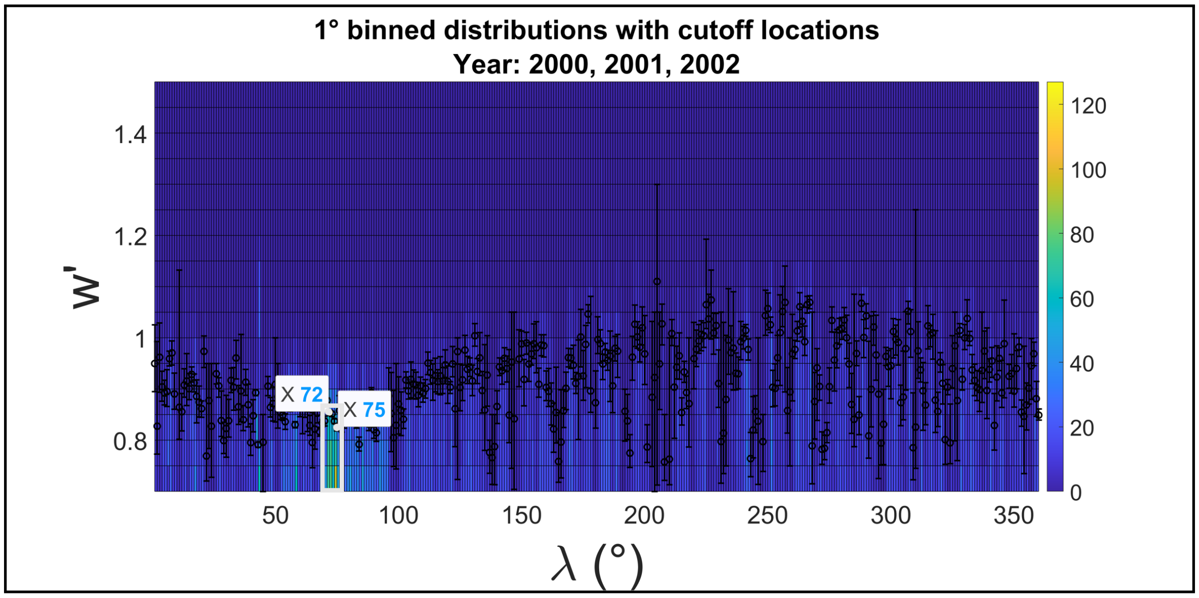


Figure 3.8: 2D histogram of combined averaged count distribution functions along with measured cutoff locations plotted as black circles with associated error bars for the 3-orbit boxcar center year 2001. Approximate longitudes are marked for the measured 2001 3-orbit flow direction (∼72°) and the measured overall flow direction (∼75°), and a white box indicates a higher flux region centered on a longitude to the left of the expected cone center, which is likely affecting the kappa fit and associated measured longitude. Distribution functions are not normalized here in order to demonstrate the enhancement in averaged combined counts.

(1974), and the first *in situ* measurements of PUIs were made by Möbius et al. (1985). These early approximations were novel at the time, but showed relatively low λ_{ISN} values with large error bars compared to the values published more recently.

Gloeckler et al. (2004) use ACE/SWICS He$^+$ PUI data from 1998 to 2002 and additionally analyze Nozomi data from 2000 and AMPTE data for 1984 and 1985 to look for signatures of the focusing cone. While the authors do not make the same considerations for PUIs and transport effects as taken in this study, they achieve a value closer to the literature average reported flow direction by using a modeling technique to account for inner source PUIs and ionization rates and taking large averages over the data, namely 9 days, or ∼8.9°, on the ACE/SWICS data and 15 days, or ∼14.8° on the Nozomi data. While the reported flow direction of 74.43° ±0.33°, while still at a lower longitude, is closer to the reported literature average, the reported error does not take into account all effects, and the large averaging technique might not lend itself to determining an accurate yearly flow direction. Other studies using PUIs in the helium focusing cone, such as Drews et al. (2012) and Gershman et al. (2013), report flow directions at higher longitudes than the most recent values. These values are likely affected by PUI transport.

Other methods reported in the literature as referenced in Table 3.2 in Appendix 3.10 use measurements of neutrals, while Möbius et al. (2015a, 2016a,b), Taut et al. (2018), and Bower et al. (2019) implement the PUI cutoff method from Möbius et al. (2015a). It can be seen that from about 2014, the literature values start to converge to $\sim 75°$.

While our method utilizes signatures of the focusing cone in the He$^+$ PUI data, we make specific considerations for PUI transport, utilizing the PUI cutoff location. It can be seen that our reported overall flow direction of $75.37° \pm 0.43°$ is not only consistent with these other literature values and the literature average, but very much in alignment with the values reported using the cutoff method by Taut et al. (2018) and Bower et al. (2019).

3.8 Conclusions

Over the ACE/SWICS He$^+$ double-coincidence data set spanning from 1998 through 2011, we track the longitudinal inflow direction, λ_{ISN}, of the interstellar wind through the Heliosphere using 13 complete solar orbits with focusing cone crossings occurring between the years 1998 and 2010. We determine a yearly flow direction for 3-orbit boxcar center years over an 11-year solar cycle using a kappa fit to the averaged count distribution values measured at the PUI cutoff. To account for flux enhancements and magnetic field disturbances that would bias our analysis, we remove measurement times associated with CMEs, while other systematic but stochastically distributed effects are mitigated with multi-year averaging and an 11-orbit center year span. We additionally analyze relatively pristine PUIs by measuring them near the PUI cutoff in w' and selecting for perpendicular solar wind velocity to IMF vector conditions. We fit the measured values taken at the PUI cutoff per $1°$ of ecliptic longitude using a kappa distribution with weights of one over the squared errors. The error for the cone center is then reported as the error computed in the fitting algorithm. The kappa center is taken as the longitude of the cone crossing, which directly corresponds to the longitudinal interstellar flow direction, λ_{ISN}.

The method we report here utilizes the focusing cone signature, which becomes apparent when analyzing the count distribution data as a measure of the higher flux found in the cone region. While the direct PUI cutoff method utilizes the signature of the broader crescent, which becomes apparent when directly analyzing the shift in the cutoff location with ecliptic longitude, we also utilize the region of the distribution near the PUI cutoff, mitigating PUI transport effects. These methods prove to be complementary, and we note that our reported flow direction is consistent with recent reported literature values, being very much in alignment with recent flow directions calculated using the direct PUI cutoff method. This consistency validates our new method and its usefulness in determining the trend of the flow over this data set.

While the shape of the focusing cone, especially as measured in the PUI data, may be affected by solar activity, as PUIs transport along the IMF and compression regions may appear as spikes in the focusing cone signature, these effects do not impact the flow direction of the interstellar neutrals into the Heliosphere, which is caused by the relative motion of the Heliosphere and the LISM. Additionally, if proper considerations are made for solar transients and solar cycle, the trend may still be measured from focusing cone orientation over time, especially over a complete solar cycle. Over the reported 11 center years, we measure a slope of 0.00° $\pm$0.51° / yr, indicating that we do not detect a varying trend over these years (Figure 3.6). This implies that we do not measure Heliospheric-scale disturbances in the local interstellar bulk flow on the order of decades.

Similarly to the 3-orbit boxcar cases, we repeat the reported focusing cone kappa fitting process for the combined data set of 13 complete orbits and determine a unified flow direction, λ_{ISN}, from these data using this method to be 75.37° $\pm$0.43°. This value is listed together with other literature values in Appendix 3.10 in Table 3.2 and plotted together with the these literature values by year published in Figure 3.7. While some fluctuations may be noted beyond the reported statistical error, we note that these are likely influenced by solar activity, namely compression regions, and our results appear to remain relatively consistent over an 11-year solar cycle.

3.9 Acknowledgments

We thank the ACE/SWICS, ACE/MAG, and ACE/SWEPAM instrument teams and the ACE Science Center for providing the ACE data.

This work is supported by NASA's Solar Orbiter and Advanced Composition Explorer missions as well as NASA grants AGS 1358268 for J. Gilbert and S. T. Lepri, NNX16AP03H and 80NSSC20K0192 for S. T. Lepri, and 80NSSC18K1212 for E. Möbius and J. Bower. This work has additionally been supported by the Michigan Plasma Sciences & Engineering Graduate Fellowship, The Michigan Space Grant Consortium Graduate Fellowship, the American Geophysical Union Student Travel Grant, the University of Michigan Rackham Student Travel Grant, the University of Michigan Diversity, Equity, and Inclusion Grant in the Department of Climate and Space Sciences & Engineering, and the University of Michigan College of Engineering Office of Student Affairs Student Travel Grant.

3.10 Appendix: Literature Values

Table 3.2: λ_{ISN} values from the literature.

Value	Trend	Publication	Data Source	Reference
72° ±3°	N/A	Oct 1974	STP 72-1	Weller & Meier (1974)
72° ±5°	N/A	Jun 1978	Mariner 10	Ajello (1978)
74.5° ±3°	N/A	May 1984	Prognoz 6	Dalaudier et al. (1984)
74.5° ±2.5°	N/A	Dec 1992	Aurélie spectrometer Observatoire de Haute-Provence	Lallement & Bertin (1992)
74.43° ±0.33°	N/A	Nov 2004	AMPTE-IRM ACE Nozomi	Gloeckler et al. (2004)
74.7° ±0.5°	N/A	Nov 2004	EUVE SOHO	Lallement et al. (2004)
74.7° ±0.5°	N/A	Nov 2004	EUVE	Vallerga et al. (2004)
74.7° ±0.5°	N/A	Nov 2004	Ulysses	Witte (2004)
79.0° -3.5°, +3°	N/A	Feb 2012	IBEX	Möbius et al. (2012)
79.0° ±0.47°	N/A	Jun 2012	IBEX	McComas et al. (2012)
77.4° ±1.9°	N/A	Sep 2012	STEREO A	Drews et al. (2012)
77.0° ±1.5°	N/A	Apr 2013	ACE	Gershman et al. (2013)
76.0° ±6.0°	N/A	Apr 2013	MESSENGER	Gershman et al. (2013)
76.5° ±1.6°	N/A	Oct 2013	IBEX	Schwadron et al. (2013)
75.3° -1.1°, +1.2°	N/A	Sep 2014	Ulysses	Bzowski et al. (2014)
75.0° ±0.3°	+5.6° ±2.4° / 40 yrs +0.14° ± 0.06° / yr	Mar 2015	Nozomi IBEX STP 72-1 Mariner 10 SOLRAD 11B	Frisch et al. (2015)

75.54° ±0.19°	<0.3° / 13 yrs <0.023° / yr (No trend detected)	Mar 2015	Ulysses	Wood et al. (2015)
74.5° ±1.7°	N/A	May 2015	IBEX	Leonard et al. (2015)
75.8° ±0.5°	N/A	Oct 2015	IBEX	Bzowski et al. (2015)
∼75°	N/A	Mar 2015	IBEX	McComas et al. (2015a)
75.6° ±1.4°	N/A	Oct 2015	IBEX	McComas et al. (2015b) (UNH, “Infinity”)
75.8° ±0.5°	N/A	Oct 2015	IBEX	McComas et al. (2015b) (WTPM, 250 AU)
∼75.7°	N/A	Oct 2015	IBEX	McComas et al. (2015b) (Combined)
76.15° ±1°	N/A	Oct 2015	IBEX	Möbius et al. (2015b)
75.6° ±1.4°	N/A	Oct 2015	IBEX	Schwadron et al. (2015)
75.21° ±0.04°	N/A	Dec 2015 Mar 2016	STEREO A	Möbius et al. (2015a) Möbius et al. (2016a)
74.0° ±0.3°	N/A	Nov 2016	ACE	Möbius et al. (2016b)
76.69° ±0.04°	N/A	Nov 2016	STEREO A	Möbius et al. (2016b)
75.62 ±0.36°	N/A	Feb 2018	IBEX	Swaczyna et al. (2018)
75.41° ±0.34°	N/A	Mar 2018	STEREO A	Taut et al. (2018)
75.5° ±0.5°	N/A	Aug 2019	STEREO A	Bower et al. (2019)
75.59 ±0.23	N/A	Jan 2022	IBEX	Swaczyna et al. (2022)
75.37° ±0.43°	0.00° ±0.51° / yr (No trend detected)	2022	ACE	Spitzer et al.

CHAPTER 4

Implications of Transport and Challenges in Interstellar Flow Measurements Using Pickup Ions

The work presented in this chapter is intended to lead to future publication and was written in collaboration with Susan T. Lepri, Jim M. Raines, Jason A. Gilbert, Eberhard Möbius, Jonathan Bower, and Ryan M. Dewey.

4.1 Abstract

Pickup ions (PUIs) are a key population useful for studying *in situ* plasmas that influence the Heliosphere, as they retain signatures relating to the Heliosphere — interstellar interactions. However, numerous challenges can arise when analyzing PUI measurements. Due to the physics that govern their pickup and subsequent transport, careful considerations are required when analyzing these measurements. These considerations considerably restrict the available data set, which is already limited due to measurement capabilities, as current spaceflight instruments are not optimized for the low charge-state PUIs, which are dominated in the distribution by solar source ions, especially in the solar radial look direction. We discuss the effects of these considerations on our efforts to quantify the nature and evolution of the Heliosphere — interstellar interactions by measuring the longitudinal inflow direction of the interstellar wind through the Heliosphere using the PUI cutoff method in the upwind crescent on ACE/SWICS He^+ measurements. We further discuss the impacts of PUI considerations on other studies from the literature as well as requirements for improved future analysis and measurement techniques.

4.2 Introduction

4.2.1 Heliosphere – Interstellar Interactions

The Sun moves through the Local Interstellar Medium (LISM), filling the surrounding space with the solar wind and interplanetary magnetic field (IMF), which expand into a region known as the Heliosphere, until reaching the boundary, known as the Heliopause, between the Heliosphere and the LISM. The Heliosphere is therefore the region of space directly affected by the Sun, including the Earth and the rest of the Solar System. The space environment throughout the Heliosphere comprises plasma from the Sun as well as from inner Heliosphere sources such as planets, comets, and dust, and from the LISM. While ions from the LISM are generally deflected around the Heliosphere due to the IMF, with the exception of highly energetic particles, namely galactic cosmic rays, the relative motion of the LISM and Heliosphere causes a constant stream of neutral particle inflow across the Heliopause, unaffected by the IMF. This inflow of neutral particles into the Heliosphere affects the plasma composition and dynamics of the entire space environment. The longitudinal inflow direction, λ_{ISN}, of the interstellar wind through the Heliosphere is an important feature of the Heliosphere — interstellar interactions that can be measured using various techniques.

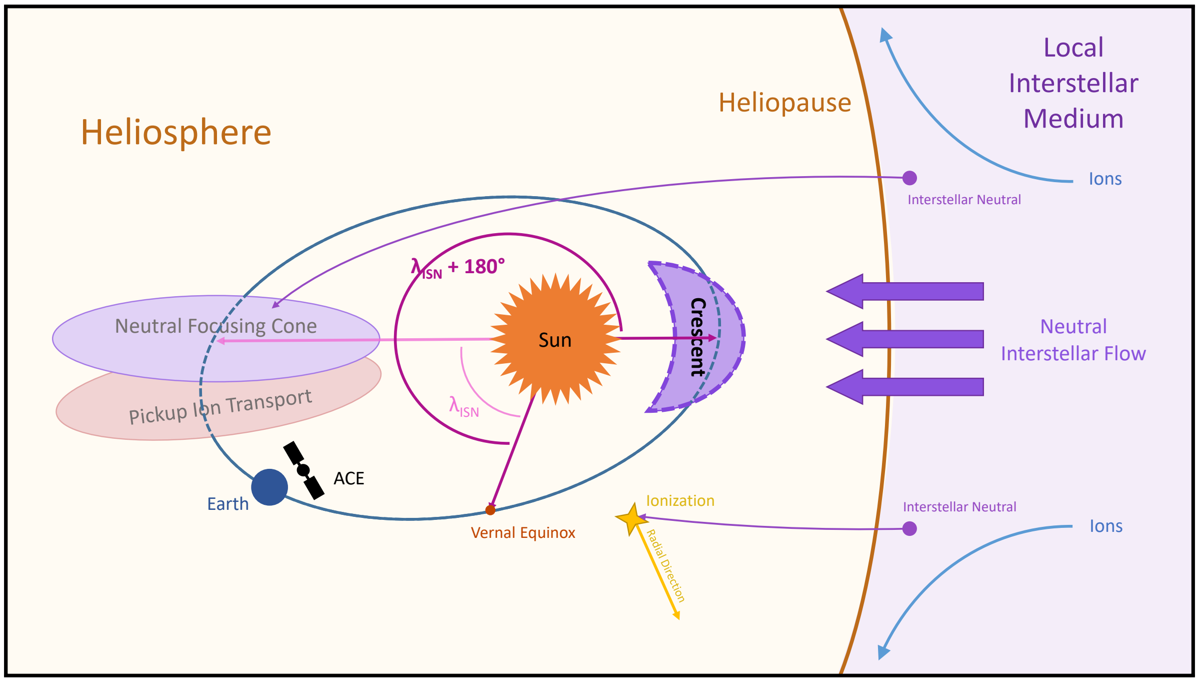


Figure 4.1: Diagram of Heliosphere – Interstellar interactions highlighting the crescent on the upwind side of the Sun.

The inflow direction, λ_{ISN}, can be measured using *in situ* data out to 1 AU (Möbius et al. (1985); Moebius et al. (1995)), as discussed in Chapter 3. While the flow direction was originally approximated using ultraviolet backscatter techniques by Bertaux & Blamont (1971); Thomas & Krassa (1971); Weller & Meier (1974), more recent studies have used *in situ* observations of neutrals and interstellar pickup ions (PUIs) to constrain the flow direction, as discussed in Chapter 3. While the PUI methods have begun to converge on comparable results, as reported in Chapter 3, there are still outstanding challenges in reporting an exact value, especially using a limited data set to find a trend over time, both from analysis constraints and from current measurement limitations.

In this chapter, we discuss ongoing analysis to further quantify the flow direction and the evolution over time of the Heliosphere – interstellar interactions as well as the current challenges and requirements for improved measurements and future studies. We discuss in detail ongoing efforts to further characterize the evolution of the Heliosphere — interstellar interactions by measuring the evolution of the longitudinal flow direction, λ_{ISN}, using the direct PUI cutoff method in the helium crescent over time.

4.2.2 The Upwind Crescent

While neutral interstellar wind atoms with a higher first ionization potential (FIP) may be focused into the helium focusing cone on the downwind side of the Sun, as discussed in Chapter 3, neutrals are also enhanced in the crescent on the upwind side of the Sun, as shown in Figure 4.1. The crescent is formed upwind due to the survival of in-flowing neutrals closer to the Heliopause and the higher energy PUI cutoff found in the upwind direction due to the anti-parallel velocities of the in-flowing neutrals and the radial solar wind (Sokół et al. (2016)). As the crescent is closer to the Heliopause, even atoms with a lower FIP may be found in the crescent before becoming ionized through processes such as photoionization, electron impact ionization, and charge exchange with the solar wind. Similarly to the focusing cone, the orientation of the crescent is directly correlated to the direction of the interstellar neutral inflow direction, λ_{ISN}, where the center of the crescent corresponds to an angle $\lambda_{ISN} + 180°$.

Helium, with a relatively high FIP, may be found in both the downwind cone and the upwind crescent. As a result, measurements of He^+ PUIs that were recently injected within each of these regions can be compared using different methods to determine the longitudinal flow direction, λ_{ISN}. The focusing cone method, as implemented in Chapter 3, yields an overall flow direction of 75.37° ±0.43°, consistent with the literature, and reveals the lack of a shifting trend in the flow direction over an 11-year solar cycle. However, the individual 3-orbit yearly flow directions obtained using this method exhibit fluctuations likely influenced by solar activity. While this affect does not appear to be solar cycle dependent, and while still revealing the trend over the 11-year solar cycle, it impacts our ability to report an accurate yearly flow direction. This chapter, therefore, discusses our ongoing efforts to track the flow direction using a signature of the upwind crescent in He^+ measurements by applying the direct PUI cutoff method, which shows promise in providing a more accurate yearly flow direction.

As neutrals become ionized and are injected into the solar wind distribution as PUIs, they begin to gyrate about the magnetic field with speeds up to two times the solar wind speed. Therefore, the distribution of ions, which can be viewed as counts or flux as a function of energy or velocity, comprises a solar wind peak around the bulk solar wind speed and an extended region out to up to two times the solar wind speed, characterized by PUIs, as well as a suprathermal tail of ions accelerated by other processes, such as shocks. Due to the relative velocities of the in-flowing neutrals to the solar wind, the cutoff, representing the maximum injection speed of newly ionized PUIs, will be found at a maximum velocity within the distribution in the upwind direction where the flow direction is anti-parallel to the radial solar wind and at a minimum velocity downwind, where the flow has been focused into a more tangential relative velocity. Therefore, by tracking the cutoff location in the distribution over ecliptic longitude, it is possible to locate the maximum cutoff location, corresponding to the upwind flow direction.

As discussed by Möbius et al. (2015a, 2016a,b), utilizing the direct PUI cutoff method to locate the point on the He^+ count distribution reflecting the injection of freshly ionized PUIs in w', where $w' = v/v_{sw} - 1$, and v_{sw} is the solar wind velocity, over 1° bins in ecliptic longitude over complete orbits around the Sun, enables the detection of the upwind crescent. Direclty utilizing the cutoff allows for more accurate measurements using the PUI dataset, as it mitigates issues such as PUI transport, as discussed in Chapter 3 and later in this chapter in Section 4.3.1. As demonstrated further in this chapter, the cutoff method in the upwind crescent shows promise in providing more accurate yearly flow direction measurements. However, the preliminary results appear to show a solar cycle dependence, and future work will be required to complete a full analysis of this data set using this method.

4.3 Background

4.3.1 Pickup Ion Transport

One major consideration that must be accounted for when measuring properties of the neutral interstellar flow using interstellar source neutrals which have become ionized as PUIs through photoionization, electron impact ionization, and charge exchange with the solar wind is the effect of PUI transport. PUI transport occurs when, due to their charge and interaction with the solar magnetic field, the PUIs begin to drift along the solar IMF, as reported by Möbius et al. (1996), Chalov & Fahr (1999), Chalov (2014), and Quinn et al. (2016).

As neutrals, the interstellar atoms are unaffected by the IMF, and therefore may be traced directly to the interstellar flow velocity, as their speed and direction are affected only by the gravitation of the Sun. However, as they become ionized, PUIs are injected in a torus distribution around the magnetic field in velocity space, but over time they begin to isotropize and into a shell distribution due to scattering from turbulence in the flow (Drews et al. (2015); Quinn et al. (2016)). Therefore, over time, the PUIs retain fewer signatures of their original properties as interstellar neutrals as they are accelerated, thermalize over time, and begin to propagate radially outward from the Sun with the solar wind.

Due to PUI transport, if proper considerations are not made to select only for measurements of newly injected PUIs which have not yet undergone significant transport, the center of the flux of focused particles will appear shifted in the ion data from that of the neutrals, as shown in the diagram for the focusing cone in Figure 4.1.

Studies using PUIs which report a flow direction, λ_{ISN}, far from the literature average, such as Drews et al. (2012) and Gershman et al. (2013) are likely influenced by effects of PUI transport. Therefore, in order to use PUIs to study properties of the neutral interstellar flow, it is necessary to study more pristine, recently injected, un-transported PUIs (Quinn et al. (2016)). We reported in Chapter 3 on a number of considerations for limiting the measurements and distribution to the region of newly injected PUIs, namely selecting for perpendicular conditions between the solar wind velocity and solar IMF vectors and for accounting for the shift in cutoff location as a function of ecliptic longitude in the PUI region within the distribution, as reported by Möbius et al. (1999), as well as neglecting regions dominated by the solar wind, suprathermals, and inner source ions, by locating the PUI cutoff in the w' range of $0.7 \leq w' \leq 1.5$ (Gloeckler & Geiss (1998); Möbius et al. (2015a)).

4.3.2 Methods for Measuring the Flow Direction Using PUI Data

In Chapter 3, we reference a number of studies from the literature which measure the flow direction, λ_{ISN}, using a variety of methods, including locating the downwind helium focusing cone longitude using flux measurements of He^+ PUIs or by determining the upwind crescent orientation using the PUI cutoff, and utilizing data sets spanning different ranges of years and originating from various missions such as ACE/SWICS, STEREO/PLASTIC, Nozomi, and AMPTE. While Gloeckler et al. (2004) account for some PUI considerations by taking large averages over the data set and accounting for inner source PUIs and ionization rates using a modeling technique, this method does not lend itself to determining an accurate flow direction across individual years to determine a trend. As previously noted, other studies using signatures of the focusing cone, such as Drews et al. (2012) and Gershman et al. (2013), are likely impacted by PUI transport effects.

Studies such as Möbius et al. (2015a, 2016a,b), Taut et al. (2018), and Bower et al. (2019) have shown that using the PUI cutoff method to locate the crescent can be applied across larger data sets to measure the flow direction, and in Chapter 3, we successfully utilize a method combining aspects of the cutoff method and measurement of the focusing cone orientation to both determine a flow direction of 75.37° $\pm$0.43° for the aggregate data set as well as to measure an absence of trend variation in the direction to a reported statistical fit error of 0° $\pm$0.51° / yr. We note, however, that our measurements are still impacted by solar source effects such as compression regions in the solar wind. We therefore discuss herein ongoing efforts to reduce such effects on the PUI cutoff method to determine a flow direction over time using the ACE/SWICS He^+ double-coincidence data set spanning from 1998 through 2011, as well as ongoing challenges and requirements for improved measurements.

4.3.3 Data Sources

In our ongoing efforts to characterize the evolution of the Heliosphere – interstellar interactions over time using the PUI cutoff method, we obtain data from the sources reported in Chapter 3, Section 3.4.2.

4.4 Measuring a Yearly Flow Direction

The following procedure describes our methodology for measuring yearly and combined flow directions, λ_{ISN}, from the orientation of the upwind crescent. We use ACE/SWICS He^+ double-coincidence measurements spanning from 1998 through 2011, with focusing cone crossings occurring in the years 1998 through 2010, and corresponding crescent crossings occurring in the years 1999 through 2011.

For consistency with our measurements in Chapter 3, we label orbits by the year during which the focusing cone crossing occurs, though it may be noted that the corresponding crescent crossing occurs in the next calendar year. Additionally, the cone crossing corresponds to the time when the spacecraft reaches the ecliptic longitude directly representing the flow direction, as opposed to the 180° shifted upwind direction measured in the crescent.

For this method, we first bin ACE/SWICS He^+ count distributions at the native 12-minute resolution. We then compute the corresponding w' bins at the 12-minute resolution and filter out time periods correspond with ICMEs at the ACE spacecraft according to the Richardson & Cane database (Cane & Richardson (2003); Richardson & Cane (2010)) and for time periods corresponding to relatively perpendicular (90° $\pm$20°) solar wind velocity to IMF vector conditions. We remove time periods associated with ICMEs, as these transients cause large flux enhancements and complicated magnetic fields that bias our measurements, and we select only measurements taken at times of relatively perpendicular solar wind velocity to IMF vector conditions to ensure the PUI torus is in the instrument's field of view, as described in Möbius et al. (2015a) and Bower et al. (2019), due to the findings reported by Chalov & Fahr (1998) and Drews et al. (2015). We then accumulate 1° count distributions, limit the range in w' to $0.7 \leq w' \leq 1.5$ to ensure we are analyzing the region of the distribution representing interstellar PUIs, and locate the PUI cutoff at half the maximum of the smoothed distribution. We analyze the cutoff values in w' as a function of ecliptic longitude, fitted to a function reported by Möbius et al. (2015a) to determine the upwind flow direction, $\lambda_{ISN} + 180^\circ$ for each 3-orbit boxcar average of normalized combined count distributions. We shift these values by 180° to determine the yearly flow direction, λ_{ISN}, and look for trend information over an 11-year solar cycle. Additionally, we repeat the method for the combined data set to determine a unified flow direction, λ_{ISN}.

We present here steps taken to measure a representative yearly flow direction from ACE/SWICS He^+ PUI data by determining the ecliptic longitudinal orientation of the upwind crescent for the 11-year solar cycle spanning the years 1999 through 2009.

4.4.1 Binning Native 12-minute Resolution ACE/SWICS He^+ Double-Coincidence Count Distributions

Similarly to the methodology presented in Chapter 3, we bin ACE/SWICS He^+ count distributions at the native 12-minute instrument resolution integrated over all instrument sectors. To account for solar transients, we remove times at the 12-minute resolution associated with ICMEs at the ACE spacecraft (Cane & Richardson (2003); Richardson & Cane (2010)). In order to ensure the PUI torus is in the field of view to minimize PUI transport effects due to findings reported by Chalov & Fahr (1998) and Drews et al. (2015), we additionally select only measurements made at times when the solar wind velocity and solar IMF vectors are approximately perpendicular, or at 90° $\pm 20^\circ$, as described by Möbius et al. (2015a) and Bower et al. (2019). We additionally calculate w' bins values for each 12-minute time step according to Equation 3.2 and the steps reported in Chapter 3. Additionally, as ACE is located at Earth's Lagrange 1 point, we map each measurement time to Earth ecliptic longitude.

4.4.2 Combining 3-orbit Boxcar Window 1° Normalized Averaged Count Distributions

We accumulate count distribution over 1° in ecliptic longitude for all 360° in each orbit as indicated by Möbius et al. (2015a) and Bower et al. (2019). To further account for solar cycle and transient effects, as in Chapter 3, we utilize a 3-orbit boxcar to combine and average our 1° count distributions similarly to the boxcar method implemented by Drews et al. (2012). We additionally normalize the distributions as in Bower et al. (2019) for comparison over a full 360° orbit. This normalization does not affect the shape of the fitted crescent signature as it would have the cone signature in Chapter 3, as the w' axis is unaffected by the normalization, unlike the count axis. This normalization helps to visualize the distributions at each ecliptic longitude over a full orbit.

4.4.3 Locating the PUI Cutoff in the Count Distribution

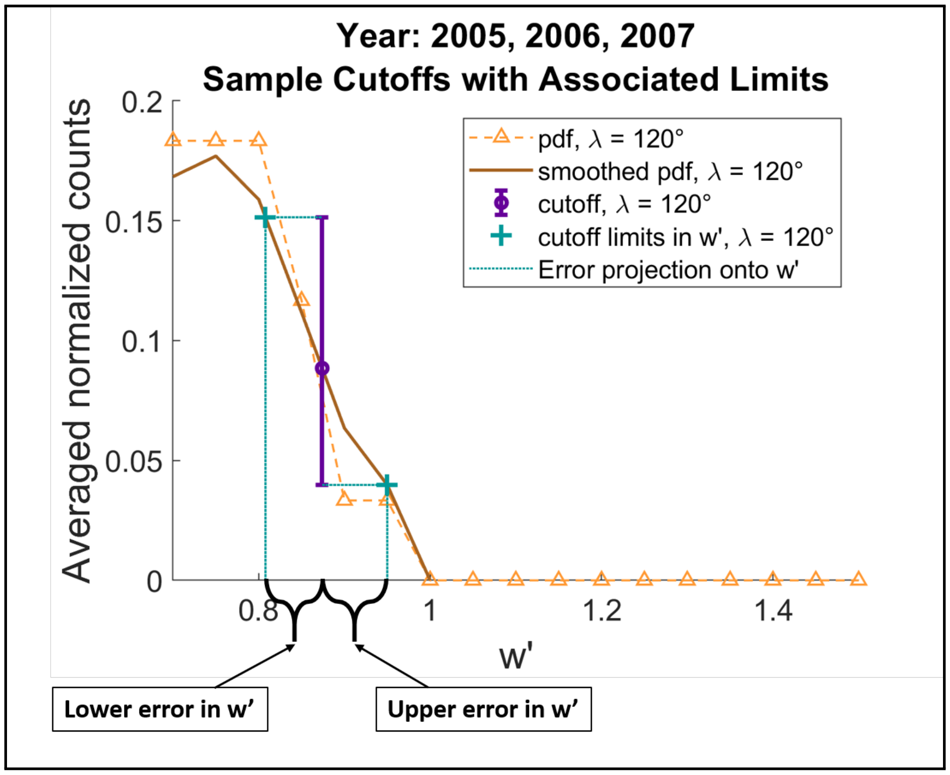


Figure 4.2: 3-orbit boxcar averaged count distribution centered on the cone crossing year occurring in 2006 (with corresponding crescent crossing in 2007) plotted against w' in the range $0.7 \leq w' \leq 1.5$ for ecliptic longitude $\lambda = 120°$ (orange squares with dashed line) along with the corresponding smoothed distribution (solid line), the distribution value at the cutoff with statistical errors (teal circle with error bars), and the errors projected as limits in w' (purple plus sign with dotted lines).

We smooth each 3-orbit boxcar combined 1° normalized count distribution with a 5-point window Gaussian smoothing algorithm, as in Chapter 3, and locate the cutoff as the linearly interpolated half-maximum on the smoothed distribution in the range $0.7 \leq w' \leq 1.5$, as described by Bower et al. (2019).

We calculate errors from the statistical Poisson measurement error of $\varepsilon_{dist} = \sqrt{n}$, where n is the number of counts, as reported in detail in Chapter 3, Section 3.7.1. We convert the error in counts to an error in w' by projecting the upper and lower statistical count errors onto the smoothed distribution and linearly interpolating to find the associated w' values, as shown in Figure 4.2 for $\lambda = 120°$ for the 3-orbit boxcar center year with cone crossing occurring in 2006. The error in w' is then taken as the absolute differences between these values and the w' cutoff value, as demonstrated in Figure 4.2.

Sample distributions with associated cutoffs and limits are shown in Figure 4.3 for the 3-orbit center year having a focusing cone crossing in 2006 (with corresponding crescent crossing in 2007) for ecliptic longitudes of $\lambda = 120°$, 210°, and 300°. The cutoffs can be shown to shift in w' with longitude due to injection velocity of the PUIs relative to the radial solar wind, as described by Möbius et al. (1999).

4.4.4 Measuring the Flow Direction from a Fit to the Upwind Crescent

We fit the cutoffs in w' as a function of ecliptic longitude with weights of one over the squared maximum errors using the functional fit adapted from Möbius et al. (2015a) and Bower et al. (2019) given as:

$$v_{\infty_1} = \frac{v_\infty}{v_E} \tag{4.1}$$

$$\lambda_0 = \arccos\left(-(1 + v_\infty^2)^{-1}\right) \tag{4.2}$$

$$\lambda = \lambda_{bin} - \lambda_\infty \tag{4.3}$$

$$\begin{aligned} v_{r+}^2 = 2 + v_{\infty_1}^2 - (1 - \cos\lambda) - \{v_{\infty_1}^2 \sin^2\lambda + \\ v_{\infty_1} \sin|\lambda|[v_{\infty_1}^2 \sin^2\lambda + 4(1 - \cos\lambda)]^{1/2}\}/2 \end{aligned} \tag{4.4}$$

$$v_{r-} = -v_{r+} \tag{4.5}$$

$$fit = -v_E v_r + offset \tag{4.6}$$

where λ_∞ is the fitted parameter for the crescent center, which corresponds to $\lambda_{ISN} + 180°$, v_∞ is the fitted parameter for v_{ISN}, v_E is the velocity of the Earth, λ_{bin} is the ecliptic longitude of a given bin, and $offset$ is the functional form vertical offset along the w' axis.

In Figure 4.4, we visualize 2D histograms of all 1° normalized combined averaged count dis-

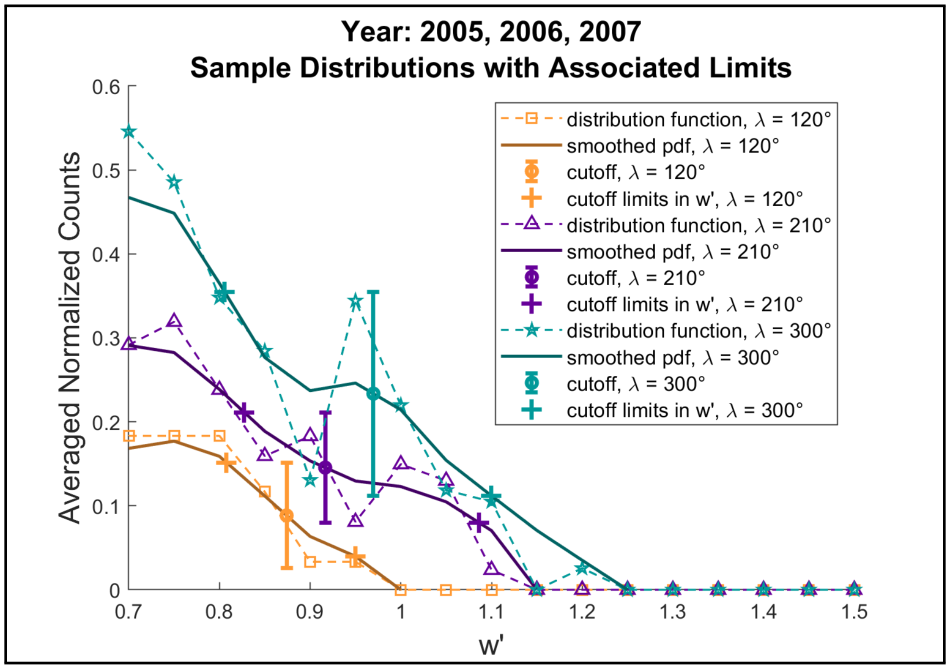


Figure 4.3: 3-orbit boxcar averaged count distribution centered on the cone crossing occurring in 2006 (with corresponding crescent crossing in 2007) plotted against w' in the range $0.7 \leq w' \leq 1.5$ for ecliptic longitudes $\lambda = 120°$ (orange squares), $210°$ (purple triangles), and $300°$ (teal stars) along with the corresponding smoothed distributions, the distribution value at the cutoff with statistical errors, and the errors projected as limits in w' in the corresponding color. The cutoff point can be seen to shift with ecliptic longitude.

tributions with w' cutoff values plotted as black circles with associated limits against longitude for the 3-orbit boxcar centered on cone crossing year 2006 as well as for all combined years, along with the crescent fit used to infer the flow direction, λ_{ISM}.

The preliminary results for a yearly flow direction are reported in Table 4.1. These values still appear to be influenced by solar cycle, as evidenced by the significantly lower longitudes measured only around solar maximum, with the lowest value occurring in 2001, at the solar maximum for Solar Cycle 23, as seen in Figure 4.5. It may be noted that the focusing cone method, reported in Chapter 3, relies on flux enhancements downwind, and while these measurements do exhibit fluctuations due to solar activity, these affects are largely stochastically distributed and accounted for using multiple year averaging. However, the direct cutoff method relies on the signature of the upwind crescent in the PUI cutoff location over ecliptic longitude, which is a direct effect of the relative flow velocities of the radial solar wind and the incoming interstellar neutral flow. Due to the measurement of the flow signature directly from the relative flow velocities, this method appears to be more impacted by the solar cycle.

In the direct PUI cutoff method, there specifically appears to be an effect of compression re-

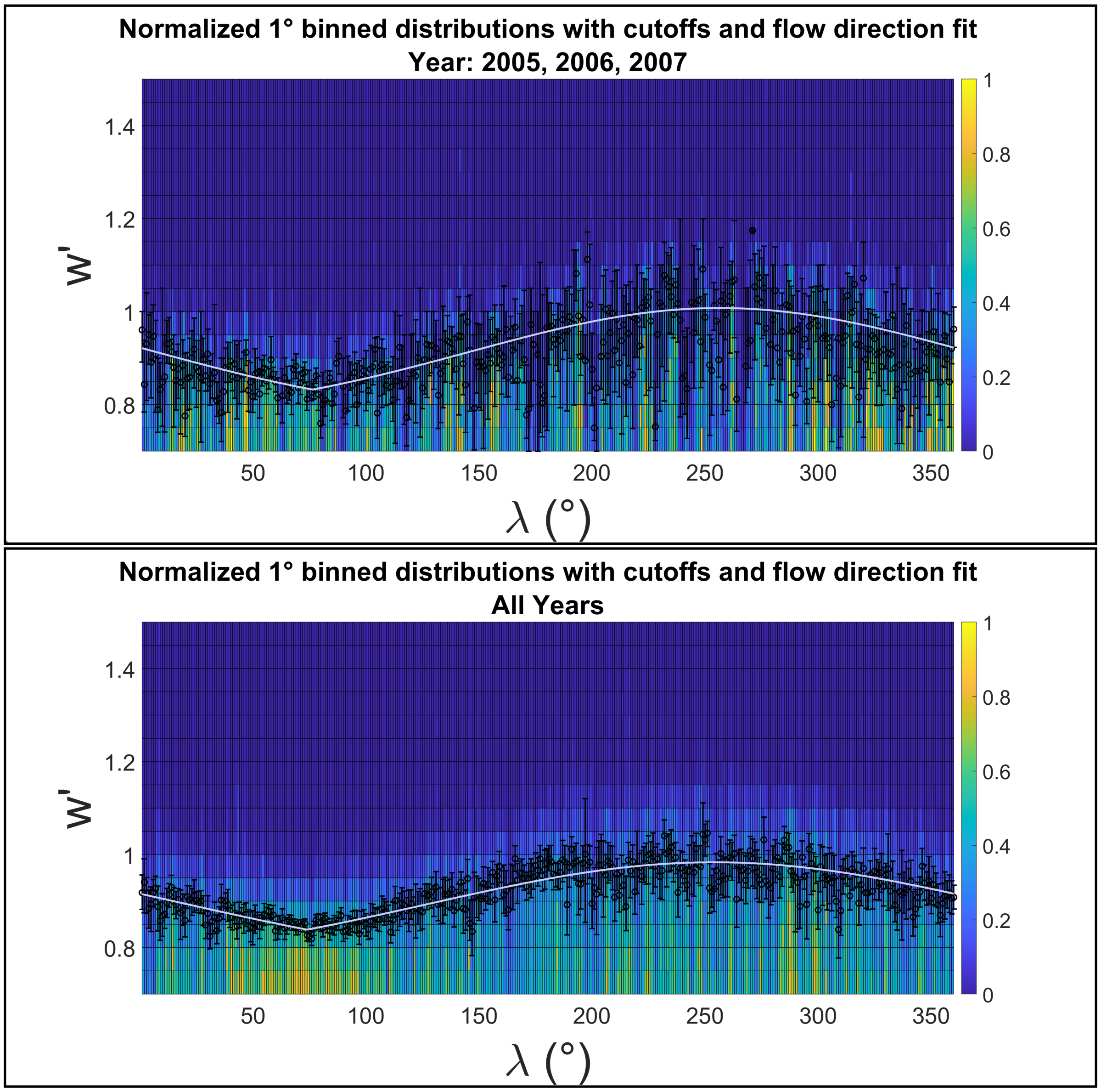


Figure 4.4: Cutoff locations determined from the upwind crescent. Limits in w' plotted as black circles over a 2D histogram of normalized averaged count distributions per 1° bin in ecliptic longitude for the 3-orbit boxcar centered on the focusing cone crossing occurring in 2006 as well as for all combined orbits. Averaged count distributions are normalized per 1° bin here for visualization purposes only to show the cutoff location with respect to each distribution.

Table 4.1: Preliminary λ_{ISN} measurements by orbit determined from the upwind crescent.

Year	λ_{ISN}	Variance	Fit Value (R^2)
1999	68.72°	±4.36°	0.66
2000	65.16°	±0.95°	0.99
2001	67.01°	±1.21°	0.98
2002	73.20°	±1.04°	0.98
2003	76.00°	±1.58°	0.97
2004	74.54°	±0.97°	0.98
2005	74.39°	±1.05°	0.98
2006	76.40°	±1.68°	1.00
2007	73.87°	±0.65°	0.99
2008	76.82°	±0.56°	0.99
2009	71.46°	±0.62°	0.99
Combined Data Set	73.74°	±2.25°	0.84

gions on the years around solar maximum, while the measurements seem to begin to converge in later years, even more so than when using the focusing cone method reported in Chapter 3. Potential compression region effects are highlighted for the year 1999 in Figure 4.6. As noted earlier, utilizing the PUI cutoff may significantly reduce effects such as transport on the measurements, as discussed by Möbius et al. (2015a, 2016a).

If these measurements can be decoupled from this apparent solar cycle dependence, this method may yield accurate yearly flow directions in addition to trend information over a complete 11-year solar cycle. This effect may be achieved by reducing the impact of compression regions on the data set. It is likely that completely removing times associated with compression regions may remove the necessary signatures for fitting the flow direction. However, it may be possible to determine a signature in the data that allows us to compensate for these flux enhancements. If it is possible to statistically determine the effect of the compression regions on the measurements, we may be able to apply a correction to reduce the effect, thereby decoupling our results from this dependence.

4.4.5 Measuring a Unified Flow Direction from the Complete Data Set

As in Chapter 3, we combine the full data set of 13 complete orbits with focusing cone crossings occurring in the years 1998 through 2010 (with corresponding crescent crossings occurring in the years 1999 through 2011) and repeat the process to determine a single flow direction by applying the crescent fit to the w' cutoff values found at each 1° in ecliptic longitude.

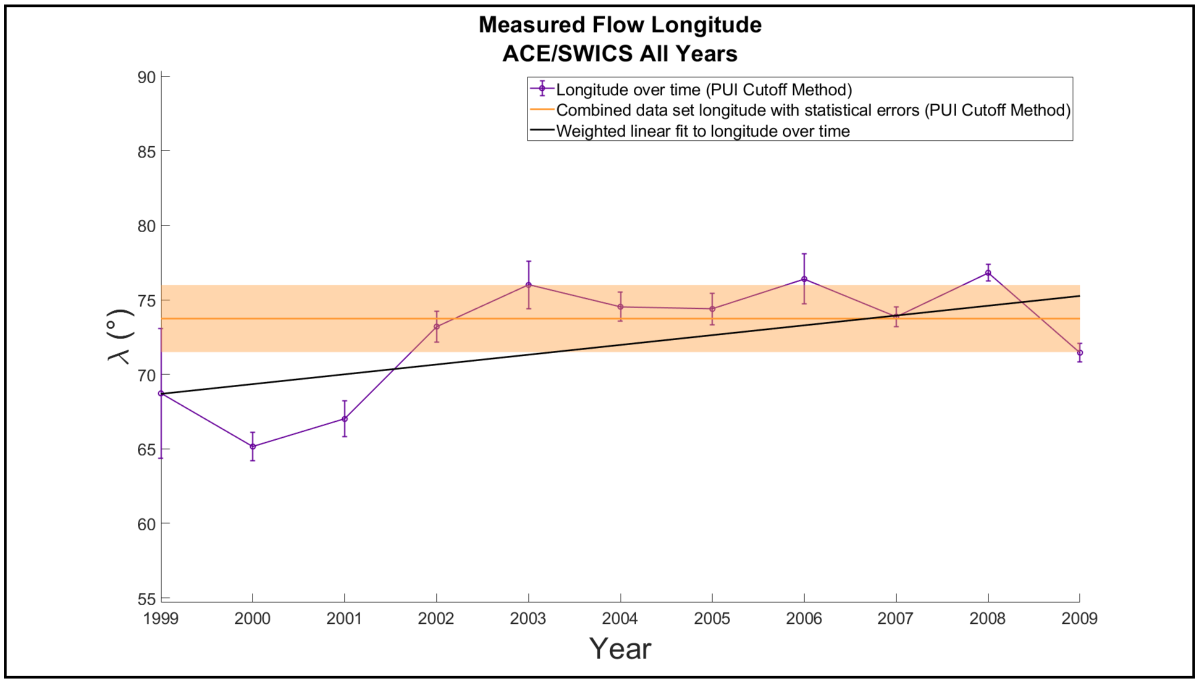


Figure 4.5: Preliminary λ_{ISN} values determined from the upwind crescent are plotted against the corresponding 3-orbit center year with statistical fit errors as dark purple circles. The combined data set flow direction, λ_{ISN}, is plotted with statistical fit error in dark orange, and the linear fit of the individual yearly λ_{ISN} values using weights derived from the statistical fit errors is plotted as a black line. The best fit line yields a slope of 0.66° ±0.71° and the preliminary combined flow directions yields a value of 73.74° ±2.25°. This fit is currently heavily influenced by the low longitudes measured in the solar maximum years influenced by compression regions in the helium focusing cone. A linear fit weighted by error to the values in the years 2002 through 2009 (corresponding to crescent crossings in 2003 through 2010) yields a slope of -0.11° ±0.90°, and a weighted average of the preliminary λ_{ISN} values in these years yields a value of 74.89° ±1.17°.

4.4.6 Preliminary Results

Determining a trend from the flow directions Over time As in Chapter 3, we fit the calculated 3-orbit boxcar flow directions to a line, using weights of the one over the squared error calculated by the crescent fitting algorithm. We plot our preliminary results for the individual flow directions with reported errors along with the fit line together in Figure 4.5.

These preliminary results can be seen to have a significant bias to lower ecliptic longitudes during solar maximum, indicating that unlike in the focusing cone method, we have not averaged out the effects of solar activity on our results. While these data result in a best-fit line with a slope of 0.66° ±0.71°, this value appears to have a strong bias, and the results are still inconclusive, as we discuss further in Section 4.5

Measuring a unified flow direction from the aggregate data set Using the method described previously to combine the full data set, as visualized in Figure 4.4, we measure a preliminary flow direction of 73.74° ±2.25°. However, this result is again biased by the solar cycle effects seen to significantly impact years around solar maximum, as we discuss further in Section 4.5.

We take a weighted average and slope of the linear fit weighted by error for non-solar maximum individual flow direction results for center years with cone crossings occurring from 2002 through 2009 (corresponding to crescent crossings in 2003 through 2010). This results in a unified average flow direction of 74.89° ±1.17° and a slope of -0.11° ±0.90°. This shows that we are likely to achieve reasonable results for the complete data set if we are able to decouple our analysis from the solar cycle dependence.

4.5 Discussion

The use of PUIs to measure parameters of the Heliosphere — interstellar interactions presents a number of challenges ranging from analysis techniques to measurement capabilities. We first discuss these challenges with respect to our preliminary implementation of the PUI cutoff method on the ACE/SWICS He^+ double-coincidence data set and then expand upon what is needed to improve future studies and measurements.

4.5.1 Challenges in Interstellar Flow Measurements Using Pickup Ions

In Section 4.4.6, we discuss preliminary results obtained by applying the PUI cutoff method to map the orientation of the upwind crescent to the longitudinal inflow direction, λ_{ISN}, of the interstellar wind through the Heliosphere. We note that our results are still significantly impacted by solar cycle and infer that there is likely an impact on our results by factors such as compression regions, which can cause seeming enhancements in the especially dense regions of He^+ being analyzed in this study (Möbius et al. (2010)). In Figure 4.6, we show a comparison of the upwind crescent fit to the cutoff values for the 3-orbit boxcar center year with cone crossing in 2004, which has a fit consistent with reported literature values of $\lambda_{ISN} \approx 75°$, as compared with the fit for the 3-orbit boxcar center year with cone crossing in 1999, having a much lower measured longitude of $\lambda_{ISN} \approx 69°$, potentially due to compression region enhancements.

While our results for years outside of the affected solar maximum years show promising results when compared with our previous study using the combined cutoff-cone method and to recent published literature values, we must perform further analysis to determine if these effects can be filtered out while maintaining statistics for measuring the yearly flow direction. These compression regions present a significant challenge by masking the signatures of the the cone and crescent regions in the PUI data, while directly removing them may result in restricting the data set to measurements that no longer contain the necessary features for fitting these regions to determine a flow direction. To overcome this challenge, we must improve our methodology to filter out the effects of these compression regions while maintaining the necessary features in the data set.

4.5.2 Impacts of Data Statistics on PUI Measurements

Many literature studies measure the overall flow direction, λ_{ISN}, both in the cone, such as Gloeckler et al. (2004), Drews et al. (2012) and Gershman et al. (2013) and in the crescent, such as Drews et al. (2012), Möbius et al. (2015a, 2016a,b), Taut et al. (2018), and Bower et al. (2019), using data sets of many combined years. Gloeckler et al. (2004) additionally use very large smoothing techniques of up to 9 and 15 day smoothing windows corresponding to nearly 8.9° and 14.8°, which provides challenges when attempting to determine a flow direction to less than a degree. Drews et al. (2012) employ a boxcar method of up to 4 orbits for combining data from multiple years. This method can help to significantly improve data statistics and increase precision for a trend over time, but reduces the certainty on the individual yearly flow direction measurements. Additionally, increasing the moving boxcar window for trend information over time will require an expanded data set over a greater number of orbits to rule out effects of solar cycle while still allowing the number of orbits to dominate the boxcar window size.

When measuring a yearly flow direction to observe trend behavior over time, it is necessary to have sufficient data statistics for very limited subsets of the full data set. As discussed earlier, discarding all compression region times, for example, may have serious impacts on that data set that could deplete or completely remove signatures of the focusing cone and crescent. Using the currently reported methods, when restricting measurements to perpendicular solar wind velocity to IMF conditions, it is necessary to discard a significant number of measurements, and a Sun-pointed spinner like ACE/SWICS does not provide ideal conditions for ensuring the highest likelihood of achieving the necessary perpendicular conditions in every scan. While other studies such as Drews et al. (2012) and Gershman et al. (2013) did not restrict the data to these conditions, PUI transport was likely a contributing factor to the determination of a higher longitude flow direction, λ_{ISN}.

For the ACE/SWICS data set from 1998 through 2011, less than 32% of He$^+$ double-coincidence measurements are found to be made under acceptable perpendicular solar wind velocity – IMF conditions, and individual years have complete 1° resolution binned data gaps of up to 41° out of 360° due to restrictions on perpendicular conditions, removal of ICME times, and instrument-level data gaps, which has a large impact on any fitting algorithm. If we further desire to reduce the data to select only the Sun sector, we would only be able to take approximately one eighth of the remaining data per scan.

These limitations introduce challenges in retaining enough data to have sufficient statistics to observe the necessary features of the focusing cone and crescent. Therefore, we must develop new analysis techniques, though we are very limited by the amount of data that we have, limit the considerations we make on the data set, or make improvements to our measurement techniques that will allow us to make future observations that are more suited for studies using the instrumental, yet currently limited PUI measurements.

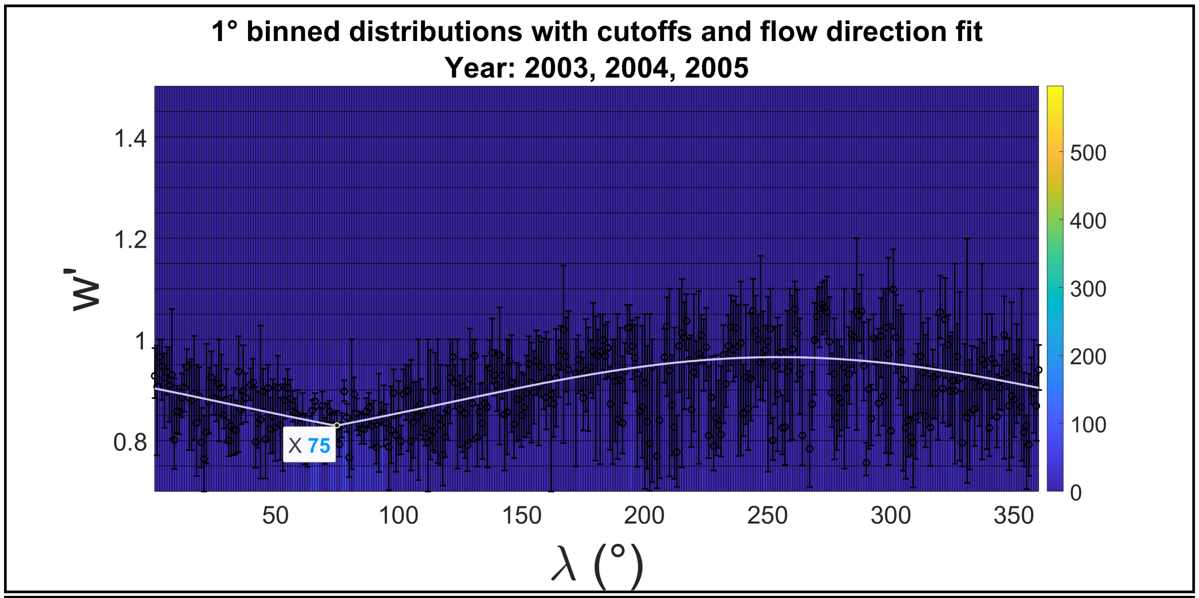
4.5.3 Requirements for Future Measurements

The Heliosphere is heavily influenced by the LISM, which provides a steady inflow of neutral particles. While measurements of PUIs, which represent a large portion of the plasma that was most recently interstellar neutrals, are an extremely useful tool in sampling the interstellar component of the plasma in the Heliosphere *in situ*, they are limited by current measurement techniques. PUIs both are necessarily characterized by a lower charge state, making them harder to detect, and represent a much lower flux within the distribution than solar source ions, making them more difficult to measure, especially when measuring in the solar radial look direction.

Since PUIs necessarily have a low charge-state, they are very insignificantly affected by the post-acceleration in current ion mass spectrometers, whose acceleration effects are limited by the voltage and the ion's charge-state. Therefore, instruments intended to measure PUIs should implement more sensitive energy detection techniques, including more sensitive energy detectors and greater post-acceleration capabilities with higher post-acceleration voltages (Gilbert et al. (2016)).

It would be beneficial to detecting and distinguishing PUIs embedded in the solar wind to achieve higher time resolutions and greater geometric factors, or an effective sampling area of the instrument with considerations for both mechanical and electrostatic effects. Additionally, ACE/SWICS is limited in its capacity as a Sun-pointed spinner to detect PUIs, as the much lower flux PUI measurements are largely dominated by the solar wind. Therefore, having an instrument pointing away from the radial direction, such as in the direction of the incoming interstellar flow, could improve measurements intended for non-solar source plasma. A more ideal spacecraft orientation for these measurements would be more similar to the Wind spacecraft, having an azimuthal field of view orthogonal to the spin axis, optimized for sampling in all directions in the Heliospheric ecliptic plane, with a large, $\pm 30°$, elevation field of view out of the ecliptic.

Lastly, future missions intending to sample *in situ* plasma in different regions of space, such as an Interstellar Probe and other future missions, including potentially numerous smaller spacecraft, could carry instruments intended for measuring PUIs with more sensitive energy detectors, higher post-acceleration voltages, higher cadence measurements, and larger geometric factors to sample the distributions in a variety of locations and look directions.



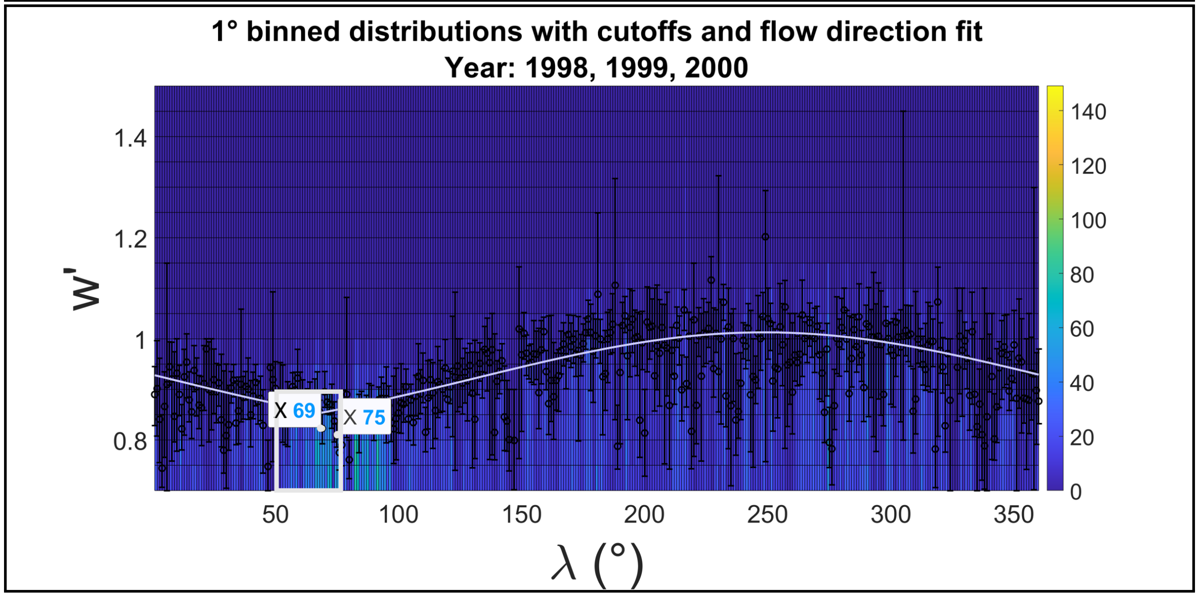


Figure 4.6: 2D histogram of averaged combined count distributions along with measured cutoff locations plotted as black circles with associated limits and cutoff fit for the 3-orbit boxcar center years with cone crossings occurring in 2004 and 1999. Approximate longitudes are marked for the measured flow direction (∼75° in 2004 and ∼69° in 1999) and the average literature flow direction (∼75°). A white box indicates a higher flux region centered on a longitude to the left of the expected flow direction in 1999, which is likely affecting the cutoff fit and associated measured longitude. Distributions are not normalized here in order to demonstrate the enhancement in averaged combined counts.

Though we bin He$^+$ count distributions into 1° bins in ecliptic longitude for our study of the longitudinal interstellar flow direction, it is necessary to determine perpendicular solar wind velocity to IMF vector conditions at a higher resolution to select measurements of un-transported newly injected PUIs. For reporting an angle, λ_{ISN}, at a precision of 0.01°, it would be beneficial to make these considerations at a higher resolution. Therefore, scans on the order of 0.001°, corresponding to $\sim$1.5 min at Earth's orbit, would improve our ability to filter and select more precise measurements. To achieve count rates at a 1.5 min resolution similar to those from SWICS at the 12 min resolution, we can approximate a desired geometric factor from the simplified equation derived from Wüest et al. (2007) as:

$$G \approx C/(J\,\Delta E\,\Delta t) \tag{4.7}$$

where G is the geometric factor, C is counts, J is flux, ΔE is energy resolution, and Δt is time resolution. Taking a ratio of the desired geometric factor (subscript d) to the SWICS geometric factor (subscript S), and solving for the desired geometric factor given the same counts, flux, and energy resolution, we get:

$$\begin{aligned} G_d/G_s &\approx \frac{C/(J\,\Delta E\,\Delta t_d)}{C/(J\,\Delta E\,\Delta t_S)} \\ &\Rightarrow G_d \approx G_s(\Delta t_s/\Delta t_d) \end{aligned} \tag{4.8}$$

Given a SWICS geometric factor of 2×10^{-3} cm^2 sr eV/eV and a SWICS measurement time resolution of 12 min (Gloeckler et al. (1998)), then a geometric factor for a similar instrument with a 1.5 min time resolution would be approximately 1.6×10^{-2} cm^2 sr eV/eV. In order to get comparable counts for the entire 1°, approximated as 1440 min (1 day), distribution at a 1.5 min resolution, a geometric factor of approximately 1.9 cm^2 sr eV/eV would be required. Such a high geometric factor would likely come at the cost of energy resolution, and the necessary ESA channels would be too large for useful measurements if all other variables were held constant. However, it may still be possible to increase geometric factor significantly by increasing the number of channels, such as proposed for the previously mentioned SPICES future instrument.

Future instruments with improvements such as those listed above will greatly increase the available data set for studying Heliosphere — interstellar interactions. One such example of a future instrument is SPICES, a NASA-funded technology development project based on the PICSPEC design (Gilbert et al. (2016)), which will have a -50 kV power supply for the post-acceleration voltage, specifically targeted at detecting low-charge-state PUIs. Some of the required measurement improvements will also be achieved from instruments such as SO-HIS, the Heavy Ion Sensor launched on board Solar Orbiter on 9 February, 2020. Measurements provided by SO-HIS that will improve our analysis in the future include higher time cadence scans as well as angular resolution in both elevation and azimuth and a unique orbit due to Solar Orbiter's planned trajectory of up to $\sim$30° out of the ecliptic plane in the extended mission.

Additionally, having direct angular resolution of these measurements will allow us to make more accurate comparisons with the IMF conditions. While SO-HIS has some limitations due to a post-acceleration voltage that is not optimized for low charge-state PUIs and mainly sampling in the radial solar direction, causing the distribution to be dominated by solar wind ions, HIS will provide scans in a low cadence Normal Mode at 5 min resolution, but also at a 30 s Normal Mode and a 4 s Burst Mode, and has a relatively large azimuthal field of view in the ecliptic, with a range of -30° to +66° from the Sun-spacecraft line, which will place more of the PUI distribution within the field of view. Additionally, HIS provides angular scans in 16 steps in the range $\pm$17° with an angular resolution of $<$3.5°. These measurements, along with future instruments with added improvements, will greatly enhance our ability to study the nature and evolution of the Heliosphere — interstellar interactions.

4.6 Conclusions

We discuss various challenges that arise when using current PUI measurements over limited subsets of existing data sets. Considerations must be made for PUI transport as well as for solar cycle and solar transients in addition to measurement conditions and instrument data loss. Taking measures such as removing transients and restricting the data set for perpendicular solar wind velocity to IMF conditions and potentially limiting data by instrument sector can be useful in accounting for these effects, however, removing too much data can also obscure or remove the necessary signatures needed for the required observations. Past studies using PUI data have either partially neglected to account for PUI transport or have depended on the combination of larger data sets or large data smoothing windows and averages. These techniques can be problematic when trying to measure a precise value to determine a trend over time from limited subsets of the data.

We present an alternative method to determine a trend from a yearly flow direction, λ_{ISN}, of the interstellar wind through the Heliosphere using the PUI cutoff method in the upwind helium crescent. Preliminary measurements yield a unified flow direction, λ_{ISN}, of 73.74° $\pm$2.25° and a weighted average of individual flow values outside solar maximum years for the years 2002 through 2009 (corresponding to crescent crossings in 2003 through 2010) of 74.89° $\pm$1.17°. While these values are still preliminary, they show promising results that the method can be improved to achieve values very close to the expected recent literature values for λ_{ISN}. While the preliminary best fit line yields a slope of 0.66° ±0.71°, a linear fit weighted by error to the values in the years 2002 through 2009 (corresponding to crescent crossings in 2003 through 2010) yields a slope of -0.11° ±0.90°. We note that our results are still heavily influenced by solar activity and compression regions, especially in years around solar maximum. Therefore, we will need to apply further techniques to filter out these effects without removing signatures of the focusing cone and crescent in order to report on a trend and measure an accurate unified flow direction from the combined data set.

Lastly, we discuss requirements for future observations to best utilize the instrumental yet currently limited PUI measurements. PUIs comprise a significant portion of the plasma distribution within the Heliosphere and provide useful measurements about non-solar source plasma sources, such as the interstellar medium, which affect the Heliosphere, yet they are limited by current measurement techniques. Many current missions, such as ACE/SWICS, a sun-pointed spinner measuring in the radial direction, are located and oriented such that measurements are dominated by the solar wind and miss much of the PUI distribution. Therefore, future missions intended to sample other portions of the *in situ* plasma from different locations would greatly increase the analysis potential of PUI measurements. Additionally, current ion mass spectrometers are limited in their ability to detect and distinguish PUI species in the solar wind. Therefore, it would be beneficial to have instruments with greater detection capabilities, such as more sensitive energy measurements and higher post-acceleration voltages, as well as higher time resolution scans and greater geometric factors. Future studies such as observations of the evolution of Heliosphere – interstellar interactions will greatly benefit from an improved and increased PUI data supply.

4.7 Acknowledgments

We thank the ACE/SWICS, ACE/MAG, and ACE/SWEPAM instrument teams and the ACE Science Center for providing the ACE data.

This work is supported by NASA's Solar Orbiter and Advanced Composition Explorer missions as well as grants AGS 1358268 for J. Gilbert, and S. T. Lepri and NNX16AP03H and 80NSSC20K0192 for S. T. Lepri. This work has additionally been supported by the Michigan Plasma Sciences & Engineering Graduate Fellowship, The Michigan Space Grant Consortium Graduate Fellowship, the American Geophysical Union Student Travel Grant, the University of Michigan Rackham Student Travel Grant, the University of Michigan Diversity, Equity, and Inclusion Grant in the Department of Climate and Space Sciences & Engineering, and the University of Michigan College of Engineering Office of Student Affairs Student Travel Grant.

CHAPTER 5

Conclusions and Future Work

This thesis explores the characterization of Heliosphere — interstellar interactions using PUI measurements to address the central science question: **How does the Heliosphere — Local Interstellar Medium interaction evolve over time?** This question is investigated through a study of the evolution of the longitudinal flow direction of the interstellar wind through the Heliosphere using PUI measurements in the downwind focusing cone and upwind crescent. Additionally, improvements to future analysis and measurement are explored through a discussion of future requirements for improved techniques and data sets, and a cross-calibration of the SO-HIS instrument, a new *in situ* TOF triple-coincidence ion mass spectrometer, is presented.

The central science question is broken down into the following sub-questions and addressed through the analyses presented in Chapters 2 through 4.

1. **To what precision can a yearly inflow direction of the interstellar wind through the Heliosphere be measured using PUI measurements in the focusing cone and crescent?**
2. **Is there evidence of variation in the longitudinal interstellar neutral inflow direction through the Heliosphere over an 11-year solar cycle?**
3. **How can new instrument designs and measurement techniques enhance our understanding of the Heliosphere — interstellar interactions?**

5.1 Cross-Calibration and Performance Assessment of the Solar Orbiter Heavy Ion Sensor with its Ion Optical Model

In Chapter 2, the SO-HIS instrument is discussed, and the methodology and results in cross-calibrating the flight instrument with its SIMION IOM are presented. The IOM is created and verified using laboratory measurements of the flight instrument, and a function of the instrument's geometric factor, a measure of its useful particle intake geometry, is characterized for standard solar wind conditions. This chapter addresses Science Question 3 by presenting the characterization of a new spaceflight instrument that will enable improved measurements that can be used to improve future studies of Heliosphere — interstellar interactions.

Improved measurements from SO-HIS will include higher cadence scans, angular resolution in both elevation and azimuth, including scanning over 16 elevation steps, and measurements taken away from the Solar equatorial plane, with inclinations of up to $\sim$30° out of the ecliptic in Solar Orbiter's extended mission. SO-HIS provides an elevation angle field of view of $\pm$17° and an azimuth field of view of -30° to +66° with angular resolutions $<$3.5°. These ranges of 34° in elevation and 96° in azimuth provide improvements of 24° in elevation and 14° in azimuth over the ACE/SWICS field of view. Higher cadence scans, especially at the 30 s Normal Mode and 4 s Burst Mode time resolutions, will enable more accurate measurement selection, along with the elevation and azimuth angular resolutions, when comparing solar wind and IMF conditions, improving the useful PUI data set for studying the longitudinal flow direction, λ_{ISN}. Measurements taken at varying inclinations and distances from the Sun will additionally provide information useful for mapping and tracking the 3D structures of the interstellar PUI distribution in the inner Heliosphere, which are created due to the interactions of the Heliosphere with the LISM.

5.2 Determining the Interstellar Wind Longitudinal Inflow Evolution Using Pickup Ions in the Helium Focusing Cone

In Chapter 3, a study of the longitudinal inflow direction, λ_{ISN}, of the interstellar wind through the Heliosphere using a new method on He$^+$ PUI measurements in the downwind focusing cone is presented. The method is validated by determining a unified flow direction for the complete data set of 75.37° $\pm$0.43°, consistent with recent literature studies. This result is compared with other measurements in a survey of literature values published since the first measurements of λ_{ISN} in 1974. The method described utilizing measurements in the focusing cone is applied to 3-orbit boxcar averages of combined count distributions, and a trend of 0.00° $\pm$0.51° is measured, indicating that a variation in the flow direction is not detected over the 11-year solar cycle with focusing cone crossings occurring in the years 1999 through 2009. This chapter addresses Science Questions 1 and 2 by measuring the interstellar flow direction and trend using PUI measurements in the downwind helium focusing cone.

5.3 Implications of Transport and Challenges in Interstellar Flow Measurements Using Pickup Ions

In Chapter 4, challenges using current measurements and analysis techniques on the existing PUI data set are discussed. Past measurements from the literature of the longitudinal flow direction, λ_{ISN}, are evaluated in the context of these challenges. Ongoing efforts to further characterize the Heliosphere — interstellar interactions by determining a trend from precise yearly flow directions measured using the direct PUI cutoff method on measurements in the downwind crescent are presented. Lastly, improvements for future measurements and analysis techniques are proposed to enhance future studies of these interactions. This chapter addresses Science Questions 1 through 3 by proposing a technique to measure the yearly flow direction, λ_{ISN}, as well as a unified flow direction from the aggregate data set, showing promising preliminary results, and discussing improvements to future measurements and analysis.

Preliminary measurements appear to be biased by a solar cycle dependency, yielding a slope to the best fit line of the 3-orbit center years 1999 through 2009 of 0.66° ±0.71° and a unified flow direction, λ_{ISN}, from the data set including complete orbits from 1998 through 2010 of 73.74° ±2.25°. However, promising results are seen when taking the best fit line for preliminary 3-orbit center year flow directions measured in the years 2002 through 2009, away from the strongly impacted years 1998 through 2001. These measurements yield a best fit line slope of -0.11° ±0.90° and a weighted average value for λ_{ISN} of 74.89° ±1.17°. These promising results indicate that future improvements applied to the direct cutoff method to decouple the measurements from the apparent solar cycle dependency may yield an even higher level of precision and accuracy for measuring a trend and yearly flow direction, λ_{ISN}.

Requirements for future measurements to study the Heliosphere — interstellar interactions using PUIs include improvements in spacecraft location and look direction as well as instrument measurement capabilities. Improved spacecraft orientations for PUI measurements include a full 360° azimuthal field of view to reduce the effects of the solar wind dominating the distribution in the solar radial direction, as well as an extended elevation field of view, such as ±30°. Improvements to instrument capabilities include enhanced PUI detection ability through greater post-acceleration voltages and higher detector efficiencies, higher time resolution scans, and greater geometric factors.

5.4 Future Work

Future work arising from the studies presented herein includes the determination of precise yearly flow directions, λ_{ISN}, and a subsequent trend measurement using the direct PUI cutoff method from PUI measurements in the upwind crescent. Additionally, an expansion of the study of the Heliosphere — interstellar interactions through the mapping and tracking of the 3D structures of the downwind cone and upwind crescent is planned. Lastly, a more complete characterization of the SO-HIS geometric factor is planned to include energies less abundant in the solar wind and to characterize edge effects below -15° elevation and azimuths beyond 0° to 30° as well as effects of the finer resolution “dead zones” on the SSD near the spokes and between the SSD grid cells.

APPENDIX A

Adding a Carbon Foil and Ion Suppression Grid to the SIMON IOM

The SO-HIS instrument includes an electron suppression grid and a carbon foil at the entrance to the TOF chamber. In the instrument, the suppression grid stops secondary electrons from flying backward into the EA-IS section, while the carbon foil allows the initial ion to pass into the TOF chamber, generally becoming neutralized in the process (Allegrini et al. (2016)), while generating secondary electrons to trigger the start timing signal. The presence of the grid and the carbon foil affect how the particles fly through the instrument and are necessary for SIMION to calculate the electric potentials which affect the trajectories of the charged particles. One very noticeable effect of leaving off the grid and carbon foil as potential arrays in SIMION is for simulated particles to fly backward into the EA-IS rather than through the TOF chamber as expected. Therefore, the grid and the carbon foil must be inserted into the simulation. However, the grid and carbon foil must be treated by SIMION as having zero thickness, as any potential array in SIMION having greater than 1 pixel of thickness is treated as a solid surface which will stop a particle and not allow it to pass through, while a 1 pixel potential array, while still carrying a potential which will affect the electric fields and therefore the trajectory of an ion or electron, is treated as having zero thickness and will allow particles to pass through.

While the grid and carbon foil are inserted as zero thickness potential arrays to allow particles to pass through, it is important to consider other properties in addition to their potentials. These other properties, namely their transmissions, effect on particle charge, scattering and energy loss, and the release of secondary electrons, are accounted for in the Monte Carlo simulation LUA code and post-processing code. Both the grid and carbon foil are found to have $\sim$73% transmission rate, which is accounted for in post-processing when calculating GF. Additionally, the carbon foil will affect the charge of the particle as it enters the TOF chamber, with the particle usually exiting the carbon foil neutralized, though it may exit with a charge, such as ± 1 or ± 2. The carbon foil additionally causes scattering and energy loss and releases secondary electrons. The statistical effects of the scattering and energy loss are simulated using the TRIM software, and all of these effects are accounted for in the Monte Carlo simulation LUA code.

Figure A.1 shows a view in SIMION of the the overlapping inserted zero thickness carbon foil and electron suppression grid potential arrays in the SO-HIS ion optical model (IOM) in the x-y plane. Figure A.2 shows a cut in the SIMION x-y plane revealing the carbon foil PA in a view looking in the direction down the TOF chamber.

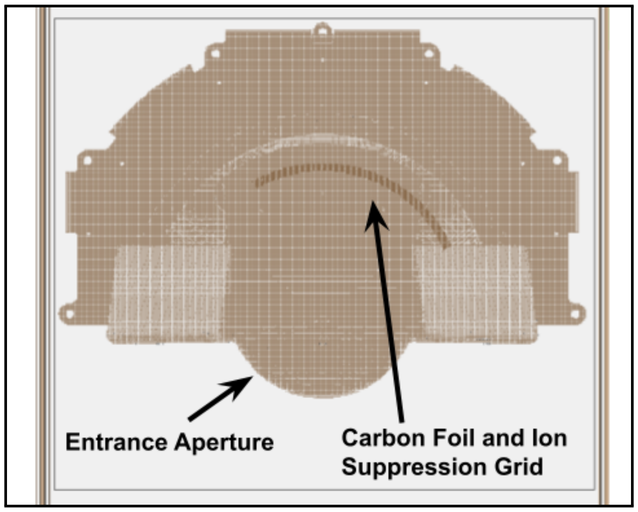


Figure A.1: View in the SIMION x-y plane showing the overlapping inserted zero thickness carbon foil and electron suppression grid potential arrays.

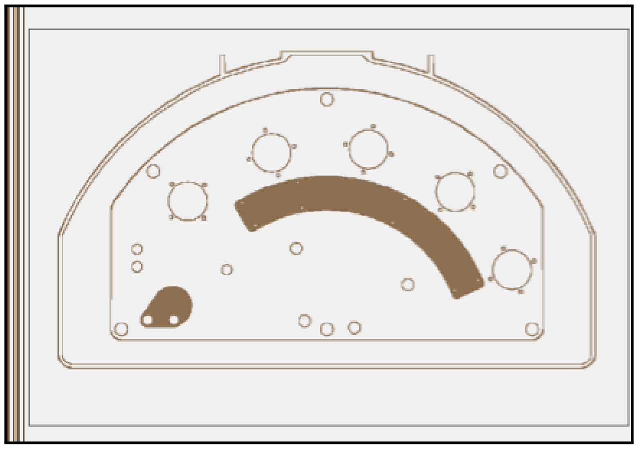


Figure A.2: A cut in the SIMION x-y plane revealing the carbon foil PA in a view looking in the direction down the TOF chamber.

The following GEM file was used to add a zero thickness carbon foil and electron suppression grid to the SO-HIS SIMION IOM at the entrance of the TOF chamber. This GEM file was based on a GEM file originally written by Jason Gilbert.

Listing A.1: SO-HIS carbon foil and electron suppression grid GEM file.

```
; HIS_cf_and_grid.gem
; Sarah A. Spitzer (based on file from Jason A. Gilbert)
; saraylet@umcih.edu (jagi@umich.edu)
; Created 12/11/2020
; Last Edited 12/15/2020
; Creates 0 thickness grid and carbon foil

;pa_define: inputs on page 316. x,y,z, defines location of
   ↪ maximum coordinates from defined origin,
;as defined by locate command; ie. this defines the size of your
   ↪ bounding box

;locate: defines the location of your origin for this bounding
   ↪ box, ie where the minimum coordinates will be
;pg. 312


pa_define(880,850,450,p,none,e) ; define array bounds in mm
locate(0,0,0,4,0,0,0) ; scale for 0.25 mm/gu

{
locate(145.44,136.15,0,1,0,0,0)
{
e(5) ; Carbon foil voltage
   {

            ; CF
      fill
      {
                  within{cylinder(0,0,108.72,73,73,0)} ; annular
                     ↪ outer circle (x,y,z (origin of cylinder),
                     ↪  radial x, radial y, z length (thickness
                     ↪ =0))
                  notin{cylinder(0,0,108.72,67,67,0)} ; annular
                     ↪ inner circle
                  locate(0,0,0,1,0,22,0) ; cut off excess (want a
                     ↪  left rotation of top of box 24 degrees
                     ↪ minus buffer)
                  {
                        notin{box3d(0,-100,108,100,100,109)} ;
                           ↪ make a really big box and cut
                           ↪ everything out
                  }
                  locate(0,0,0,1,0,-58,0) ; cut off excess (want
                     ↪ a left rotation of bottom of box 60
                     ↪ degrees minus buffer)
                  {
                        notin{box3d(0,-100,108,100,100,109)} ;
                           ↪ make a really big box and cut
                           ↪ everything out of the other side
                  }


         }


            ; grid
      fill
      {
                  within{cylinder(0,0,109.75,73,73,0)} ; annular
                     ↪ outer circle (x,y,z (origin of cylinder),
                     ↪  radial x, radial y, z length (thickness
                     ↪ =0))
                  notin{cylinder(0,0,109.75,67,67,0)} ; annular
                     ↪ inner circle
                  locate(0,0,0,1,0,22,0) ; cut off excess (want a
                     ↪  left rotation of top of box 24 degrees
                     ↪ minus buffer)
                  {
                        notin{box3d(0,-100,109,100,100,110)} ;
                           ↪ make a really big box and cut
```

```
                          ↪ everything out
                  }
                  locate(0,0,0,1,0,-58,0) ; cut off excess (want
                      ↪ a left rotation of bottom of box 60
                      ↪ degrees minus buffer)
                  {
                        notin{box3d(0,-100,109,100,100,110)} ;
                          ↪ make a really big box and cut
                          ↪ everything out of the other side
                  }

         }

      } ; end of voltage setting
}
} ; end of main locate command
```

APPENDIX B

The SO-HIS SIMION IOM Source Surface

B.1 Background

The source surface in the SO-HIS SIMION ion optical model (IOM) is the region in the z-y plane from which particles are defined to be generated. The source surface parameters are defined in the FLY2 code, included in Appendix D.

The following calculations are used to ensure that the SIMON source surface is appropriately sized in azimuth to fully cover the SO-HIS aperture.

In general, rays coming at a uniform angle from a single direction can enter 180° of a circle, as shown in Figure B.1.

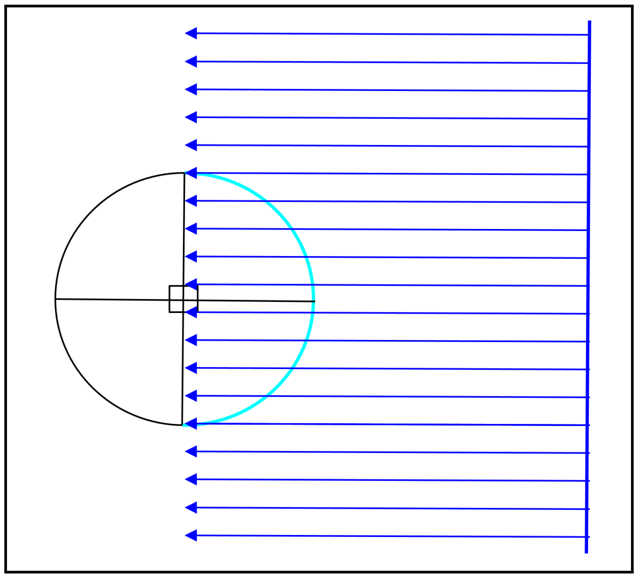


Figure B.1: Diagram showing the maximum 180° illumination of a theoretical circle in the instrument's azimuthal plane.

These rays, representing particle trajectories, at 0°, can penetrate angles on the circle between +90° and -90°. In general, rays can penetrate an angular range of ϕ + 90° to ϕ - 90°.

On HIS, we are looking at the arc on the circle corresponding to ray angles of +66° to -30°, as shown in Figure B.2.

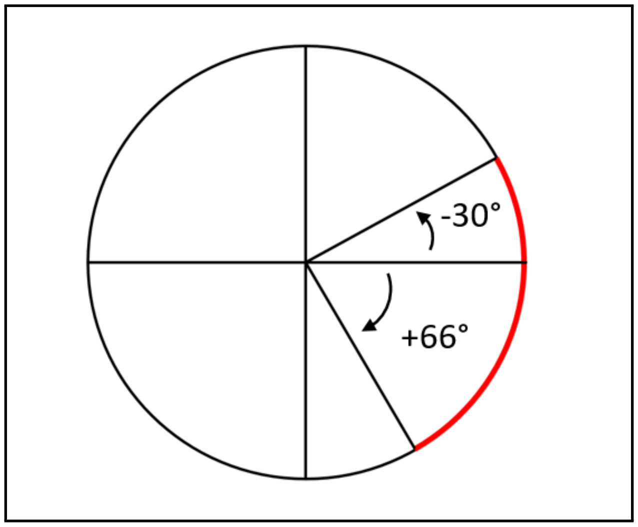


Figure B.2: Diagram showing the SO-HIS azimuthal FOV from -30° to +66° projected onto a circle.

Rays from +66° can penetrate the arc from angles of 66° + 90° = 156° to 66° - 90° = -24°. This means that in this case, angles from -24° to -33° on the aperture arc will not be penetrated, as shown in Figure B.3.

Similarly, -30° rays can penetrate the arc from angles of -30° + 90° = 60° to -30° - 90° = -120°. This means that in this case, angles from 60° to 66°on the aperture arc will not be penetrated.

Since the SO-HIS acceptance range is 96°, there may be up to 6° on either edge of the arc that could not be penetrated.

In general, for rays ranging from angles ϕ_{min} to ϕ_{max}, angles that can be penetrated include ϕ_{max} + 90° through ϕ_{min} - 90°. Any outlying angles on the aperture will not be penetrated.

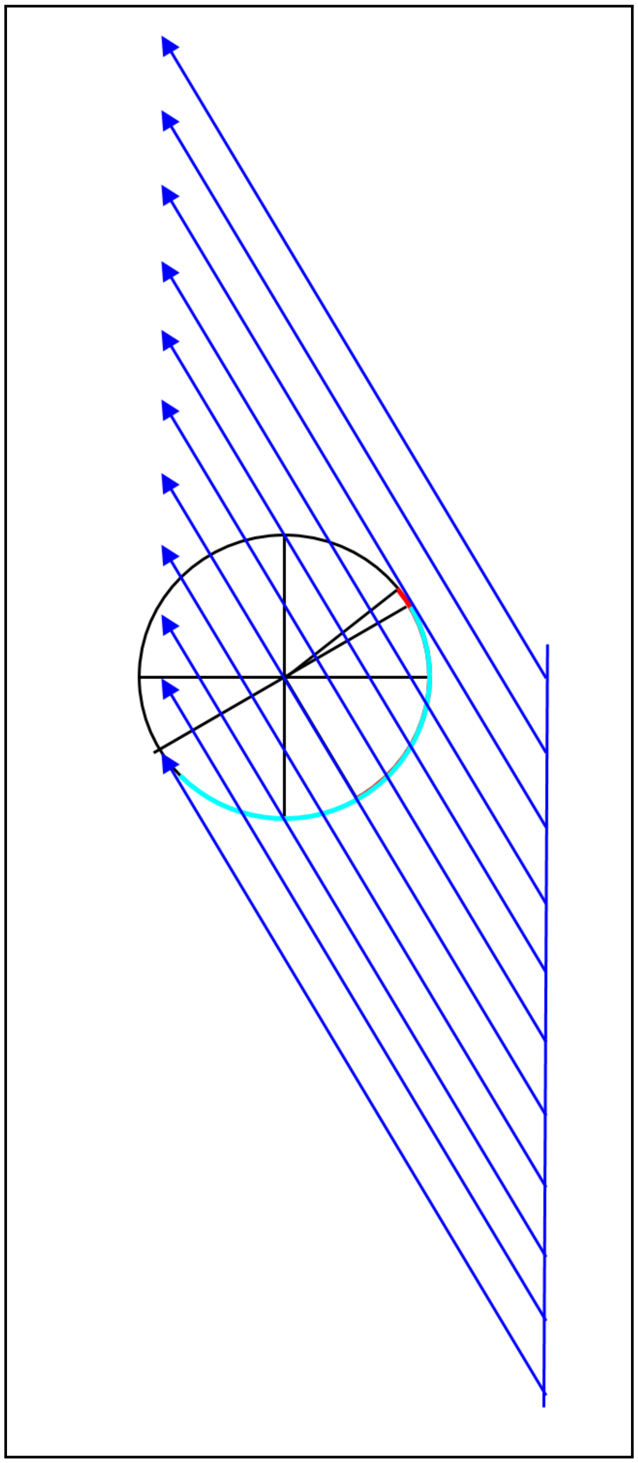


Figure B.3: Diagram showing the illumination of the HIS aperture FOV projected onto a circle for rays, representing particle trajectories, at an azimuthal angle of +66°.

B.2 Defining the SO-HIS SIMION IOM Source Surface Limits

B.2.1 SO-HIS SIMION Coordinate Definitions

The 2D planar orientations for the SO-HIS SIMION IOM are shown in Figure B.4 through Figure B.6. The origin is located at the center of the top-cap along the x-axis, the center of the entrance aperture along the z-axis, and at 0° azimuth along the y-axis. The labeled values z_{min}, z_{max}, x_{min}, x_{max}, y_{min}, and y_{max} refer to the minimum and maximum Cartesian bounds of the heat shield aperture. xSS_{min} and xSS_{max} refer to the minimum and maximum distances of the source surface from the heat shield aperture bounds along the x-axis. Source surface x is defined as the location of the source surface on the x-axis. This value is set to 50 mm for our simulations to ensure coverage of the instrument aperture. No appreciable entrance aperture fringe field effects are found at this distance.

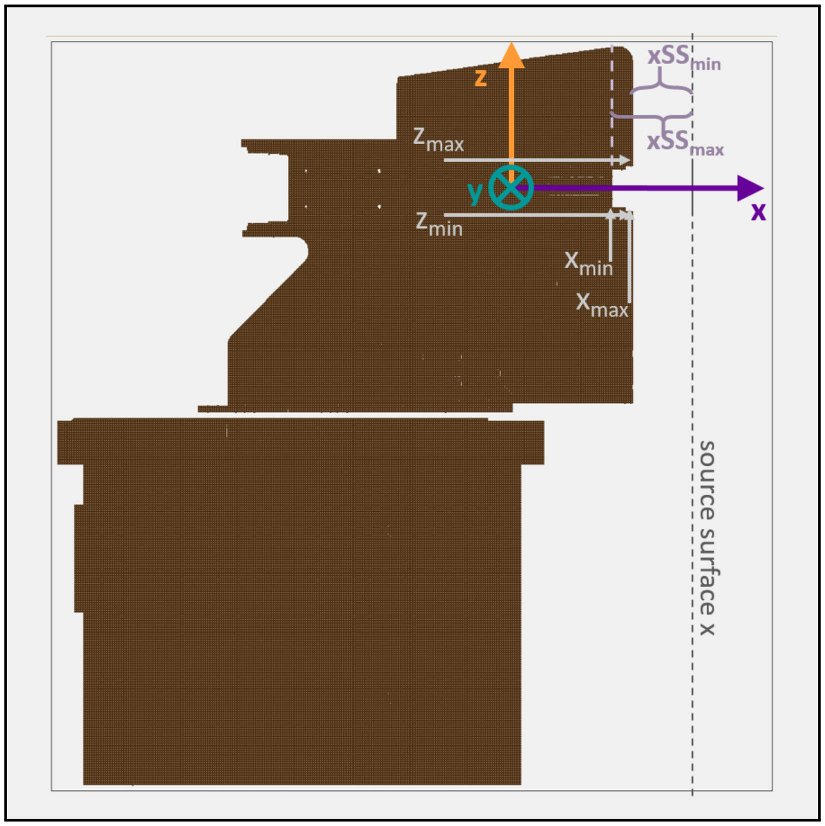


Figure B.4: 2D z-x planar orientation for the SO-HIS SIMION IOM with labels for the source surface x location, the minimum and maximum Cartesian bounds, z_{min}, z_{max}, x_{min}, and x_{max}, for the heat shield aperture in along the z-axis and x-axis, and the minimum and maximum distances of the source surface from the heat shield aperture bounds along the x-axis, xSS_{min} and xSS_{max}.

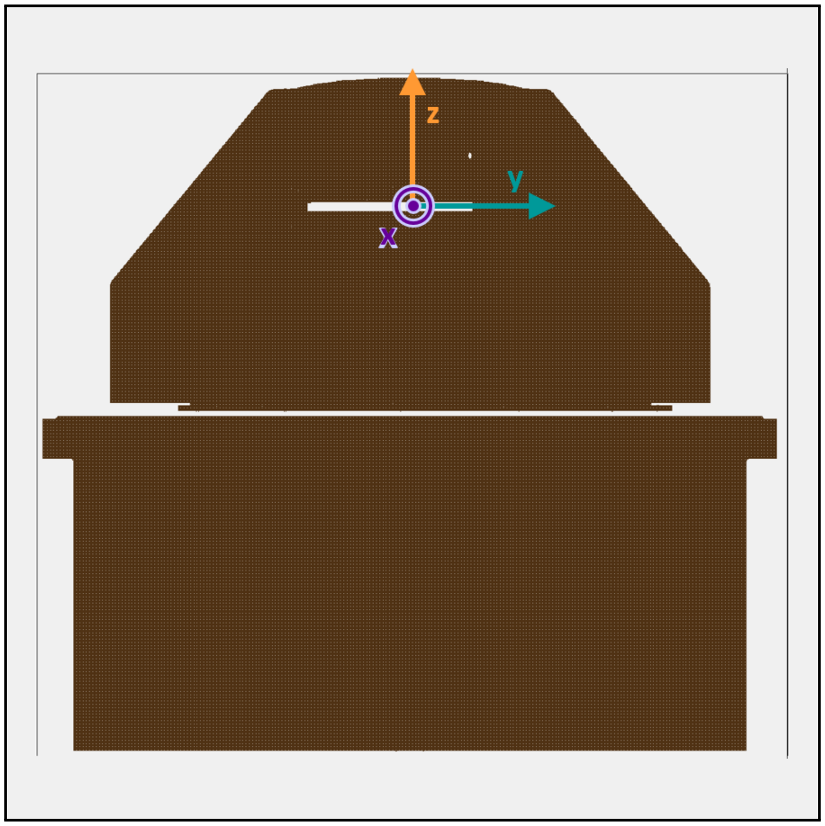


Figure B.5: 2D z-y planar orientation for the SO-HIS SIMION IOM.

B.2.2 Angular Input Bounds

The angular bounds for initial particles are defined from user inputs. For a uniform or arithmetic distribution, the minimum and maximum angles are set directly by the user as θ_{min} and θ_{max} for elevation and ϕ_{min} and ϕ_{max} for azimuth. These values are used directly when calculating the source surface bounds. For a normal distribution, the mean angle and standard deviation are set by the user as θ_{mean} and σ_{elv} for elevation and ϕ_{mean} and σ_{az} for azimuth. The minimum and maximum angles used in calculating the bounds for the source surface for a normal distribution are then calculated as:

$$\theta_{min} = \theta_{mean} - 3\,\sigma_{elv} \tag{B.1}$$

$$\theta_{max} = \theta_{mean} + 3\,\sigma_{elv} \tag{B.2}$$

$$\phi_{min} = \phi_{mean} - 3\,\sigma_{az} \tag{B.3}$$

$$\phi_{max} = \phi_{mean} + 3\,\sigma_{az} \tag{B.4}$$

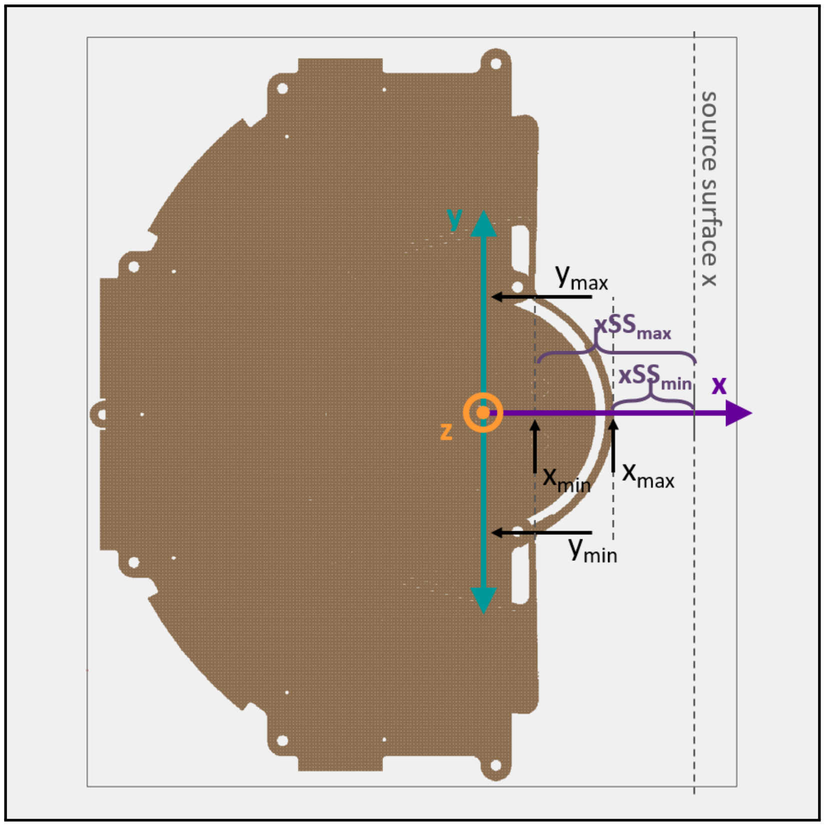


Figure B.6: 2D x-y planar orientation for the SO-HIS SIMION IOM with labels for the source surface x location, the minimum and maximum Cartesian bounds, y_{min}, y_{max}, x_{min}, and x_{max}, for the heat shield aperture in along the y-axis and x-axis, and the minimum and maximum distances of the source surface from the heat shield aperture bounds along the x-axis, xSS_{min} and xSS_{max}.

Particles are then generated at the source surface uniformly for a uniform distribution and using arithmetic steps set by the user input and currently defined, though not used by our simulations, as 96 for azimuth and 40 for elevation, for an arithmetic distribution between the given minimum and maximum angular bounds or using a normal distribution with the input mean and standard deviation.

It may be noted that given the instrument orientation and the requirements on azimuth and elevation angles in the SIMION environment shown in Figure B.7, the elevation parameters must be input to the SIMION azimuth variable with a 180° shift, and the azimuth parameters must be input to the SIMION elevation variable. This is accounted for in the FLY2 code, which defines particles generated at the source surface.

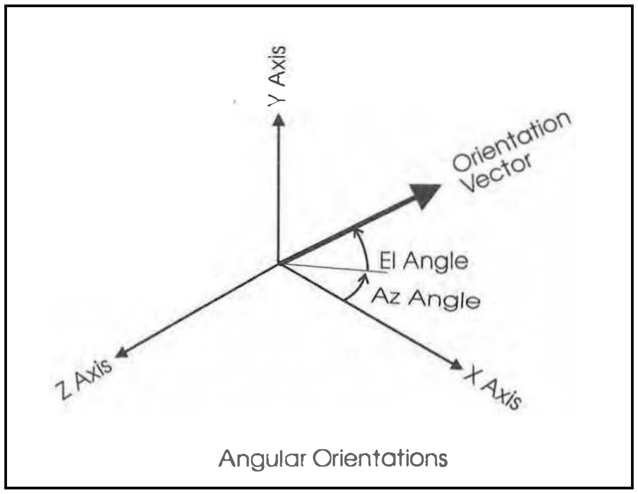


Figure B.7: Figure L.2 from the Simion Version 8.0 User Manual showing the SIMION defined azimuth and elevation angles (Manura & Dahl (2007)).

B.2.3 Source Surface Bounds

Source surface examples are shown with limits along the z-axis in Figures B.8 and B.9 along with the definition of positive and negative elevation angles in the top left corner, the general length calculation in the z direction is shown in Figure B.10, source surface examples with limits along the y-axis are shown in Figures B.11 and B.12 along with the definition of positive and negative azimuth angles in the top left corner, and the general length calculation in the y direction is shown in Figure B.13.

The limits for the source surface bounds, zSS_{min} and zSS_{max}, along the z-axis are given as:

$$\theta < 0 : x_{d1} = xSS_{max}, x_{d2} = xSS_{min} \tag{B.5}$$

$$\theta \geq 0 : x_{d1} = xSS_{min}, x_{d2} = xSS_{max} \tag{B.6}$$

$$zSS_{min} = z_{min} + x_{d1}\,\tan\theta_{min} \tag{B.7}$$

$$zSS_{max} = z_{max} + x_{d2}\,\tan\theta_{max} \tag{B.8}$$

where the “equal to 0” condition is assigned to the “greater than” case.

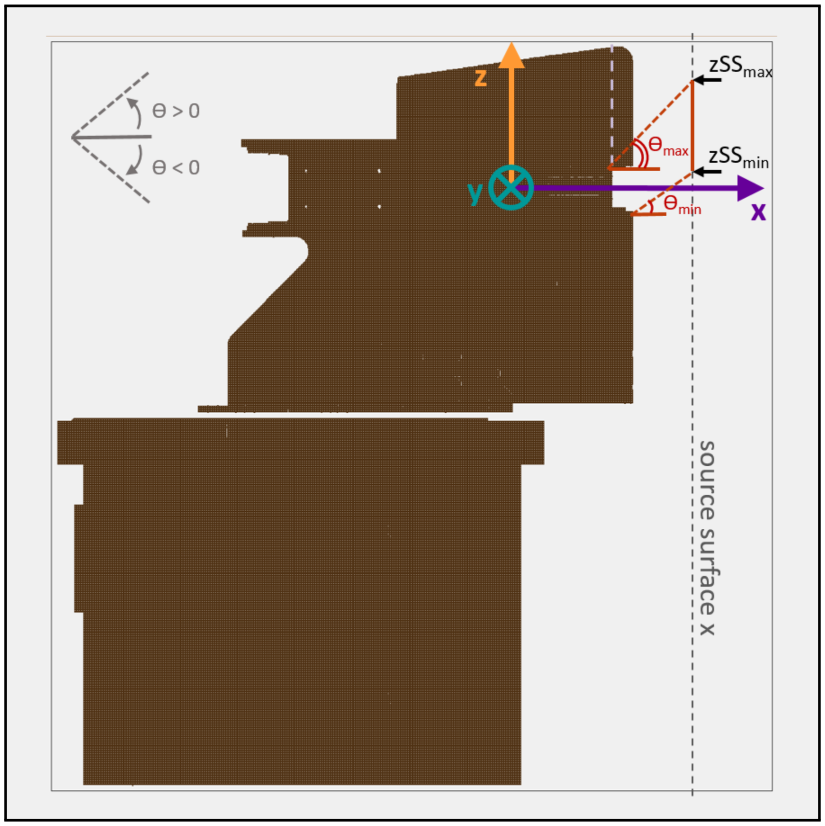


Figure B.8: Source surface example with limits along the z-axis, for only positive initial particle elevation trajectories. The minimum bound on the source surface along the z-axis, zSS_{min}, is defined from the closest point along the x-axis on the heat shield aperture and the minimum elevation angle of initial particle trajectories. The maximum bound on the source surface along the z-axis, zSS_{max}, is defined from the farthest point along the x-axis on the heat shield aperture and the maximum elevation angle of initial particle trajectories. The definition of positive and negative elevation angles is shown in the top left corner.

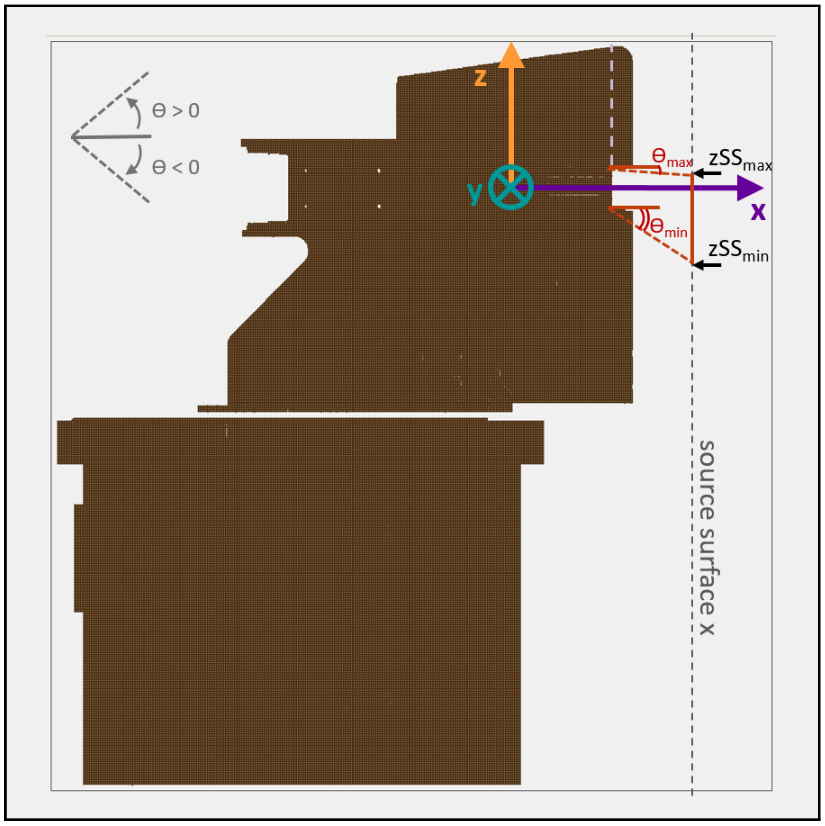


Figure B.9: Source surface example with limits along the z-axis, for only negative initial particle elevation trajectories. The minimum bound on the source surface along the z-axis, zSS_{min}, is defined from the farthest point along the x-axis on the heat shield aperture and the minimum elevation angle of initial particle trajectories, which will be a negative angle in this case. The maximum bound on the source surface along the z-axis, zSS_{max}, is defined from the closest point along the x-axis on the heat shield aperture and the maximum elevation angle of initial particle trajectories, which will be a negative angle in this case. The definition of positive and negative elevation angles is shown in the top left corner.

The general case for a length in the z direction from elevation angle, θ, is given as:

$$\begin{aligned} \tan\theta &= z/x \\ z &= x\,\tan\theta \end{aligned} \tag{B.9}$$

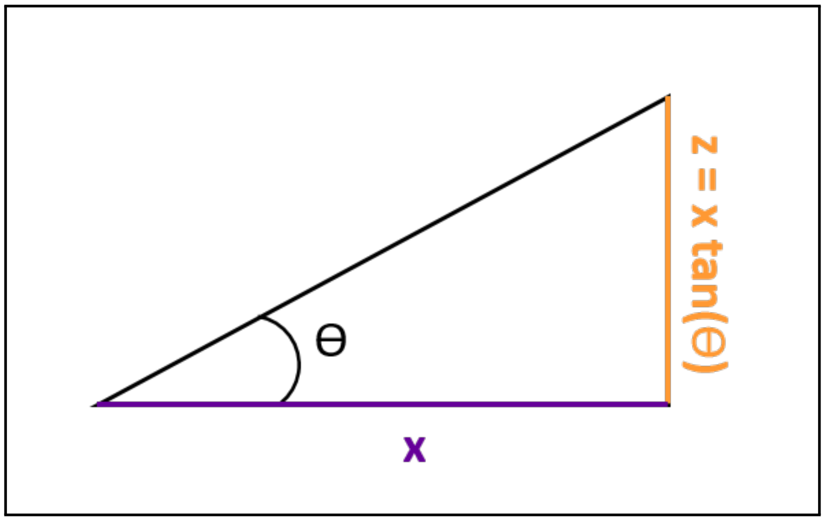


Figure B.10: The general case for determining a length in the z direction from elevation angle, θ.

The limits for the source surface bounds, ySS_{min} and ySS_{max}, along the y-axis are given as:

$$\phi < 0 : x_{d1} = xSS_{min}, x_{d2} = xSS_{max} \tag{B.10}$$

$$\phi \geq 0 : x_{d1} = xSS_{max}, x_{d2} = xSS_{min} \tag{B.11}$$

$$ySS_{min} = y_{min} - x_{d1}\,\tan\phi_{max} \tag{B.12}$$

$$ySS_{max} = y_{max} - x_{d2}\,\tan\phi_{min} \tag{B.13}$$

where the “equal to 0” condition is assigned to the “greater than” case.

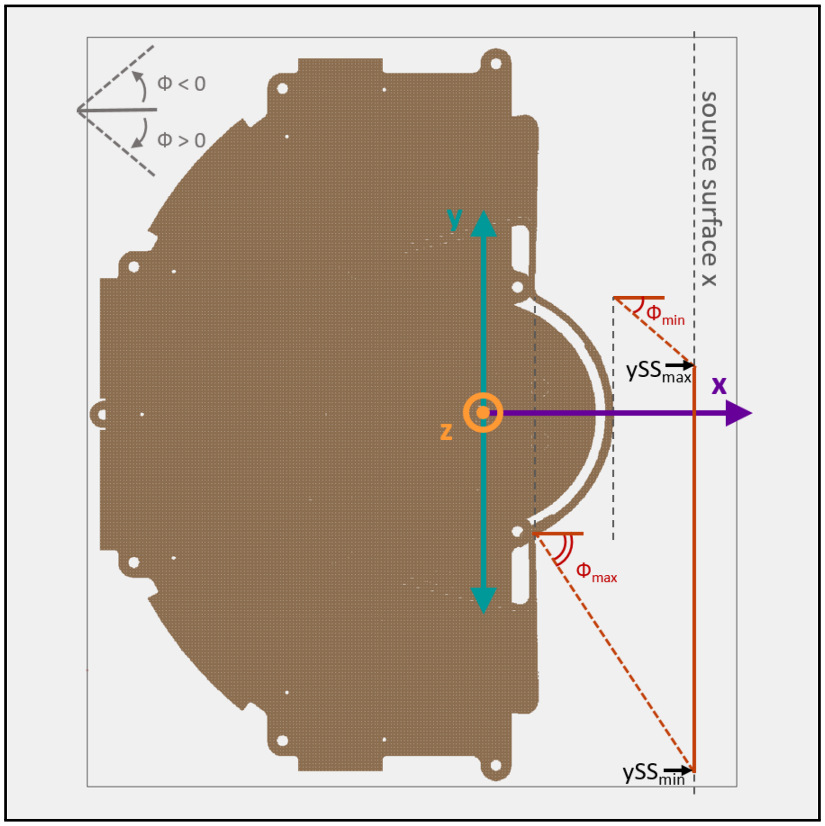


Figure B.11: Source surface example with limits along the y-axis, for only positive initial particle azimuth trajectories. The minimum bound on the source surface along the y-axis, ySS_{min}, is defined from the farthest point along the x-axis on the heat shield aperture and the maximum azimuth angle of initial particle trajectories. The maximum bound on the source surface along the y-axis, ySS_{max}, is defined from the closest point along the x-axis on the heat shield aperture and the minimum azimuth angle of initial particle trajectories. The definition of positive and negative azimuth angles is shown in the top left corner.

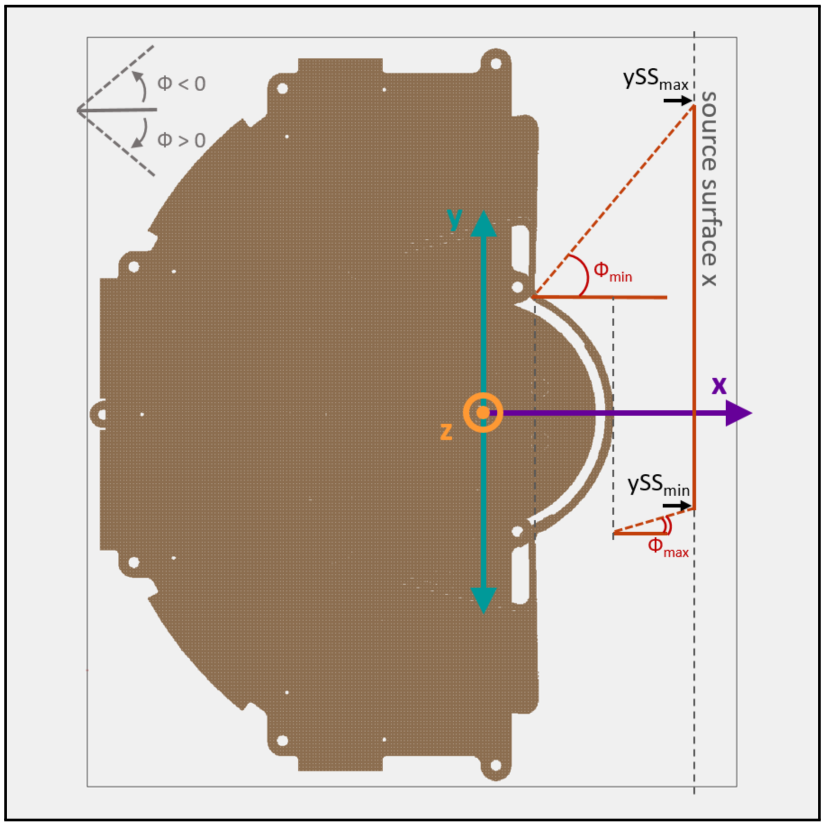


Figure B.12: Source surface example with limits along the y-axis, for only negative initial particle azimuth trajectories. The minimum bound on the source surface along the y-axis, ySS$_{min}$, is defined from the closest point along the x-axis on the heat shield aperture and the maximum azimuth angle of initial particle trajectories, which will be a negative angle in this case. The maximum bound on the source surface along the y-axis, ySS$_{max}$, is defined from the farthest point along the x-axis on the heat shield aperture and the minimum azimuth angle of initial particle trajectories, which will be a negative angle in this case. The definition of positive and negative azimuth angles is shown in the top left corner.

The general case for a length in the y direction from azimuth angle, ϕ, is given as:

$$\tan\phi = y/x$$
$$y = x\,\tan\phi \tag{B.14}$$

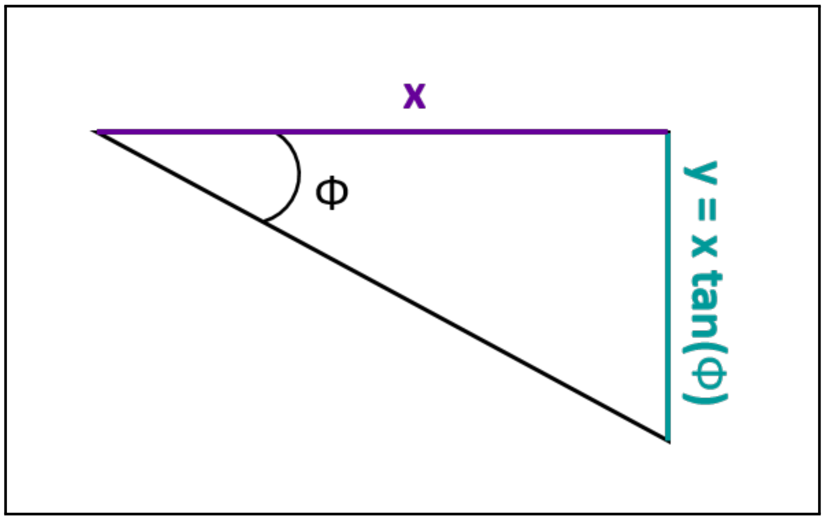


Figure B.13: The general case for determining a length in the y direction from azimuth angle, ϕ.

APPENDIX C

A Beam on the HIS Aperture

This appendix demonstrates how a uniform beam on the aperture will create a seeming “hot spot” at the center of the beam and how a straight source surface with angled particle trajectories can be used to represent an angled solar wind beam.

The panel plot in Figure C.1 shows particles that reach various surfaces in the SO-HIS SIMION ion optical model (IOM). The top panel shows a uniformly illuminated aperture given a limited initial number of particles (100k). The second panel shows a “hot spot” on the aperture at the center of the beam in azimuth and elevation, 0° for both in this case (see hit angle axis in azimuth at the bottom of the figure). The rest of the panels show particles that make it through to the exit of the ESA, to the carbon foil, past the carbon foil, to the SSD, to the Start MCP, and to the Stop MCP in polar coordinates, with the radial component on the vertical axis and the angular component on the horizontal axis. Hit angle for the source surface, not plotted here, is defined in the z-y plane. All other hit angles are defined in the x-y, azimuthal plane.

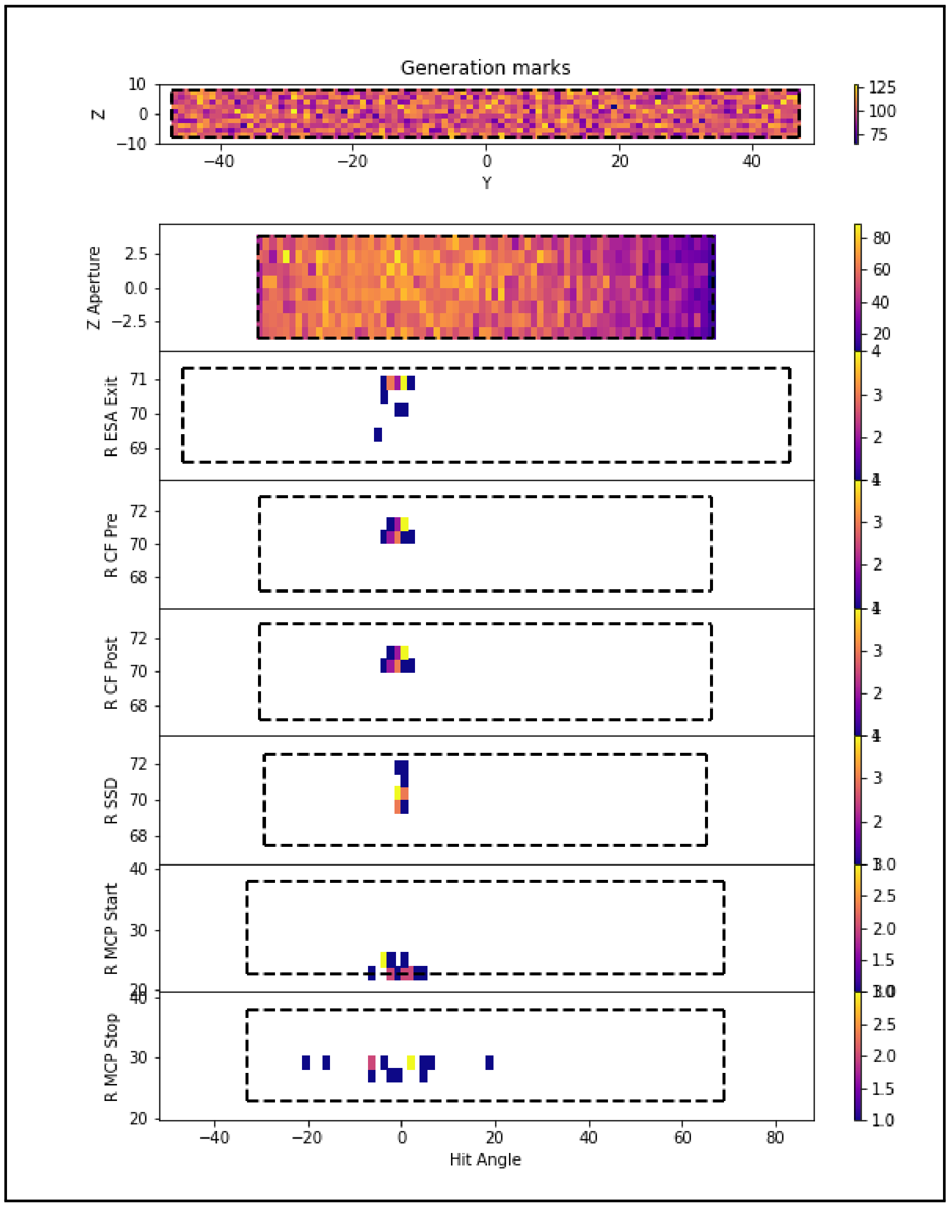


Figure C.1: Panel plot showing particles that reach various surfaces in the SO-HIS SIMION IOM. The top panel shows a uniformly illuminated aperture given a limited initial number of particles (100k). The second panel shows a “hot spot” on the aperture at the center of the beam in azimuth and elevation, 0° for both in this case (see hit angle axis in azimuth at the bottom of the figure). The rest of the panels show particles that make it through to the exit of the ESA, to the carbon foil, past the carbon foil, to the SSD, to the Start MCP, and to the Stop MCP in polar coordinates, with the radial component on the vertical axis and the angular component on the horizontal axis. Hit angle for the source surface, not plotted here, is defined in the z-y plane. All other hit angles are defined in the x-y, azimuthal plane.

The following figures demonstrate the general geometry for how rays penetrating a circle, and therefore particles entering the instrument aperture, at angles increasingly tangential to the circle will become more spread out.

Figure C.2 shows rays, or particle trajectories, entering the aperture at 0° azimuth.

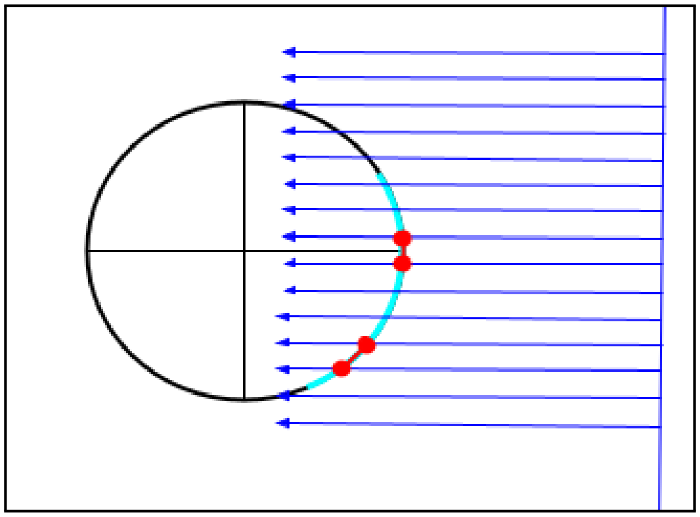


Figure C.2: Diagram demonstrating the spacing between rays entering a circle, representing particle trajectories entering the aperture at 0° azimuth.

Rays are spaced uniformly apart with spacing d on a straight source surface (beam). Rays entering approximately perpendicular to the tangent, or with angle $\alpha = 90°$ relative to the tangent of the circle, shown in Figure C.3, will have spacing approximately d apart. At increasingly tangential angles, or smaller angles α relative to the tangent on the circle, the spacing between the penetrating rays will increase. The full distance between two rays centered about a point with angle α to the tangent will be approximately $d/\sin\alpha$. The approximate half-distance is demonstrated in Figure C.3. At the point perpendicular to the tangent on the cirlce, that is $d/\sin 90° = d$, will be the minimum spacing. At angles decreasing from 90° to approaching 0° (where rays become tangent and can no longer enter), the spacing $d/\sin\alpha$ will continue to increase, and rays will enter the circle farther and farther apart.

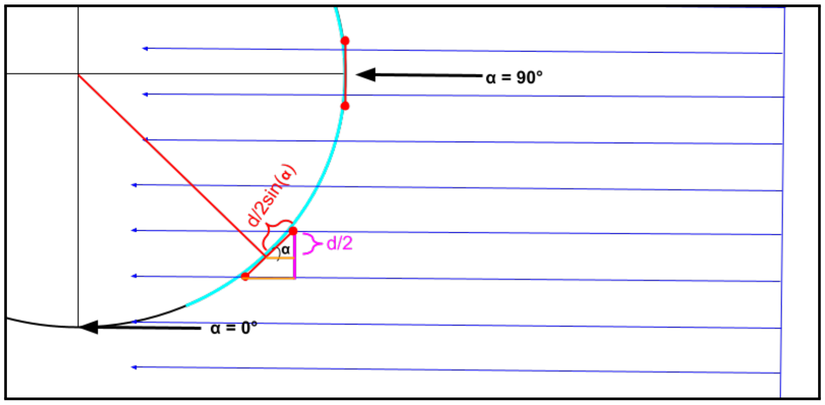


Figure C.3: Diagram showing how the spacing between rays will increase as they enter the circle more tangentially, or as α approaches 0°.

When considering an angled source surface (solar wind beam) versus a simulated or approximated beam using a non-angled source with angled ions, I suggest that the result is the same, assuming we use the cosine of the angle to determine the actual source surface area of the angled beam, as shown in Figure C.4, where S is the simulated source surface and B is the angled solar wind beam.

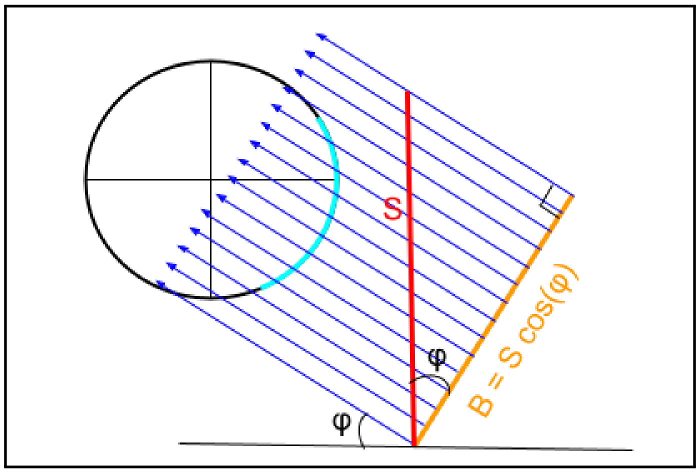


Figure C.4: Diagram showing how a straight source surface (S, red) with angled rays, or particle trajectories, can be used to represent an angled solar wind beam (B, orange) with straight rays, or particle trajectories at azimuth angle ϕ.

APPENDIX D

SIMION Fly2 Code

Details of the SIMION Monte Carlo simulation are discussed in Chapter 2. The SIMION FLY2 code used to describe the source surface is included below. The source surface is discussed in more detail in Appendix B.

This code was contributed to by: Jason Gilbert, Sarah Spitzer, Austin Glass, Matthew Panning, Connor Raines, and Mark Stakhiv.

Listing D.1: FLY2 code used to define the SIMION source surface used in the Monte Carlo Simulation.

```
-- This file controls the generation of particles based on the
   ↪ initial conditions of the SimParameters or
   ↪ SimParametersBatch file.

batch = {}
dofile("SimParameters.lua") -- Load in the parameters detailed in
   ↪  the file generated by the Excel spreadsheet



if vars.simulation_batch_mode == 1 then -- If you have indicated
   ↪ a batch run, also load SimParametersBatch params
  dofile("SimParametersBatch.lua")
end

-- In all cases, take either the batch version (if present) or
   ↪ the initial values from SimParameters
local aberration = batch.aberration or vars.aberration
local az_fov_max = batch.az_fov_max or vars.az_fov_max
local az_fov_min = batch.az_fov_min or vars.az_fov_min
local az_fov_spacing = batch.az_fov_spacing or
   ↪ vars.az_fov_spacing
local cfoil_neutralize = batch.cfoil_neutralize or
   ↪ vars.cfoil_neutralize
local cfoil_thickness = batch.cfoil_thickness or
   ↪ vars.cfoil_thickness
local cfoil_transmission = batch.cfoil_transmission or
   ↪ vars.cfoil_transmission
local dwell = batch.dwell or vars.dwell
local flux = batch.flux or vars.flux
local ke_dist = batch.ke_dist or vars.ke_dist
local ke_max = batch.ke_max or vars.ke_max
local ke_mean = batch.ke_mean or vars.ke_mean
local ke_min = batch.ke_min or vars.ke_min
local ke_stdev = batch.ke_stdev or vars.ke_stdev
local m = batch.m or vars.m
local q = batch.q or vars.q
local simulation_batch_mode = batch.simulation_batch_mode or
   ↪ vars.simulation_batch_mode
local simulation_eais_only = batch.simulation_eais_only or
   ↪ vars.simulation_eais_only
local simulation_k = batch.simulation_k or vars.simulation_k
local simulation_mode = batch.simulation_mode or
   ↪ vars.simulation_mode
local simulation_sec_e_total = batch.simulation_sec_e_total or
   ↪ vars.simulation_sec_e_total
local simulation_sec_p_total = batch.simulation_sec_p_total or
   ↪ vars.simulation_sec_p_total
local simulation_tr_qual = batch.simulation_tr_qual or
   ↪ vars.simulation_tr_qual
local source_surface_style = batch.source_surface_style or
   ↪ vars.source_surface_style
local source_surface_shape = batch.source_surface_shape or
   ↪ vars.source_surface_shape
local source_surface_x = batch.source_surface_x or
   ↪ vars.source_surface_x
```

```
local simulation_view_retain = batch.simulation_view_retain or
   ↪ vars.simulation_view_retain
local species = batch.species or vars.species
local tr_az_cover_the_spread = batch.tr_az_cover_the_spread or
   ↪ vars.tr_az_cover_the_spread
local tr_az_dist = batch.tr_az_dist or vars.tr_az_dist
local tr_az_max = batch.tr_az_max or vars.tr_az_max
local tr_az_mean = batch.tr_az_mean or vars.tr_az_mean
local tr_az_min = batch.tr_az_min or vars.tr_az_min
local tr_az_stdev = batch.tr_az_stdev or vars.tr_az_stdev
local tr_az_steps = batch.tr_az_steps or vars.tr_az_steps
local tr_el_dist = batch.tr_el_dist or vars.tr_el_dist
local tr_el_max = batch.tr_el_max or vars.tr_el_max
local tr_el_mean = batch.tr_el_mean or vars.tr_el_mean
local tr_el_min = batch.tr_el_min or vars.tr_el_min
local tr_el_stdev = batch.tr_el_stdev or vars.tr_el_stdev
local tr_el_steps = batch.tr_el_steps or vars.tr_el_steps
local U_Ground = batch.U_Ground or vars.U_Ground
local U_MCP_Stop = batch.U_MCP_Stop or vars.U_MCP_Stop
local U_MCP_Start = batch.U_MCP_Start or vars.U_MCP_Start
local U_MCP_Case = batch.U_MCP_Case or vars.U_MCP_Case
local U_PAC = batch.U_PAC or vars.U_PAC
local U_TOF_UL_Defl = batch.U_TOF_UL_Defl or vars.U_TOF_UL_Defl
local U_Van = batch.U_Van or vars.U_Van
local U_Vld = batch.U_Vld or vars.U_Vld
local U_Vth = batch.U_Vth or vars.U_Vth
local U_Vud = batch.U_Vud or vars.U_Vud
local x_center = batch.x_center or vars.x_center
local xy_ap_in_radius = batch.xy_ap_in_radius or
   ↪ vars.xy_ap_in_radius
local xy_ap_mid_radius = batch.xy_ap_mid_radius or
   ↪ vars.xy_ap_mid_radius
local xy_ap_out_radius = batch.xy_ap_out_radius or
   ↪ vars.xy_ap_out_radius
local z_ap_height = batch.z_ap_height or vars.z_ap_height
local z_ap_mid = batch.z_ap_mid or vars.z_ap_mid
local z_grid = batch.z_grid or vars.z_grid
local z_cf = batch.z_cf or vars.z_cf
local z_ssd = batch.z_ssd or vars.z_ssd
local z_stop_mcp = batch.z_stop_mcp or vars.z_stop_mcp
local z_start_mcp = batch.z_start_mcp or vars.z_start_mcp
local esa_exit_z = batch.esa_exit_z or vars.esa_exit_z
local y_center = batch.y_center or vars.y_center
local ssd_r_min = batch.ssd_r_min or vars.ssd_r_min
local ssd_r_max = batch.ssd_r_max or vars.ssd_r_max
local ssd_az_min = batch.ssd_az_min or vars.ssd_az_min
local ssd_az_max = batch.ssd_az_max or vars.ssd_az_max
local mcp_r_min = batch.mcp_r_min or vars.mcp_r_min
local mcp_r_max = batch.mcp_r_max or vars.mcp_r_max
local mcp_az_min = batch.mcp_az_min or vars.mcp_az_min
local mcp_az_max = batch.mcp_az_max or vars.mcp_az_max
local esa_exit_r_min = batch.esa_exit_r_min or
   ↪ vars.esa_exit_r_min
local esa_exit_r_max = batch.esa_exit_r_max or
   ↪ vars.esa_exit_r_max
local esa_exit_az_min = batch.esa_exit_az_min or
   ↪ vars.esa_exit_az_min
local esa_exit_az_max = batch.esa_exit_az_max or
   ↪ vars.esa_exit_az_max
local cf_r_min = batch.cf_r_min or vars.cf_r_min
local cf_r_max = batch.cf_r_max or vars.cf_r_max
local cf_az_min = batch.cf_az_min or vars.cf_az_min
local cf_az_max = batch.cf_az_max or vars.cf_az_max
local lab_volt_opt = batch.lab_volt_opt or vars.lab_volt_opt
local heat_shield_ap_min_x = batch.heat_shield_ap_min_x or
   ↪ vars.heat_shield_ap_min_x
local heat_shield_ap_max_x = batch.heat_shield_ap_max_x or
   ↪ vars.heat_shield_ap_max_x
local heat_shield_ap_min_y = batch.heat_shield_ap_min_y or
   ↪ vars.heat_shield_ap_min_y
local heat_shield_ap_max_y = batch.heat_shield_ap_max_y or
   ↪ vars.heat_shield_ap_max_y
```

```
local heat_shield_ap_min_z = batch.heat_shield_ap_min_z or
   ↪ vars.heat_shield_ap_min_z
local heat_shield_ap_max_z = batch.heat_shield_ap_max_z or
   ↪ vars.heat_shield_ap_max_z
local simulation_min_y = batch.simulation_min_y or
   ↪ vars.simulation_min_y
local simulation_max_y = batch.simulation_max_y or
   ↪ vars.simulation_max_y
local simulation_min_z = batch.simulation_min_z or
   ↪ vars.simulation_min_z
local simulation_max_z = batch.simulation_max_z or
   ↪ vars.simulation_max_z
local tof_extent_x = batch.tof_extent_x or vars.tof_extent_x
local tof_extent_y = batch.tof_extent_y or vars.tof_extent_y
local model_res = batch.model_res or vars.model_res
local sim_D = batch.sim_D or vars.sim_D
local sim_T = batch.sim_T or vars.sim_T
local sim_k = batch.sim_k or vars.sim_k


-- This describes a flat plane source surface with a uniform
   ↪ distribution in all dimensions.
      -- The size of the source surface in both elevation and
         ↪ azimuth is set to cover the aperture based on
         ↪ particle trajectories

local ssx_min = source_surface_x - heat_shield_ap_max_x
local ssx_max = source_surface_x - heat_shield_ap_min_x
local x_min = -1
local x_max = -1

if tr_az_dist == 1 then -- Normal distribution in azimuth
      if tr_az_mean < 0 then
            x_min = ssx_min
            x_max = ssx_max
      else
```

```
            x_min = ssx_max
            x_max = ssx_min
      end
      y_min = math.max(heat_shield_ap_min_y-x_min*math.tan(rad(
         ↪ tr_az_mean+(3*tr_az_stdev))),simulation_min_y) --3
         ↪ stdev plus extra for acceptance assurance
      y_max = math.min(heat_shield_ap_max_y-x_max*math.tan(rad(
         ↪ tr_az_mean-(3*tr_az_stdev))),simulation_max_y) --3
         ↪ stdev plus extra for acceptance assurance
else -- Uniform or arithmetic distribution in azimuth
      if tr_az_max < 0 then
            x_min = ssx_min
      else
            x_min = ssx_max
      end
      if tr_az_min < 0 then
            x_max = ssx_max
      else
            x_max = ssx_min
      end
      y_min = math.max(heat_shield_ap_min_y-x_min*math.tan(rad(
         ↪ tr_az_max)),simulation_min_y) --extra amount for
         ↪ acceptance assurance
      y_max = math.min(heat_shield_ap_max_y-x_max*math.tan(rad(
         ↪ tr_az_min)),simulation_max_y) --extra amount for
         ↪ acceptance assurance
end

if tr_el_dist == 1 then -- Normal distribution in elevation
      if tr_el_mean < 0 then
            x_min = ssx_max
            x_max = ssx_min
      else
            x_min = ssx_min
            x_max = ssx_max
      end
      z_min = math.max(heat_shield_ap_min_z+x_min*math.tan(rad(
         ↪ tr_el_mean-(3*tr_el_stdev))),simulation_min_z) --3
         ↪ stdev plus extra for acceptance assurance
      z_max = math.min(heat_shield_ap_max_z+x_max*math.tan(rad(
         ↪ tr_el_mean+(3*tr_el_stdev))),simulation_max_z) --3
         ↪ stdev plus extra for acceptance assurance
else -- Uniform or arithmetic distribution in elevation
      if tr_el_max < 0 then
            x_min = ssx_max
      else
            x_min = ssx_min
      end
      if tr_el_min < 0 then
            x_max = ssx_min
      else
            x_max = ssx_max
      end
      z_min = math.max(heat_shield_ap_min_z+x_min*math.tan(rad(
         ↪ tr_el_min)),simulation_min_z) --extra amount for
         ↪ acceptance assurance
      z_max = math.min(heat_shield_ap_max_z+x_max*math.tan(rad(
         ↪ tr_el_max)),simulation_max_z) --extra amount for
         ↪ acceptance assurance
end

-- Reduce size down to 10% for a point source (style 2)
if source_surface_style == 2 then
      y_ext = (y_max-y_min)*0.1
      y_mid = (y_max+y_min)/2
      y_max = y_mid+y_ext
      y_min = y_mid-y_ext
      z_ext = (z_max-z_min)*0.1
      z_mid = (z_max+z_min)/2
      z_max = z_mid+z_ext
      z_min = z_mid-z_ext
end
```

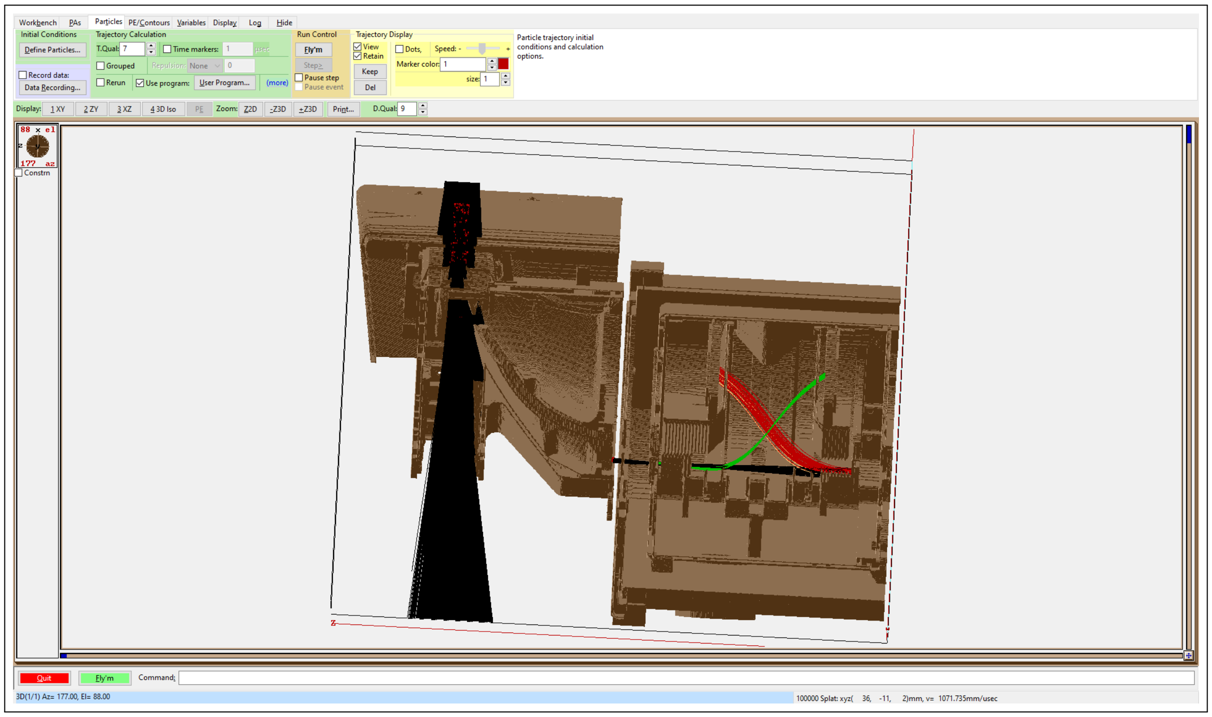
```
local A = (y_max-y_min)*(z_min-z_max)/100 --cm2 -- Source surface
   ↪  area
local total_n = flux
--local beams = {}

local number_of_y_bins = 1000.0 -- Adjust for greater resolution
   ↪ in Y.
number_of_y_bins = min(number_of_y_bins, total_n) -- To have
   ↪ fewer particles possible
local particles_per_bin = ceil(total_n/number_of_y_bins)

--for k=1,number_of_y_bins do -- Step through the z bins
   ↪ sequentially, creating a particle group for each one (this
   ↪ is the "particles(beams)" command below)

      -- SIMION only considers azimuth and elevation in a
         ↪ specific plane (see Manual figure L.2) which are
         ↪ opposite to our definition of azimuth and elevation
         ↪ for HIS.
      --  this azimuth distribution will be saved to SIMION's
         ↪ elevation variable
      if tr_az_dist == 0 then -- Uniform Distribution
            az_dist = uniform_distribution {min = tr_az_min,max =
               ↪  tr_az_max}
      elseif tr_az_dist == 1 then -- Normal Distribution
            az_dist = gaussian_distribution{mean=tr_az_mean,stdev
               ↪ =tr_az_stdev}
      elseif tr_az_dist == 2 then -- Arithmetic Distribution
            az_dist = arithmetic_sequence {first = tr_az_min,last
               ↪  = tr_az_max,n=tr_az_steps}
      end

      -- SIMION only considers azimuth and elevation in a
         ↪ specific plane (see Manual figure L.2) which are
         ↪ opposite to our definition of azimuth and elevation
         ↪ for HIS.
      --  this elevation distribution will be saved to SIMION's
         ↪ azimuth variable, but also with a 180 degree shift
         ↪ due to the instrument's orientation (180 minus
         ↪ desired angle)
      local shift = 180
      if tr_el_dist == 0 then -- Uniform Distribution
            el_dist = uniform_distribution {min = shift-tr_el_min
               ↪ ,max = shift-tr_el_max}
      elseif tr_el_dist == 1 then -- Normal Distribution
            el_dist = gaussian_distribution{mean=shift-tr_el_mean
               ↪ ,stdev=tr_el_stdev}
      elseif tr_el_dist == 2 then -- Arithmetic Distribution
            el_dist = arithmetic_sequence {first = shift-
               ↪ tr_el_min,last = shift-tr_el_max,n=tr_el_steps}
      end

      beams={standard_beam {
            --n=particles_per_bin,
            n=total_n,
            tob = 0,
            mass = m,
            charge = q,
            cwf = 1,
            color = 0,

            x = source_surface_x,
            z = uniform_distribution {min=z_min,max=z_max},
            y = uniform_distribution {min=y_min,max=y_max},

            az = el_dist, -- SIMION considers azimuth to be what
               ↪ we consider to be elevation (and currently with
               ↪  a 180 degree shift)
            el = az_dist -- SIMION considers elevation to be what
               ↪  we consider to be azimuth
      }}
```

```
      particles(beams)
--end


vars.aberration = false
vars.az_fov_max = false
vars.az_fov_min = false
vars.az_fov_spacing = false
vars.cfoil_neutralize = false
vars.cfoil_thickness = false
vars.cfoil_transmission = false
vars.dwell = false
vars.flux = false
vars.ke_dist = false
vars.ke_max = false
vars.ke_mean = false
vars.ke_min = false
vars.ke_stdev = false
vars.m = false
vars.q = false
vars.simulation_batch_mode = false
vars.simulation_eais_only = false
vars.simulation_k = false
vars.simulation_mode = false
vars.simulation_sec_e_total = false
vars.simulation_sec_p_total = false
vars.simulation_tr_qual = false
vars.source_surface_style = false
vars.source_surface_shape = false
vars.source_surface_x = false
vars.simulation_view_retain = false
vars.species = false
vars.tr_az_cover_the_spread = false
vars.tr_az_dist = false
vars.tr_az_max = false
vars.tr_az_mean = false
vars.tr_az_min = false
vars.tr_az_stdev = false
vars.tr_az_steps = false
vars.tr_el_dist = false
vars.tr_el_max = false
vars.tr_el_mean = false
vars.tr_el_min = false
vars.tr_el_stdev = false
vars.tr_el_steps = false
vars.U_Ground = false
vars.U_MCP_Stop = false
vars.U_MCP_Start = false
vars.U_MCP_Case = false
vars.U_PAC = false
vars.U_TOF_UL_Defl = false
vars.U_Van = false
vars.U_Vld = false
vars.U_Vth = false
vars.U_Vud = false
vars.x_center = false
vars.xy_ap_in_radius = false
vars.xy_ap_mid_radius = false
vars.xy_ap_out_radius = false
vars.z_ap_height = false
vars.z_ap_mid = false
vars.z_grid = false
vars.z_cf = false
vars.z_ssd = false
vars.z_start_mcp = false
vars.z_stop_mcp = false
vars.esa_exit_z = false
vars.y_center = false
vars.ssd_r_min = false
vars.ssd_r_max = false
vars.ssd_az_min = false
vars.ssd_az_max = false
vars.mcp_r_min = false
vars.mcp_r_max = false
vars.mcp_az_min = false
vars.mcp_az_max = false
vars.esa_exit_r_min = false
vars.esa_exit_r_max = false
vars.esa_exit_az_min = false
vars.esa_exit_az_max = false
vars.cf_r_min = false
vars.cf_r_max = false
vars.cf_az_min = false
vars.cf_az_max = false
vars.lab_volt_opt = false
vars.heat_shield_ap_min_x = false
vars.heat_shield_ap_max_x = false
vars.heat_shield_ap_min_y = false
vars.heat_shield_ap_max_y = false
vars.heat_shield_ap_min_z = false
vars.heat_shield_ap_max_z = false
vars.simulation_min_y = false
vars.simulation_max_y = false
vars.simulation_min_z = false
vars.simulation_max_z = false
vars.tof_extent_x = false
vars.tof_extent_y = false
vars.model_res = false
vars.sim_D = false
vars.sim_T = false
vars.sim_k = false

batch.aberration = false
batch.az_fov_max = false
batch.az_fov_min = false
batch.az_fov_spacing = false
batch.cfoil_neutralize = false
batch.cfoil_thickness = false
batch.cfoil_transmission = false
batch.dwell = false
batch.flux = false
batch.ke_dist = false
batch.ke_max = false
batch.ke_mean = false
batch.ke_min = false
batch.ke_stdev = false
batch.m = false
batch.q = false
batch.simulation_batch_mode = false
batch.simulation_eais_only = false
batch.simulation_k = false
batch.simulation_mode = false
batch.simulation_sec_e_total = false
batch.simulation_sec_p_total = false
batch.simulation_tr_qual = false
batch.source_surface_style = false
batch.source_surface_shape = false
batch.source_surface_x = false
batch.simulation_view_retain = false
batch.species = false
batch.tr_az_cover_the_spread = false
batch.tr_az_dist = false
batch.tr_az_max = false
batch.tr_az_mean = false
batch.tr_az_min = false
batch.tr_az_stdev = false
batch.tr_az_steps = false
batch.tr_el_dist = false
batch.tr_el_max = false
batch.tr_el_mean = false
batch.tr_el_min = false
batch.tr_el_stdev = false
batch.tr_el_steps = false
batch.U_Ground = false
batch.U_MCP_Stop = false
batch.U_MCP_Start = false
batch.U_MCP_Case = false
batch.U_PAC = false
batch.U_TOF_UL_Defl = false
batch.U_Van = false
batch.U_Vld = false
batch.U_Vth = false
batch.U_Vud = false
batch.x_center = false
batch.xy_ap_in_radius = false
batch.xy_ap_min_radius = false
batch.xy_ap_out_radius = false
batch.z_ap_height = false
batch.z_ap_mid = false
batch.z_grid = false
batch.z_cf = false
batch.z_ssd = false
batch.z_start_mcp = false
batch.z_stop_mcp = false
batch.esa_exit_z = false
batch.y_center = false
batch.ssd_r_min = false
batch.ssd_r_max = false
batch.ssd_az_min = false
batch.ssd_az_max = false
batch.mcp_r_min = false
batch.mcp_r_max = false
batch.mcp_az_min = false
batch.mcp_az_max = false
batch.esa_exit_r_min = false
batch.esa_exit_r_max = false
batch.esa_exit_az_min = false
batch.esa_exit_az_max = false
batch.cf_r_min = false
batch.cf_r_max = false
batch.cf_az_min = false
batch.cf_az_max = false
batch.lab_volt_opt = false
batch.heat_shield_ap_min_x = false
batch.heat_shield_ap_max_x = false
batch.heat_shield_ap_min_y = false
batch.heat_shield_ap_max_y = false
batch.heat_shield_ap_min_z = false
batch.heat_shield_ap_max_z = false
batch.simulation_min_y = false
batch.simulation_max_y = false
batch.simulation_min_z = false
batch.simulation_max_z = false
batch.tof_extent_x = false
batch.tof_extent_y = false
batch.model_res = false
batch.sim_D = false
batch.sim_T = false
batch.sim_k = false
```

APPENDIX E

SIMION Lua Code

Details of the SIMION Monte Carlo simulation are discussed in Chapter 2. Figure E.1 demonstrates a run in SIMION and Figure E.2 shows a labeled schematic of the ion optical model (IOM).



T.Qual: 7 · Time markers: 1 µsec · Repulsion: None 0 · Use program: ✓ · View ✓ · Retain ✓ · Dots, · Pause step · Pause event · Marker color: 1 · size: 1 · D.Qual: 9

3D(1/1) Az= 177.00, El= 88.00

100000 Splat: xyz(36, -11, 2)mm, v= 1071.735mm/usec

Figure E.1: A SIMION run with the SO-HIS instrument viewed in an exploded 3D-isometric view, with the instrument shown mostly in the z-x plane. Particles start at a flat SS outside the heat shield aperture. Initial particle trajectories are black, start electron tracks are green, stop electron tracks are red, and tracks of electrons that miss a detector are yellow.

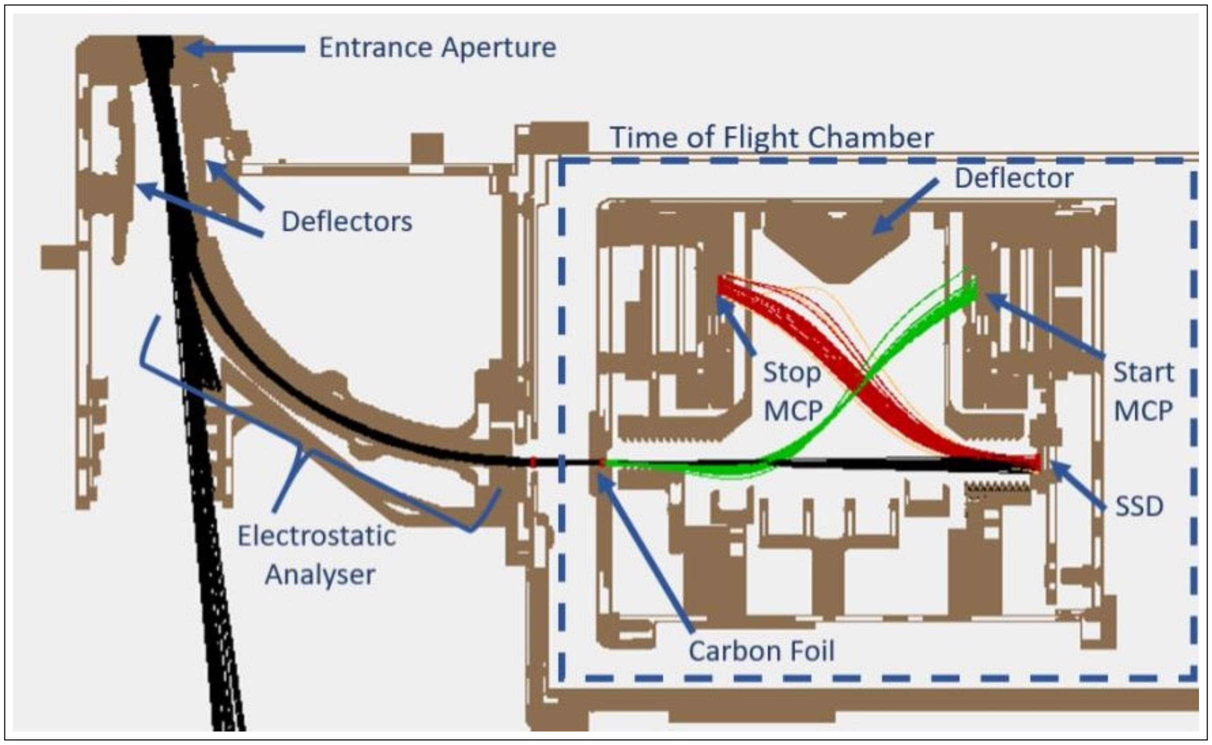


Figure E.2: A labeled schematic of the SO-HIS SIMION IOM in the z-x plane.

The SIMION LUA code used to define the Monte Carlo simulation parameters for SIMION is included below. This code was contributed to by: Jason Gilbert, Sarah Spitzer, Austin Glass, Matthew Panning, Connor Raines, and Mark Stakhiv.

Listing E.1: LUA code used to define the SIMION Monte Carlo Simulation.

```
-- Associated .rec file settings
-- Data to record: Ion n, TOF, Mass, Charge, X, Y, Z, Azm, Elv,
   ↪ KE
-- When to record data: Ion's Start, All Markers
-- Format: Delimited, G, After Each Fly'm
dofile("SimParameters.lua")

batch = {}

if vars.simulation_batch_mode == 1 then
  dofile("SimParametersBatch.lua")
```

```
end

local aberration = batch.aberration or vars.aberration
local az_fov_max = batch.az_fov_max or vars.az_fov_max
local az_fov_min = batch.az_fov_min or vars.az_fov_min
local az_fov_spacing = batch.az_fov_spacing or
   ↪ vars.az_fov_spacing
local cfoil_neutralize = batch.cfoil_neutralize or
   ↪ vars.cfoil_neutralize
local cfoil_thickness = batch.cfoil_thickness or
   ↪ vars.cfoil_thickness
local cfoil_transmission = batch.cfoil_transmission or
   ↪ vars.cfoil_transmission
local dwell = batch.dwell or vars.dwell
local flux = batch.flux or vars.flux
local ke_dist = batch.ke_dist or vars.ke_dist
local ke_max = batch.ke_max or vars.ke_max
local ke_mean = batch.ke_mean or vars.ke_mean
local ke_min = batch.ke_min or vars.ke_min
local ke_stdev = batch.ke_stdev or vars.ke_stdev
local m = batch.m or vars.m
local q = batch.q or vars.q
local simulation_batch_mode = batch.simulation_batch_mode or
   ↪ vars.simulation_batch_mode
local simulation_eais_only = batch.simulation_eais_only or
   ↪ vars.simulation_eais_only
local simulation_k = batch.simulation_k or vars.simulation_k
local simulation_mode = batch.simulation_mode or
   ↪ vars.simulation_mode
local simulation_sec_e_total = batch.simulation_sec_e_total or
   ↪ vars.simulation_sec_e_total
local simulation_sec_p_total = batch.simulation_sec_p_total or
   ↪ vars.simulation_sec_p_total
local simulation_tr_qual = batch.simulation_tr_qual or
   ↪ vars.simulation_tr_qual
local source_surface_style = batch.source_surface_style or
```

```
    ↪ vars.source_surface_style
local source_surface_shape = batch.source_surface_shape or
    ↪ vars.source_surface_shape
local source_surface_x = batch.source_surface_x or
    ↪ vars.source_surface_x
local simulation_view_retain = batch.simulation_view_retain or
    ↪ vars.simulation_view_retain
local species = batch.species or vars.species
local tr_az_cover_the_spread = batch.tr_az_cover_the_spread or
    ↪ vars.tr_az_cover_the_spread
local tr_az_dist = batch.tr_az_dist or vars.tr_az_dist
local tr_az_max = batch.tr_az_max or vars.tr_az_max
local tr_az_mean = batch.tr_az_mean or vars.tr_az_mean
local tr_az_min = batch.tr_az_min or vars.tr_az_min
local tr_az_stdev = batch.tr_az_stdev or vars.tr_az_stdev
local tr_az_steps = batch.tr_az_steps or vars.tr_az_steps
local tr_el_dist = batch.tr_el_dist or vars.tr_el_dist
local tr_el_max = batch.tr_el_max or vars.tr_el_max
local tr_el_mean = batch.tr_el_mean or vars.tr_el_mean
local tr_el_min = batch.tr_el_min or vars.tr_el_min
local tr_el_stdev = batch.tr_el_stdev or vars.tr_el_stdev
local tr_el_steps = batch.tr_el_steps or vars.tr_el_steps
local U_Ground = batch.U_Ground or vars.U_Ground
local U_MCP_Stop = batch.U_MCP_Stop or vars.U_MCP_Stop
local U_MCP_Start = batch.U_MCP_Start or vars.U_MCP_Start
local U_MCP_Case = batch.U_MCP_Case or vars.U_MCP_Case
local U_PAC = batch.U_PAC or vars.U_PAC
local U_TOF_UL_Defl = batch.U_TOF_UL_Defl or vars.U_TOF_UL_Defl
local U_Van = batch.U_Van or vars.U_Van
local U_Vld = batch.U_Vld or vars.U_Vld
local U_Vth = batch.U_Vth or vars.U_Vth
local U_Vud = batch.U_Vud or vars.U_Vud
local x_center = batch.x_center or vars.x_center
local xy_ap_in_radius = batch.xy_ap_in_radius or
    ↪ vars.xy_ap_in_radius
local xy_ap_mid_radius = batch.xy_ap_mid_radius or
```

```
    ↪ vars.xy_ap_mid_radius
local xy_ap_out_radius = batch.xy_ap_out_radius or
    ↪ vars.xy_ap_out_radius
local z_ap_height = batch.z_ap_height or vars.z_ap_height
local z_ap_mid = batch.z_ap_mid or vars.z_ap_mid
local z_grid = batch.z_grid or vars.z_grid
local z_cf = batch.z_cf or vars.z_cf
local z_ssd = batch.z_ssd or vars.z_ssd
local z_stop_mcp = batch.z_stop_mcp or vars.z_stop_mcp
local z_start_mcp = batch.z_start_mcp or vars.z_start_mcp
local esa_exit_z = batch.esa_exit_z or vars.esa_exit_z
local y_center = batch.y_center or vars.y_center
local ssd_r_min = batch.ssd_r_min or vars.ssd_r_min
local ssd_r_max = batch.ssd_r_max or vars.ssd_r_max
local ssd_az_min = batch.ssd_az_min or vars.ssd_az_min
local ssd_az_max = batch.ssd_az_max or vars.ssd_az_max
local mcp_r_min = batch.mcp_r_min or vars.mcp_r_min
local mcp_r_max = batch.mcp_r_max or vars.mcp_r_max
local mcp_az_min = batch.mcp_az_min or vars.mcp_az_min
local mcp_az_max = batch.mcp_az_max or vars.mcp_az_max
local esa_exit_r_min = batch.esa_exit_r_min or
    ↪ vars.esa_exit_r_min
local esa_exit_r_max = batch.esa_exit_r_max or
    ↪ vars.esa_exit_r_max
local esa_exit_az_min = batch.esa_exit_az_min or
    ↪ vars.esa_exit_az_min
local esa_exit_az_max = batch.esa_exit_az_max or
    ↪ vars.esa_exit_az_max
local cf_r_min = batch.cf_r_min or vars.cf_r_min
local cf_r_max = batch.cf_r_max or vars.cf_r_max
local cf_az_min = batch.cf_az_min or vars.cf_az_min
local cf_az_max = batch.cf_az_max or vars.cf_az_max
local lab_volt_opt = batch.lab_volt_opt or vars.lab_volt_opt
local heat_shield_ap_min_x = batch.heat_shield_ap_min_x or
    ↪ vars.heat_shield_ap_min_x
local heat_shield_ap_max_x = batch.heat_shield_ap_max_x or
```

```
   ↪ vars.heat_shield_ap_max_x
local heat_shield_ap_min_y = batch.heat_shield_ap_min_y or
   ↪ vars.heat_shield_ap_min_y
local heat_shield_ap_max_y = batch.heat_shield_ap_max_y or
   ↪ vars.heat_shield_ap_max_y
local heat_shield_ap_min_z = batch.heat_shield_ap_min_z or
   ↪ vars.heat_shield_ap_min_z
local heat_shield_ap_max_z = batch.heat_shield_ap_max_z or
   ↪ vars.heat_shield_ap_max_z
local simulation_min_y = batch.simulation_min_y or
   ↪ vars.simulation_min_y
local simulation_max_y = batch.simulation_max_y or
   ↪ vars.simulation_max_y
local simulation_min_z = batch.simulation_min_z or
   ↪ vars.simulation_min_z
local simulation_max_z = batch.simulation_max_z or
   ↪ vars.simulation_max_z
local tof_extent_x = batch.tof_extent_x or vars.tof_extent_x
local tof_extent_y = batch.tof_extent_y or vars.tof_extent_y
local model_res = batch.model_res or vars.model_res
local sim_D = batch.sim_D or vars.sim_D
local sim_T = batch.sim_T or vars.sim_T
local sim_k = batch.sim_k or vars.sim_k

simion.workbench_program()

-- allow batch mode override of these variables, if available,
   ↪ otherwise use values above
--local potentials_changed = 0

function segment.load()
  -- For improved speed, set the trajectory computational quality
     ↪  (T.Qual) to 0.
  sim_trajectory_quality = simulation_tr_qual
end
```

```
 function segment.init_p_values()
 pa = simion.pas[1]
 pa:fast_adjust {[1] = U_Vud,[2] = U_Vld,[3] = U_Vth,[4] = U_Van
    ↪ ,[5] = U_PAC,[6] = U_MCP_Case,[7] = U_MCP_Stop,[8] =
    ↪ U_MCP_Start,[9] = U_TOF_UL_Defl}
 end

function segment.initialize() -- dynamically change an ion's
   ↪ initial parameters and conditions

-- the following are flags for recorded events
mark_aperture = 0
mark_esa_exit = 0
mark_pre_foil = 0
foil_hit = 0
mark_post_foil = 0
start_mcp = 0
ssd = 0
stop_mcp = 0

-- measurement tolerances
tol_x = model_res
tol_z = model_res
tol_r = sqrt(tol_x^2 + tol_z^2)
tol_theta_mcp = 2 * asin((sqrt(2)/8)/mcp_r_min) * 180/math.pi
tol_theta_ssd = 2 * asin((sqrt(2)/8)/ssd_r_min) * 180/math.pi
tol_theta_ap = 2 * asin((sqrt(2)/8)/xy_ap_in_radius) * 180/
   ↪ math.pi

-- particle energies and initial conditions
ke=0
      while (ke <= 0) do

            if ke_dist == 1 then
                  ke = ke_mean+ke_stdev/0.7*sqrt(-ln(rand()+0
                     ↪ .000001))*cos(2*3.14159265*rand())
            else
                  ke = ke_min + rand()*(ke_max - ke_min)
            end
            speed, az, el = rect3d_to_polar3d(ion_vx_mm,ion_vy_mm
               ↪ ,ion_vz_mm)
            speed = ke_to_speed(ke,ion_mass)
            ion_vx_mm, ion_vy_mm, ion_vz_mm = polar3d_to_rect3d(
               ↪ speed,az,el)

      end
end

function segment.other_actions() -- called after each time step

      sim_trajectory_image_control = simulation_view_retain

      -- specifies if the simulation should match the lab
         ↪ settings
      -- ie No carbon foil, all voltages in TOF chamber equal -25
         ↪ kV
      --  Should also be used with no neutralization at CF and
      --  no secondary electrons
      -- flag to turn off carbon foil scattering / loss
      if lab_volt_opt == 1 then
            carbon_foil_off = 1 -- set to 0 to retain carbon foil
               ↪ , 1 to remove scattering/loss
      else
            carbon_foil_off = 0
      end

      -- ion limits from ESA exit through SSD
      tof_x_min = x_center-tof_extent_x
      tof_x_max = x_center
      tof_y_min = y_center-tof_extent_y/2
      tof_y_max = y_center+tof_extent_y/2
```

```
-- initialize some flags
if ion_pz_mm > heat_shield_ap_min_z then
      if ssd == 1 then ssd = 0 end
      if start_mcp == 1 then start_mcp = 0 end
      if stop_mcp == 1 then stop_mcp = 0 end
      if mark_pre_foil == 1 then mark_pre_foil = 0 end
      if foil_hit == 1 then foil_hit = 0 end
      if mark_post_foil == 1 then mark_post_foil = 0 end
      if mark_esa_exit == 1 then mark_esa_exit = 0 end
      sec_e = 0
      sec_p = 0
end
if ion_px_mm >= source_surface_x-tol_x then
      if mark_aperture == 1 then mark_aperture = 0 end
end

-- esa exit mark
if ion_pz_mm <= esa_exit_z and ion_px_mm > tof_x_min and
   ↪ ion_px_mm < tof_x_max and ion_py_mm > tof_y_min and
   ↪ ion_py_mm < tof_y_max and mark_esa_exit == 0 then
      mark()
      mark_esa_exit = 1
end

-- carbon foil
if ion_pz_mm < z_cf and mark_pre_foil == 1 then -- carbon
   ↪ foil impact
      if foil_hit == 1 and mark_post_foil == 0 then -- mark
         ↪  one time step after changes
            mark() -- mark cf post
            mark_post_foil = 1
      end

      if foil_hit == 0 then
            foil_hit = 1
            cf_px_mm = ion_px_mm -- record carbon foil
   ↪ impact position and trajectory
cf_py_mm = ion_py_mm -- to use later for start
   ↪ electron genesis
cf_pz_mm = ion_pz_mm
cf_vx_mm = ion_vx_mm
cf_vy_mm = ion_vy_mm
cf_vz_mm = ion_vz_mm

set_color('black')
cf_speed, cf_az, cf_el = rect3d_to_polar3d(
   ↪ ion_vx_mm,ion_vy_mm,ion_vz_mm)

kinetic_energy = speed_to_ke(cf_speed, ion_mass
   ↪ )
incident_ke = kinetic_energy
if not carbon_foil_off then
      ke_loss = cfoil_loss(ion_mass,
         ↪ kinetic_energy/1000,
         ↪ cfoil_thickness) -- function takes
         ↪ amu, keV, ug/cm2
      scatter_amt = cfoil_scatter(0,ion_mass,
         ↪ incident_ke/1000,cfoil_thickness)
else
      ke_loss = 0
      scatter_amt = 0
end
kinetic_energy = kinetic_energy - 1000*ke_loss
cf_speed = ke_to_speed(kinetic_energy,ion_mass)
   ↪  -- (from eV to mm/usec)
ion_vx_mm, ion_vy_mm, ion_vz_mm =
   ↪ polar3d_to_rect3d(cf_speed,cf_az,cf_el+
   ↪ scatter_amt)

if cfoil_neutralize == 1 then
      ion_charge = 0
end
```

```
      end
end

r_pos = math.sqrt((ion_px_mm-x_center)^2+(ion_py_mm-
   ↪ y_center)^2)
az_pos = -math.atan((ion_py_mm-y_center)/(ion_px_mm-
   ↪ x_center)) * (180/math.pi)

-- other polar limits
ssd_r_pos_min = ssd_r_min-tol_r -- 67.84
ssd_r_pos_max = ssd_r_max+tol_r -- 72.16
ssd_az_pos_min = ssd_az_min-tol_theta_ssd -- -29
ssd_az_pos_max = ssd_az_max+tol_theta_ssd -- 65
mcp_r_pos_min = mcp_r_min-tol_r -- 22.5
mcp_r_pos_max = mcp_r_max+tol_r -- 38.0
mcp_az_pos_min = mcp_az_min-tol_theta_mcp -- -32
mcp_az_pos_max = mcp_az_max+tol_theta_mcp -- 68

-- entrance aperture limits
aperture_z_min = z_ap_mid-z_ap_height-tol_z
aperture_z_max = z_ap_mid+z_ap_height+tol_z
aperture_out_r = xy_ap_out_radius+tol_r
aperture_mid_r = xy_ap_mid_radius+tol_r
aperture_in_r = xy_ap_in_radius+tol_r
aperture_az_min = az_fov_min-tol_theta_ap
aperture_az_max = az_fov_max+tol_theta_ap

-- entrance aperture mark
if mark_aperture == 0 and r_pos <= aperture_out_r and
   ↪ az_pos >= aperture_az_min and az_pos <=
   ↪ aperture_az_max and ion_pz_mm >= aperture_z_min and
   ↪ ion_pz_mm <= aperture_z_max then
      mark()
      mark_aperture = 1
end
```

```
-- tof
if ion_pz_mm < z_cf and ion_px_mm > tof_x_min and ion_px_mm
   ↪  < tof_x_max and ion_py_mm > tof_y_min and ion_py_mm
   ↪ < tof_y_max then
      if foil_hit == 0 and mark_pre_foil == 0 then
            mark() -- mark cf pre
            if simulation_eais_only == 1 then
                  ion_splat = -4
            end
            mark_pre_foil = 1
      end


      if ion_splat == -1 then

            -- start mcp
            if start_mcp == 0 and r_pos > mcp_r_pos_min and
               ↪  r_pos < mcp_r_pos_max and az_pos >
               ↪ mcp_az_pos_min and az_pos <
               ↪ mcp_az_pos_max and ion_pz_mm <
               ↪ z_start_mcp + tol_z and ion_pz_mm >
               ↪ z_start_mcp - (2*tol_z) then
                  start_px_mm = ion_px_mm -- record start
                     ↪ mcp impact position and trajectory
                  start_py_mm = ion_py_mm -- to use later
                     ↪ for proton ejection
                  start_pz_mm = ion_pz_mm
                  start_vx_mm = ion_vx_mm
                  start_vy_mm = ion_vy_mm
                  start_vz_mm = ion_vz_mm
                  mark()

                  start_mcp = 1
            end
```

```
-- stop mcp
if stop_mcp == 0 and r_pos > mcp_r_pos_min and
    ↪ r_pos < mcp_r_pos_max and az_pos >
    ↪ mcp_az_pos_min and az_pos <
    ↪ mcp_az_pos_max and ion_pz_mm > z_stop_mcp
    ↪  - tol_z and ion_pz_mm < z_stop_mcp + (2*
    ↪ tol_z) then

      stop_px_mm = ion_px_mm -- record stop mcp
         ↪  impact position and trajectory
      stop_py_mm = ion_py_mm -- to use later
         ↪ for proton ejection
      stop_pz_mm = ion_pz_mm
      stop_vx_mm = ion_vx_mm
      stop_vy_mm = ion_vy_mm
      stop_vz_mm = ion_vz_mm
      mark()

      stop_mcp = 1
end

-- ssd
if ssd == 0 and r_pos > ssd_r_pos_min and r_pos
    ↪  < ssd_r_pos_max and az_pos >
    ↪ ssd_az_pos_min and az_pos <
    ↪ ssd_az_pos_max and ion_pz_mm < z_ssd +
    ↪ tol_z and ion_pz_mm > z_ssd - (2*tol_z)
    ↪ then
      mark()
      ssd = 1
end

sec_e, sec_p = splat_logic(sec_e,
    ↪ simulation_sec_e_total,sec_p,
    ↪ simulation_sec_p_total,
            ssd, start_mcp, stop_mcp,
                                        cf_px_mm,cf_py_mm,cf_pz_mm,
                                        cf_vx_mm,cf_vy_mm,cf_vz_mm,
                                        start_px_mm,start_py_mm,start_pz_mm
                                           ↪ ,
                                        start_vx_mm,start_vy_mm,start_vz_mm
                                           ↪ ,
                                        stop_px_mm,stop_py_mm,stop_pz_mm,
                                        stop_vx_mm,stop_vy_mm,stop_vz_mm,
                                        simulation_view_retain)

                  end
            end
      end

function segment.flym()
  reload_fly2('HIS.fly2', {})
  run()
end

function segment.terminate()

vars.aberration = false
vars.az_fov_max = false
vars.az_fov_min = false
vars.az_fov_spacing = false
vars.cfoil_neutralize = false
vars.cfoil_thickness = false
vars.cfoil_transmission = false
vars.dwell = false
vars.flux = false
vars.ke_dist = false
vars.ke_max = false
vars.ke_mean = false
vars.ke_min = false
vars.ke_stdev = false
vars.m = false
vars.q = false
vars.simulation_batch_mode = false
vars.simulation_eais_only = false
vars.simulation_k = false
vars.simulation_mode = false
vars.simulation_sec_e_total = false
vars.simulation_sec_p_total = false
vars.simulation_tr_qual = false
vars.source_surface_style = false
vars.source_surface_shape = false
vars.source_surface_x = false
vars.simulation_view_retain = false
vars.species = false
vars.tr_az_cover_the_spread = false
vars.tr_az_dist = false
vars.tr_az_max = false
vars.tr_az_mean = false
vars.tr_az_min = false
vars.tr_az_stdev = false
vars.tr_az_steps = false
vars.tr_el_dist = false
vars.tr_el_max = false
vars.tr_el_mean = false
vars.tr_el_min = false
vars.tr_el_stdev = false
vars.tr_el_steps = false
vars.U_Ground = false
vars.U_MCP_Stop = false
vars.U_MCP_Start = false
vars.U_MCP_Case = false
vars.U_PAC = false
vars.U_TOF_UL_Defl = false
vars.U_Van = false
vars.U_Vld = false
vars.U_Vth = false
vars.U_Vud = false
vars.x_center = false
vars.xy_ap_in_radius = false
vars.xy_ap_min_radius = false
vars.xy_ap_out_radius = false
vars.z_ap_height = false
vars.z_ap_mid = false
vars.z_grid = false
vars.z_cf = false
vars.z_ssd = false
vars.z_stop_mcp = false
vars.z_start_mcp = false
vars.esa_exit_z = false
vars.y_center = false
vars.ssd_r_min = false
vars.ssd_r_max = false
vars.ssd_az_min = false
vars.ssd_az_max = false
vars.mcp_r_min = false
vars.mcp_r_max = false
vars.mcp_az_min = false
vars.mcp_az_max = false
vars.esa_exit_r_min = false
vars.esa_exit_r_max = false
vars.esa_exit_az_min = false
vars.esa_exit_az_max = false
vars.cf_r_min = false
vars.cf_r_max = false
vars.cf_az_min = false
vars.cf_az_max = false
vars.lab_volt_opt = false
vars.heat_shield_ap_min_x = false
vars.heat_shield_ap_max_x = false
vars.heat_shield_ap_min_y = false
vars.heat_shield_ap_max_y = false
vars.heat_shield_ap_min_z = false
vars.heat_shield_ap_max_z = false
vars.simulation_min_y = false
vars.simulation_max_y = false
vars.simulation_min_z = false
vars.simulation_max_z = false
vars.tof_extent_x = false
vars.tof_extent_y = false
vars.model_res = false
vars.sim_D = false
vars.sim_T = false
vars.sim_k = false

batch.aberration = false
batch.az_fov_max = false
batch.az_fov_min = false
batch.az_fov_spacing = false
batch.cfoil_neutralize = false
batch.cfoil_thickness = false
batch.cfoil_transmission = false
batch.dwell = false
batch.flux = false
batch.ke_dist = false
batch.ke_max = false
batch.ke_mean = false
batch.ke_min = false
batch.ke_stdev = false
batch.m = false
batch.q = false
batch.simulation_batch_mode = false
batch.simulation_eais_only = false
batch.simulation_k = false
batch.simulation_mode = false
batch.simulation_sec_e_total = false
batch.simulation_sec_p_total = false
batch.simulation_tr_qual = false
batch.source_surface_style = false
batch.source_surface_shape = false
batch.source_surface_x = false
batch.simulation_view_retain = false
batch.species = false
batch.tr_az_cover_the_spread = false
batch.tr_az_dist = false
batch.tr_az_max = false
batch.tr_az_mean = false
batch.tr_az_min = false
batch.tr_az_stdev = false
batch.tr_az_steps = false
batch.tr_el_dist = false
batch.tr_el_max = false
batch.tr_el_mean = false
batch.tr_el_min = false
batch.tr_el_stdev = false
batch.tr_el_steps = false
batch.U_Ground = false
batch.U_MCP_Stop = false
batch.U_MCP_Start = false
batch.U_MCP_Case = false
batch.U_PAC = false
batch.U_TOF_UL_Defl = false
batch.U_Van = false
batch.U_Vld = false
batch.U_Vth = false
batch.U_Vud = false
batch.x_center = false
batch.xy_ap_in_radius = false
batch.xy_ap_min_radius = false
batch.xy_ap_out_radius = false
batch.z_ap_height = false
batch.z_ap_mid = false
batch.z_grid = false
batch.z_cf = false
batch.z_ssd = false
batch.z_stop_mcp = false
batch.z_start_mcp = false
batch.esa_exit_z = false
batch.y_center = false
batch.ssd_r_min = false
batch.ssd_r_max = false
batch.ssd_az_min = false
batch.ssd_az_max = false
batch.mcp_r_min = false
batch.mcp_r_max = false
batch.mcp_az_min = false
batch.mcp_az_max = false
batch.esa_exit_r_min = false
batch.esa_exit_r_max = false
batch.esa_exit_az_min = false
batch.esa_exit_az_max = false
batch.cf_r_min = false
batch.cf_r_max = false
batch.cf_az_min = false
batch.cf_az_max = false
batch.lab_volt_opt = false
batch.heat_shield_ap_min_x = false
batch.heat_shield_ap_max_x = false
batch.heat_shield_ap_min_y = false
batch.heat_shield_ap_max_y = false
batch.heat_shield_ap_min_z = false
batch.heat_shield_ap_max_z = false
batch.simulation_min_y = false
batch.simulation_max_y = false
batch.simulation_min_z = false
batch.simulation_max_z = false
batch.tof_extent_x = false
batch.tof_extent_y = false
batch.model_res = false
batch.sim_D = false
batch.sim_T = false
batch.sim_k = false
```

```
end

function splat_logic(sec_e,simulation_sec_e_total,sec_p,
   ↪ simulation_sec_p_total,
                                    ssd, start_mcp, stop_mcp,
                                    cf_px_mm,cf_py_mm,cf_pz_mm,
                                    cf_vx_mm,cf_vy_mm,cf_vz_mm,
                                    start_px_mm,start_py_mm,
                                       ↪ start_pz_mm,
                                    start_vx_mm,start_vy_mm,
                                       ↪ start_vz_mm,
                                    stop_px_mm,stop_py_mm,
                                       ↪ stop_pz_mm,
                                    stop_vx_mm,stop_vy_mm,
                                       ↪ stop_vz_mm,
                                    simulation_view_retain)

-- This function determines and creates the proper secondary
   ↪ particle after an ion or electron splat in the TOF. Should
   ↪ work for double and triple coincidence TOF instruments
-- It will create a series of secondary electrons up to quantity
   ↪ simulation_sec_e_total after initial ion impact inside the
   ↪ TOF and also from the point of carbon foil transit.
-- simulation_sec_e_total must be > 0 in order to simulate
   ↪ secondary electron effects.

-- Input Arguments:
--   sec_e,simulation_sec_e_total,sec_p,simulation_sec_p_total,
   ↪ Secondary electron/proton index and total. Indices should
   ↪ be initialized to 0 for each ion.
--              Total simulation_sec_e_total is the total number
   ↪ of secondary electrons that can be created in a series,
--              beginning with ion transit of the carbon foil, or
   ↪ initial ion impact inside of the the TOF.
--              simulation_sec_p_total can only be 0 or 1 (a
   ↪ cascade of secondary proton is not modeled
--  ssd, start_mcp, stop_mcp,     Flags to indicate impact has
   ↪ occured at the SSD, Start and Stop MCPS
--  cf_px_mm,cf_py_mm,cf_pz_mm,     Carbon Foil transit position
   ↪ and velocity
--  cf_px_mm,cf_py_mm,cf_pz_mm,
--  start_px_mm,start_py_mm,start_pz_mm,  Start MCP transit
   ↪ position and velocity
--  start_px_mm,start_py_mm,start_pz_mm,
--  stop_px_mm,stop_py_mm,stop_pz_mm,   Stop MCP transit position
   ↪  and velocity
--  stop_px_mm,stop_py_mm,stop_pz_mm)

-- Returned Outputs:
--  sec_e          An index to tell the function which secondary
   ↪ electrons have been created in series.
--              This is NOT a direct count of how many secondary
   ↪ electrons have been flown.
--  sec_p          An index to tell the function which secondary
   ↪ protons have been created in series.
--              This is NOT a direct count of how many secondary
   ↪ protons have been flown.



     --SECONDARY ELECTRON CREATION AFTER SPLAT
     if ion_splat == -1 and sec_e <= 2*simulation_sec_e_total +
        ↪ 1 then -- if there has been any kind of a splat after
        ↪  the carbon foil
           sec_e = sec_e + 1
     ----- INITIAL ION / NEUTRAL SPLAT INSIDE TOF ---------
           if ion_charge > -1 then -- an ion or neutral splats
              ↪ after the carbon foil
                 if simulation_sec_e_total ~= 0 then -- make a
                    ↪ secondary electron if total secondaries
                    ↪ desired is not 0
                       gen_secondary('electron','reverse',4,30)
```

```
                        ↪
                      -- create a "stop" electron on the SSD if
                        ↪  a triple coincidence instrument
                      if ssd == 1 then
                          set_color('red')
                      -- create a secondary electron if a wall
                        ↪ or stop mcp impact
                      else
                          set_color('orange')
                      end
                 end

-----ELECTRON SPLATS---------
            elseif ion_charge == -1 then -- an electron splats
              ↪ somewhere after the carbon foil
                 -- if this electron is not the last electron
                   ↪ permitted to be created in series after
                   ↪ ion impact, continue on with another
                   ↪ electron
                 if sec_e <= simulation_sec_e_total then
                      gen_secondary('electron','reverse',4,30)
                      set_color('orange')
                 -- this electron was the last electron
                   ↪ permitted to be created in the ion impact
                   ↪  series, set speed to 0, magically and
                   ↪ invisibly move to CF entry position
                 elseif sec_e > simulation_sec_e_total then
                      -- create a secondary electron on the CF
                        ↪ after all secondaries of the ion
                        ↪ impact are flown. Electron was
                        ↪ previously moved here.
                      if sec_e < simulation_sec_e_total+3 then
                        ↪ -- allow one timestep for marking
                        ↪ the stop MCP
                          ion_splat = 0
                          sim_trajectory_image_control =
                                ↪ simulation_view_retain
                      elseif sec_e == simulation_sec_e_total+3
                        ↪ then -- now relocate to the CF
                          sim_trajectory_image_control = 3
                          set_color('green')
                          ion_px_mm = cf_px_mm
                          ion_py_mm = cf_py_mm
                          ion_pz_mm = cf_pz_mm
                          ion_vx_mm = cf_vx_mm -- resume
                             ↪ trajectory at carbon foil (
                             ↪ basis for scatter direction)
                          ion_vy_mm = cf_vy_mm
                          ion_vz_mm = cf_vz_mm

                          gen_secondary('electron','
                             ↪ do_not_reverse_direction'
                             ↪ ,4,30)

                     -- "start" electron series numbers 2 ->
                        ↪ sec_e
                      else
                          gen_secondary('electron','reverse'
                             ↪ ,4,30)
                          set_color('orange')
                      end
                 end
            end
      end

----------- PROTON EJECTIONS FROM MCPS AFTER ALL ELECTRONS
   ↪ FLOWN -----------------

if (sec_e == 2*simulation_sec_e_total +2 or
   ↪ simulation_sec_e_total == 0) and ion_splat == -1 and
   ↪ simulation_sec_p_total > 0 then
            if start_mcp == 1 then
```

```
            if sec_p == 0 then
                  sec_p = 1
                  ion_splat = 0
                  kinetic_energy = 0
                  speed = 0
                  ion_px_mm = start_px_mm
                  ion_py_mm = start_py_mm
                  ion_pz_mm = start_pz_mm
                  sim_trajectory_image_control = 3
            elseif sec_p == 1 then
                  sim_trajectory_image_control = 0
                  set_color('blue')
                  ion_vx_mm = start_vx_mm -- resume
                     ↪ trajectory at start (basis for
                     ↪ scatter direction)
                  ion_vy_mm = start_vy_mm
                  ion_vz_mm = start_vz_mm
                  gen_secondary('p','reverse',4,30)
                  sec_p = 2
            end
      else
            sec_p = 2
      end

if stop_mcp == 1 and (sec_e == 2*simulation_sec_e_total +2
   ↪ or simulation_sec_e_total == 0) and ion_splat == -1
   ↪ and sec_p >= 2 then

            if sec_p == 2 then
                  sec_p = 3
                  ion_splat = 0
                  kinetic_energy = 0
                  speed = 0
                  ion_px_mm = stop_px_mm
                  ion_py_mm = stop_py_mm
                  ion_pz_mm = stop_pz_mm
```

```
                        sim_trajectory_image_control = 3
                elseif sec_p == 3 then
                        sim_trajectory_image_control = 0
                        set_color('blue')
                        ion_vx_mm = stop_vx_mm -- resume
                           ↪ trajectory at start (basis for
                           ↪ scatter direction)
                        ion_vy_mm = stop_vy_mm
                        ion_vz_mm = stop_vz_mm
                        gen_secondary('p','reverse',4,30)
                        sec_p = 4
                end

        end
    end

    return sec_e, sec_p
end

function gen_secondary(p_or_e,reversal,sec_ke,scatter_half_cone)

    ion_splat = 0 -- if you've called this function, you
       ↪ probably want your particle to live on

    if (p_or_e == 'proton' or p_or_e == 'p') then
        ion_mass = 1.007276466812 -- (amu)
        ion_charge = 1
    else
        ion_mass = 5.485799033e-4 -- (amu)
        ion_charge = -1
    end

    speed, az, el = rect3d_to_polar3d(ion_vx_mm,ion_vy_mm,
       ↪ ion_vz_mm)-- record ion direction

    kinetic_energy = sec_ke*rand() -- (eV) assign some energy
```

```
        ↪ to secondary particle
      speed = ke_to_speed(kinetic_energy,ion_mass) -- calculate
        ↪ speed based on new mass and kinetic energy(from eV to
        ↪  mm/usec)

      if (reversal == 'reverse') then -- reverse direction if
        ↪ requested
            ion_vx_mm, ion_vy_mm, ion_vz_mm = polar3d_to_rect3d(
              ↪ speed,az,el)
            speed, az, el = rect3d_to_polar3d(-ion_vx_mm,-
              ↪ ion_vy_mm,-ion_vz_mm)
      end

      ion_vx_mm, ion_vy_mm, ion_vz_mm = polar3d_to_rect3d(speed,
        ↪ az,el+scatter_half_cone*(2*rand()-1)) -- apply new
        ↪ speed with a scattering angle

end

function set_color(color) -- use to change color by text. easier
  ↪ to read.
      if (color == 'black') then ion_color = 0
      elseif (color == 'red') then ion_color = 1
      elseif (color == 'green') then ion_color = 2
      elseif (color == 'blue') then ion_color = 3
      elseif (color == 'light␣yellow') then ion_color = 4
      elseif (color == 'rose') then ion_color = 5
      elseif (color == 'light␣green') then ion_color = 6
      elseif (color == 'light␣purple') then ion_color = 7
      elseif (color == 'light␣blue') then ion_color = 8
      elseif (color == 'light␣pink') then ion_color = 9
      elseif (color == 'orange') then ion_color = 10
      elseif (color == 'brown') then ion_color = 11
      elseif (color == 'beige') then ion_color = 12
      elseif (color == 'light␣brown') then ion_color = 13
      elseif (color == 'lighter␣brown') then ion_color = 14
      elseif (color == 'white') then ion_color = 15
      end
end

function random(mini,maxi)  -- create a random number between two
   ↪  values
  return mini + (maxi-mini)*rand()
end

function rand_gauss(mean,sigma) -- create random number, normal
   ↪ distribution
  if iset==0 then -- (doesn't allow numbers < 0)
     rsq = 2 -- initialize this
     while ((rsq>=1) or (rsq==0)) do
        v1 = 2*rand()-1
        v2 = 2*rand()-1
        rsq=v1*v1+v2*v2
     end
     fac=sqrt(-2*ln(rsq)/rsq)
     gset=v1*fac
     iset=1
     rand_temp=(v2*fac)*sigma+mean
  else
     iset=0
     rand_temp=gset*sigma+mean3
  end
  if rand_temp < 0 then
     rand_temp = rand_gauss(mean,sigma)
  end
  return rand_temp
end

function cfoil_loss(mass, ke, cfm)

co = {}
```

```
if mass>26.982-0.01 and mass<26.982+0.01  then co = { species =
   ↪ Al, m = 26.982, mn = 0.0037062, mx = 370.6174, fit = "
   ↪ Fourier", w = 0.88806, a0 = 0.5299, a1 = -0.065645, a2 = 0
   ↪ .011977, a3 = 0.00031421, a4 = 0.018231, a5 = -0.012369, a6
   ↪  = 0.0058772, b1 = 0.44073, b2 = -0.32809, b3 = 0.10666, b4
   ↪  = -0.041778, b5 = 0.014151, b6 = -0.0049159, c1 = 0, c2 =
   ↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>39.948-0.01 and mass<39.948+0.01  then co = { species
   ↪  = Ar, m = 39.948, mn = 0.0025024, mx = 250.2377, fit = "
   ↪ Fourier", w = 0.20842, a0 = -3143185.6479, a1 = 5421010.124
   ↪ , a2 = -3450265.4754, a3 = 1580746.581, a4 = -494916.6729,
   ↪ a5 = 95093.0152, a6 = -8481.2105, b1 = -140076.6812, b2 =
   ↪ 178675.7633, b3 = -123227.9576, b4 = 51713.1886, b5 =
   ↪ -12511.4266, b6 = 1352.2893, c1 = 0, c2 = 0, c3 = 0, c4 =
   ↪ 0, c5 = 0, c6 = 0}
elseif mass>10.812-0.01 and mass<10.812+0.01  then co = { species
   ↪  = B, m = 10.812, mn = 0.0090835, mx = 908.3477, fit = "
   ↪ Fourier", w = 0.96363, a0 = 0.20557, a1 = -0.087712, a2 = 0
   ↪ .028986, a3 = -0.039177, a4 = 0.022831, a5 = -0.024065, a6
   ↪ = 0.0093501, b1 = 0.433, b2 = -0.21439, b3 = 0.078501, b4 =
   ↪  -0.017736, b5 = 0.00056413, b6 = -0.0055936, c1 = 0, c2 =
   ↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>9.012-0.01 and mass<9.012+0.01  then co = { species
   ↪ = Be, m = 9.012, mn = 0.011096, mx = 1109.6316, fit = "
   ↪ Fourier", w = 0.99127, a0 = 0.078382, a1 = -0.16581, a2 = 0
   ↪ .045649, a3 = -0.050989, a4 = 0.026374, a5 = -0.012669, a6
   ↪ = 0.010358, b1 = 0.39939, b2 = -0.20249, b3 = 0.051752, b4
   ↪ = -0.0083149, b5 = -0.012209, b6 = 0.0059269, c1 = 0, c2 =
   ↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>12.011-0.01 and mass<12.011+0.01  then co = { species
   ↪  = C, m = 12.011, mn = 0.0083333, mx = 833.3333, fit = "
   ↪ Fourier", w = 0.92675, a0 = 0.26554, a1 = -0.077235, a2 = 0
   ↪ .020435, a3 = -0.029411, a4 = 0.022711, a5 = -0.01847, a6 =
   ↪  0.0057352, b1 = 0.45215, b2 = -0.23527, b3 = 0.07531, b4 =
   ↪  -0.017189, b5 = 0.0029611, b6 = -0.0026864, c1 = 0, c2 =
   ↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
```

```
elseif mass>40.079-0.01 and mass<40.079+0.01  then co = { species
    ↪  = Ca, m = 40.079, mn = 0.0025023, mx = 250.2315, fit = "
    ↪ Gauss", w = 0, a0 = 0, a1 = -0.39232, a2 = 1.2588, a3 = 0
    ↪ .027829, a4 = 0.69806, a5 = 0, a6 = 0, b1 = -3.4979, b2 = 2
    ↪ .7473, b3 = 0.94553, b4 = -0.081756, b5 = 0, b6 = 0, c1 = 1
    ↪ .1551, c2 = 1.3175, c3 = 0.26277, c4 = 2.154, c5 = 0, c6 =
    ↪ 0}
elseif mass>35.453-0.01 and mass<35.453+0.01  then co = { species
    ↪  = Cl, m = 35.453, mn = 0.0028597, mx = 285.9676, fit = "
    ↪ Fourier", w = 0.94973, a0 = 0.62457, a1 = 0.07525, a2 = -0
    ↪ .04293, a3 = 0.067143, a4 = -0.028971, a5 = 0.012096, a6 =
    ↪ -0.012148, b1 = 0.41234, b2 = -0.30575, b3 = 0.13792, b4 =
    ↪ -0.051806, b5 = 0.031859, b6 = -0.0082689, c1 = 0, c2 = 0,
    ↪ c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>51.996-0.01 and mass<51.996+0.01  then co = { species
    ↪  = Cr, m = 51.996, mn = 0.0019253, mx = 192.5298, fit = "
    ↪ Gauss", w = 0, a0 = 0, a1 = -0.00039861, a2 = -0.058654, a3
    ↪  = 1.1004, a4 = -0.015176, a5 = 0.79939, a6 = -1.506, b1 =
    ↪ 2.3843, b2 = 1.3722, b3 = 2.6732, b4 = 0.49488, b5 = 0
    ↪ .022579, b6 = -5.9639, c1 = 0.57607, c2 = 0.36369, c3 = 1
    ↪ .1195, c4 = 0.19345, c5 = 2.7788, c6 = 2.6774}
elseif mass>18.998-0.01 and mass<18.998+0.01  then co = { species
    ↪  = F, m = 18.998, mn = 0.0052637, mx = 526.3712, fit = "
    ↪ Fourier", w = 0.86058, a0 = 0.49071, a1 = -0.19998, a2 = 0
    ↪ .098608, a3 = -0.076262, a4 = 0.051399, a5 = -0.02261, a6 =
    ↪  0.0068985, b1 = 0.43632, b2 = -0.21291, b3 = -0.0031722,
    ↪ b4 = 0.039654, b5 = -0.031795, b6 = 0.014854, c1 = 0, c2 =
    ↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>55.847-0.01 and mass<55.847+0.01  then co = { species
    ↪  = Fe, m = 55.847, mn = 0.0017878, mx = 178.7789, fit = "
    ↪ Fourier", w = 0.88929, a0 = 0.65521, a1 = 0.19589, a2 = -0
    ↪ .13436, a3 = 0.1442, a4 = -0.063328, a5 = 0.020802, a6 = -0
    ↪ .0065427, b1 = 0.41708, b2 = -0.32383, b3 = 0.10337, b4 =
    ↪ -0.013468, b5 = 0.0019637, b6 = 0.0011789, c1 = 0, c2 = 0,
    ↪ c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>1.008-0.01 and mass<1.008+0.01   then co = { species
```

```
↪ = H, m = 1.008, mn = 0.099206, mx = 9920.6349, fit = "Gauss
↪ ", w = 0, a0 = 0, a1 = -1.2907, a2 = 0.61403, a3 = -0
↪ .051859, a4 = 1.9022, a5 = -0.067703, a6 = -3.4625, b1 = -1
↪ .9226, b2 = 2.2932, b3 = 0.12056, b4 = 3.9919, b5 = 2.591,
↪ b6 = 4.3747, c1 = 2.5305, c2 = 1.112, c3 = 0.85214, c4 = 1
↪ .4289, c5 = 0.50814, c6 = 2.0014}
elseif mass>4.003-0.01 and mass<4.003+0.01  then co = { species
↪ = He, m = 4.003, mn = 0.024981, mx = 2498.1264, fit = "
↪ Fourier", w = 0.96078, a0 = -0.29056, a1 = -0.088292, a2 =
↪ -0.020826, a3 = -0.0083957, a4 = -0.0052959, a5 = -0
↪ .00073533, a6 = 0.001221, b1 = 0.47251, b2 = -0.13002, b3 =
↪  0.015555, b4 = 0.0040059, b5 = -0.0016747, b6 = -0.0027024
↪ , c1 = 0, c2 = 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>3.016-0.01 and mass<3.016+0.01  then co = { species
↪ = He3, m = 3.016, mn = 0.033156, mx = 3315.6499, fit = "
↪ Fourier", w = 1.0101, a0 = -0.28641, a1 = -0.14587, a2 = 0
↪ .026627, a3 = -0.018889, a4 = -0.0054679, a5 = 0.0018445,
↪ a6 = 0.0010344, b1 = 0.48028, b2 = -0.1167, b3 = 0.0070012,
↪  b4 = 0.0077125, b5 = -0.0030363, b6 = -0.0020792, c1 = 0,
↪ c2 = 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>39.098-0.01 and mass<39.098+0.01  then co = { species
↪  = K, m = 39.098, mn = 0.0025665, mx = 256.6472, fit = "
↪ Fourier", w = 0.7611, a0 = 0.89923, a1 = -0.39613, a2 = 0
↪ .32707, a3 = -0.20587, a4 = 0.13275, a5 = -0.055174, a6 = 0
↪ .010244, b1 = 0.62025, b2 = -0.44001, b3 = 0.17996, b4 = -0
↪ .088826, b5 = 0.045785, b6 = -0.014943, c1 = 0, c2 = 0, c3
↪ = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>6.941-0.01 and mass<6.941+0.01  then co = { species
↪ = Li, m = 6.941, mn = 0.014253, mx = 1425.3136, fit = "
↪ Fourier", w = 0.97584, a0 = -0.062422, a1 = -0.15882, a2 =
↪ 0.017742, a3 = -0.02683, a4 = 0.009513, a5 = -0.0068372, a6
↪  = 0.0035657, b1 = 0.4261, b2 = -0.19072, b3 = 0.038738, b4
↪  = 0.0072922, b5 = -0.013947, b6 = 0.005456, c1 = 0, c2 =
↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>24.305-0.01 and mass<24.305+0.01  then co = { species
↪  = Mg, m = 24.305, mn = 0.0041693, mx = 416.9272, fit = "
↪ Fourier", w = 0.82257, a0 = 0.39013, a1 = 0.10188, a2 = -0
↪ .14697, a3 = 0.098263, a4 = -0.026359, a5 = 0.012345, a6 =
↪ -0.005432, b1 = 0.49799, b2 = -0.35124, b3 = 0.084325, b4 =
↪  -0.031654, b5 = 0.014517, b6 = -0.0021904, c1 = 0, c2 = 0,
↪  c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>54.938-0.01 and mass<54.938+0.01  then co = { species
↪  = Mn, m = 54.938, mn = 0.0018202, mx = 182.0167, fit = "
↪ Fourier", w = 0.20946, a0 = 3196440.6041, a1 = -5489949
↪ .5967, a2 = 3450316.9511, a3 = -1547177.623, a4 = 469608
↪ .9354, a5 = -86557.0081, a6 = 7318.5279, b1 = 488010.729,
↪ b2 = -619050.0834, b3 = 422904.0676, b4 = -175010.2843, b5
↪ = 41535.5596, b6 = -4376.5572, c1 = 0, c2 = 0, c3 = 0, c4 =
↪  0, c5 = 0, c6 = 0}
elseif mass>14.007-0.01 and mass<14.007+0.01  then co = { species
↪  = N, m = 14.007, mn = 0.0071413, mx = 714.1327, fit = "
↪ Fourier", w = 0.93959, a0 = 0.33316, a1 = -0.06642, a2 = 0
↪ .020839, a3 = -0.022151, a4 = 0.023014, a5 = -0.021666, a6
↪ = 0.0049863, b1 = 0.4439, b2 = -0.25241, b3 = 0.092322, b4
↪ = -0.026662, b5 = 0.012061, b6 = -0.0074352, c1 = 0, c2 =
↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>22.99-0.01 and mass<22.99+0.01   then co = { species
↪ = Na, m = 22.99, mn = 0.0043497, mx = 434.9717, fit = "
↪ Fourier", w = 0.8169, a0 = 0.49519, a1 = -0.12567, a2 = 0
↪ .044834, a3 = -0.037947, a4 = 0.048986, a5 = -0.024552, a6
↪ = 0.0074218, b1 = 0.47993, b2 = -0.30706, b3 = 0.050333, b4
↪  = -0.0047883, b5 = -0.0013168, b6 = 0.0007335, c1 = 0, c2
↪ = 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>20.18-0.01 and mass<20.18+0.01   then co = { species
↪ = Ne, m = 20.18, mn = 0.005002, mx = 500.2001, fit = "
↪ Fourier", w = 0.92727, a0 = 0.41196, a1 = -0.027019, a2 =
↪ -0.028997, a3 = 0.017998, a4 = -0.00032641, a5 = -0.0052843
↪ , a6 = -0.0037974, b1 = 0.43535, b2 = -0.30235, b3 = 0
↪ .11514, b4 = -0.050785, b5 = 0.026139, b6 = -0.010665, c1 =
↪  0, c2 = 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>15.999-0.01 and mass<15.999+0.01  then co = { species
↪  = O, m = 15.999, mn = 0.0062519, mx = 625.1954, fit = "
```

```
  ↪ Fourier", w = 0.96021, a0 = 0.39708, a1 = -0.11413, a2 = 0
  ↪ .055043, a3 = -0.035116, a4 = 0.028891, a5 = -0.022677, a6
  ↪ = 0.010561, b1 = 0.41147, b2 = -0.25387, b3 = 0.086336, b4
  ↪ = -0.019264, b5 = 0.000098738, b6 = -0.00047996, c1 = 0, c2
  ↪  = 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>30.974-0.01 and mass<30.974+0.01  then co = { species
  ↪  = P, m = 30.974, mn = 0.0032285, mx = 322.8514, fit = "
  ↪ Fourier", w = 0.89648, a0 = 0.61491, a1 = -0.049191, a2 = 0
  ↪ .030612, a3 = -0.016678, a4 = 0.02435, a5 = -0.017778, a6 =
  ↪  0.0060618, b1 = 0.45352, b2 = -0.32004, b3 = 0.12421, b4 =
  ↪  -0.054594, b5 = 0.023639, b6 = -0.010665, c1 = 0, c2 = 0,
  ↪ c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>32.067-0.01 and mass<32.067+0.01  then co = { species
  ↪  = S, m = 32.067, mn = 0.0031277, mx = 312.7737, fit = "
  ↪ Fourier", w = 0.78251, a0 = 0.63608, a1 = -0.030625, a2 = 0
  ↪ .02563, a3 = -0.01169, a4 = 0.034352, a5 = -0.015327, a6 =
  ↪ 0.0049294, b1 = 0.53961, b2 = -0.33857, b3 = 0.081034, b4 =
  ↪  -0.018409, b5 = 0.0054123, b6 = 0.00099286, c1 = 0, c2 =
  ↪ 0, c3 = 0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>44.956-0.01 and mass<44.956+0.01  then co = { species
  ↪  = Sc, m = 44.956, mn = 0.0022244, mx = 222.4397, fit = "
  ↪ Fourier", w = 0.77792, a0 = 0.89217, a1 = -0.33772, a2 = 0
  ↪ .29392, a3 = -0.20054, a4 = 0.14025, a5 = -0.067547, a6 = 0
  ↪ .019099, b1 = 0.57217, b2 = -0.39555, b3 = 0.14341, b4 = -0
  ↪ .067564, b5 = 0.033117, b6 = -0.01452, c1 = 0, c2 = 0, c3 =
  ↪  0, c4 = 0, c5 = 0, c6 = 0}
elseif mass>28.086-0.01 and mass<28.086+0.01  then co = { species
  ↪  = Si, m = 28.086, mn = 0.0035744, mx = 357.4365, fit = "
  ↪ Fourier", w = 0.26168, a0 = -1005743.0274, a1 = 1741500
  ↪ .8914, a2 = -1121730.6821, a3 = 524322.1043, a4 = -168866
  ↪ .4698, a5 = 33659.3231, a6 = -3141.5568, b1 = -5265.9282,
  ↪ b2 = 7109.7542, b3 = -5376.4253, b4 = 2555.8412, b5 = -721
  ↪ .5715, b6 = 93.4103, c1 = 0, c2 = 0, c3 = 0, c4 = 0, c5 =
  ↪ 0, c6 = 0}
elseif mass>47.883-0.01 and mass<47.883+0.01  then co = { species
  ↪  = Ti, m = 47.883, mn = 0.0020855, mx = 208.5506, fit = "
```

```
    ↪ Gauss", w = 0, a0 = 0, a1 = 0, a2 = 1.7614, a3 = 0.06668,
    ↪ a4 = -21260330823.6476, a5 = 5.1893, a6 = 0, b1 = 15.1003,
    ↪ b2 = 2.8954, b3 = 0.83248, b4 = 784.4674, b5 = 0.8198, b6 =
    ↪  0, c1 = 2.0237, c2 = 1.271, c3 = 0.39524, c4 = 166.0902,
    ↪ c5 = 6.0114, c6 = 0}
elseif mass>50.942-0.01 and mass<50.942+0.01  then co = { species
    ↪  = V, m = 50.942, mn = 0.0019631, mx = 196.3094, fit = "
    ↪ Fourier", w = 0.55279, a0 = 70.3663, a1 = -123.1289, a2 =
    ↪ 84.5328, a3 = -43.8535, a4 = 16.2736, a5 = -3.7803, a6 = 0
    ↪ .39433, b1 = 19.6165, b2 = -26.9609, b3 = 21.8665, b4 = -11
    ↪ .8477, b5 = 4.0297, b6 = -0.6599, c1 = 0, c2 = 0, c3 = 0,
    ↪ c4 = 0, c5 = 0, c6 = 0}
else
co = { species = None, m = 0, mn = 0, mx = 0, fit = "None", w =
    ↪ 0, a0 = 0, a1 = 0, a2 = 0, a3 = 0, a4 = 0, a5 = 0, a6 = 0,
    ↪ b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, b6 = 0, c1 = 1, c2
    ↪ = 1, c3 = 1, c4 = 1, c5 = 1, c6 = 1}
end

kev_per_nuc = ke/1000/mass

if co.mn<kev_per_nuc and co.mx>kev_per_nuc then
   sigma = 0

   if co.fit == "Fourier" then
      for i=1,6 do
            ai,bi=co["a"..i],co["b"..i]
            dsigma = ai*math.cos(i*math.log10(kev_per_nuc)*co.w)+
               ↪ bi*math.sin(i*math.log10(kev_per_nuc)*co.w)
            sigma = sigma + dsigma
      end
      sigma = sigma + co.a0
   else
      for i=1,6 do
            ai,bi,ci=co["a"..i],co["b"..i],co["c"..i]
            dsigma = ai*2.7182818284^(-((math.log10(kev_per_nuc)-
                ↪ bi)/ci)^2)
            sigma = sigma + dsigma
        end
    end

    dEdX= 10^sigma

    dE = cfm*dEdX
else
    dE = ke/1000 -- If it's outside of the range used in the TRIM
        ↪ calculations, just kill it
end

return dE


end

function cfoil_scatter(mean_deflection,mass,ke,cfm) --

co = {}
if mass>26.982-0.01 and mass<26.982+0.01 then co = { species = Al
    ↪ , Z = 13, m = 26.982, c0 = 15.746, c1 = 0.7471, c2 = 0.6843
    ↪ , c3 = -0.8818}
elseif mass>39.948-0.01 and mass<39.948+0.01 then co = { species
    ↪ = Ar, Z = 18, m = 39.948, c0 = 15.746, c1 = 0.7471, c2 = 0
    ↪ .6843, c3 = -0.8818}
elseif mass>10.812-0.01 and mass<10.812+0.01 then co = { species
    ↪ = B, Z = 5, m = 10.812, c0 = 15.746, c1 = 0.7471, c2 = 0
    ↪ .6843, c3 = -0.8818}
elseif mass>9.012-0.01 and mass<9.012+0.01 then co = { species =
    ↪ Be, Z = 4, m = 9.012, c0 = 15.746, c1 = 0.7471, c2 = 0.6843
    ↪ , c3 = -0.8818}
elseif mass>12.011-0.01 and mass<12.011+0.01 then co = { species
    ↪ = C, Z = 6, m = 12.011, c0 = 15.746, c1 = 0.7471, c2 = 0
    ↪ .6843, c3 = -0.8818}
```

```
elseif mass>40.079-0.01 and mass<40.079+0.01 then co = { species
   ↪ = Ca, Z = 20, m = 40.079, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>35.453-0.01 and mass<35.453+0.01 then co = { species
   ↪ = Cl, Z = 17, m = 35.453, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>51.996-0.01 and mass<51.996+0.01 then co = { species
   ↪ = Cr, Z = 24, m = 51.996, c0 = 13.131, c1 = 0.7471, c2 = 0
   ↪ .7038, c3 = -0.8955}
elseif mass>18.998-0.01 and mass<18.998+0.01 then co = { species
   ↪ = F, Z = 9, m = 18.998, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>55.847-0.01 and mass<55.847+0.01 then co = { species
   ↪ = Fe, Z = 26, m = 55.847, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>1.008-0.01 and mass<1.008+0.01 then co = { species =
   ↪ H, Z = 1, m = 1.008, c0 = 15.746, c1 = 0.7471, c2 = 0.6843,
   ↪  c3 = -0.8818}
elseif mass>4.003-0.01 and mass<4.003+0.01 then co = { species =
   ↪ He, Z = 2, m = 4.003, c0 = 15.746, c1 = 0.7471, c2 = 0.6843
   ↪ , c3 = -0.8818}
elseif mass>3.016-0.01 and mass<3.016+0.01 then co = { species =
   ↪ He3, Z = 2, m = 3.016, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>39.098-0.01 and mass<39.098+0.01 then co = { species
   ↪ = K, Z = 19, m = 39.098, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>6.941-0.01 and mass<6.941+0.01 then co = { species =
   ↪ Li, Z = 3, m = 6.941, c0 = 15.746, c1 = 0.7471, c2 = 0.6843
   ↪ , c3 = -0.8818}
elseif mass>24.305-0.01 and mass<24.305+0.01 then co = { species
   ↪ = Mg, Z = 12, m = 24.305, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>54.938-0.01 and mass<54.938+0.01 then co = { species
   ↪ = Mn, Z = 25, m = 54.938, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
```

```
elseif mass>14.007-0.01 and mass<14.007+0.01 then co = { species
   ↪ = N, Z = 7, m = 14.007, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>22.99-0.01 and mass<22.99+0.01 then co = { species =
   ↪ Na, Z = 11, m = 22.99, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>20.18-0.01 and mass<20.18+0.01 then co = { species =
   ↪ Ne, Z = 10, m = 20.18, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>15.999-0.01 and mass<15.999+0.01 then co = { species
   ↪ = O, Z = 8, m = 15.999, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>30.974-0.01 and mass<30.974+0.01 then co = { species
   ↪ = P, Z = 15, m = 30.974, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>32.067-0.01 and mass<32.067+0.01 then co = { species
   ↪ = S, Z = 16, m = 32.067, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>44.956-0.01 and mass<44.956+0.01 then co = { species
   ↪ = Sc, Z = 21, m = 44.956, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>28.086-0.01 and mass<28.086+0.01 then co = { species
   ↪ = Si, Z = 14, m = 28.086, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>47.883-0.01 and mass<47.883+0.01 then co = { species
   ↪ = Ti, Z = 22, m = 47.883, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
elseif mass>50.942-0.01 and mass<50.942+0.01 then co = { species
   ↪ = V, Z = 23, m = 50.942, c0 = 15.746, c1 = 0.7471, c2 = 0
   ↪ .6843, c3 = -0.8818}
else
co = { species = NA, Z = 0, m = 0, c0 = 0, c1 = 0, c2 = 0, c3 =
   ↪ 0}
end
```

```
        s = 0.1356
        phi=co.c0*co.Z^co.c1*cfm^co.c2*(ke-cfm*s*mass/2)^co.c3
        return mean_deflection + phi*math.tan(math.pi*(rand() - 0.5
           ↪ ))
end

function reload_fly2(filename, var)
  local key
  for k,v in pairs(debug.getregistry()) do
    if type(v)=='table' and v.iterator then key = k; break end
  end
  assert(key)
  local ok, err = xpcall(function()
    _G.var = var -- set vars
    debug.getregistry()[key] = simion.iob2.load_fly2_file(filename
       ↪ )
  end, debug.traceback)
  _G.var = nil -- clear (the pcall ensures this is always
     ↪ executed.
  if not ok then error(err) end
end
```

APPENDIX F

HIS Python Library

The SO-HIS post-processing analysis Python library is included below.

Listing F.1: Python library used for post-processing of SO-HIS SIMION runs.

```python
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 17 19:31:13 2020

@author: saraylet
"""

__author__ = 'Sarah␣Spitzer;␣saraylet@umich.edu'


import math
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.gridspec as gridspec
from scipy.spatial import distance
import scipy.stats as stats
from scipy.interpolate import UnivariateSpline
from scipy.optimize import curve_fit
import csv
import os
# DEBUG
import pdb
import warnings


# global constants
DEGREE_SIGN = u"\N{DEGREE␣SIGN}"
SIMION_TO_SWRI_FACTOR = 4


# function to help print a number in human readable format
def hrn(num):

   k_str = ''
   print_num = num

   if num >= 1000000:
      print_num = num / 1000000
      k_str = 'M'
   elif num >= 1000:
      print_num = num / 1000
      k_str = 'k'

   return k_str, str(print_num)


# global HIS coordiate constants
def coordinates(x_center, y_center, z_center, cf_z, ssd_z,
   ↪ mcp_start_z, mcp_stop_z, \
            esa_exit_z, tof_extent_x, tof_extent_y,
               ↪ heat_shield_ap_min_x, \
            heat_shield_ap_max_x, heat_shield_ap_min_y,
               ↪ heat_shield_ap_max_y, \
            heat_shield_ap_min_z, heat_shield_ap_max_z,
               ↪ z_ap_height, \
            aperture_out_r, aperture_mid_r, aperture_in_r,
               ↪ aperture_theta_min, \
            aperture_theta_max, ssd_r_min, ssd_r_max, ssd_az_min,
                ↪  ssd_az_max,\
            mcp_r_min, mcp_r_max, mcp_az_min, mcp_az_max,
                ↪ esa_exit_r_min, \
            esa_exit_r_max, esa_exit_az_min, esa_exit_az_max,
                ↪ cf_r_min, \
            cf_r_max, cf_az_min, cf_az_max, model_res,
                ↪ source_surface_x, \
            tr_az_dist, tr_az_max, tr_az_mean, tr_az_min,
                ↪ tr_az_stdev, \
            tr_az_steps, tr_el_dist, tr_el_max, tr_el_mean,
                ↪ tr_el_min, \
            tr_el_stdev, tr_el_steps, ke_dist, ke_max, ke_mean,
                ↪ ke_min, ke_stdev):


    tol_x = model_res
    tol_z = model_res
    tol_r = math.sqrt(tol_x**2 + tol_z**2)
    tol_theta_mcp = 2 * math.asin((math.sqrt(2)/8)/mcp_r_min) *
        ↪ 180/math.pi
    tol_theta_ssd = 2 * math.asin((math.sqrt(2)/8)/ssd_r_min) *
        ↪ 180/math.pi
    tol_theta_ap = 2 * math.asin((math.sqrt(2)/8)/aperture_in_r) *
        ↪  180/math.pi
    tol_theta_esa_exit = 2 * math.asin((math.sqrt(2)/8)/
        ↪ esa_exit_r_min) * 180/math.pi
    tol_theta_cf = 2 * math.asin((math.sqrt(2)/8)/cf_r_min) * 180/
        ↪ math.pi

    # coordinate configuration dictionary
    his_config = {

        "x_center" : x_center,
        "y_center" : y_center,
        "z_center" : z_center,
        "cf_z" : cf_z,
        "ssd_z" : ssd_z,
        "mcp_start_z" : mcp_start_z,
        "mcap_stop_z" : mcp_stop_z,

        "x_tol" : tol_x,
        "z_tol" : tol_z,
        "r_tol" : tol_r,
        "theta_tol" : tol_theta_mcp,
        "theta_tol_ssd" : tol_theta_ssd,
        "theta_tol_ap" : tol_theta_ap,

        "cf_z_lims" : [cf_z-tol_z, cf_z+tol_z],
        "ssd_z_lims" : [ssd_z-tol_z, ssd_z+(2*tol_z)],
        "mcp_start_z_lims" : [mcp_start_z-tol_z, mcp_start_z+(2*
           ↪ tol_z)],
        "mcp_stop_z_lims" : [mcp_stop_z-(2*tol_z), mcp_stop_z+tol_z
           ↪ ],

        "tof_x_lims" : [x_center-tof_extent_x,x_center],
        "tof_y_lims" : [y_center-tof_extent_y/2,y_center+
           ↪ tof_extent_y/2],

        "aperture_x_lims": [x_center,heat_shield_ap_max_x],
        "aperture_y_lims": [heat_shield_ap_min_y,
           ↪ heat_shield_ap_max_y],
        "aperture_z_lims": [z_center-z_ap_height-tol_z,z_center+
           ↪ z_ap_height+tol_z],

        "esa_exit_z" : esa_exit_z,
        "esa_exit_z_lims" : [esa_exit_z-tol_z, esa_exit_z+tol_z],

        "mcp_r_lims" : [mcp_r_min-tol_r, mcp_r_max+tol_r],
        "mcp_theta_lims" : [mcp_az_min-tol_theta_mcp, mcp_az_max+
           ↪ tol_theta_mcp],
        "ssd_r_lims" : [ssd_r_min-tol_r, ssd_r_max+tol_r],
        "ssd_theta_lims" : [ssd_az_min-tol_theta_ssd, ssd_az_max+
            ↪ tol_theta_ssd],
        "ssd_total_area_no_tol": (np.pi/360)*(ssd_r_max**2 -
            ↪ ssd_r_min**2)*(ssd_az_max - ssd_az_min),
        "ssd_total_area_tol": (np.pi/360)*((ssd_r_max+tol_r)**2 - (
            ↪ ssd_r_min-tol_r)**2)*((ssd_az_max+tol_theta_ssd) - (
            ↪ ssd_az_min-tol_theta_ssd)),

        "aperture_out_r" : aperture_out_r,
        "aperture_out_r_lims" : [aperture_out_r-tol_r,
            ↪ aperture_out_r+tol_r],
        "aperture_mid_r" : aperture_mid_r,
        "aperture_mid_r_lims" : [aperture_mid_r-tol_r,
            ↪ aperture_mid_r+tol_r],
        "aperture_in_r" : aperture_in_r,
        "aperture_in_r_lims" : [aperture_in_r-tol_r,aperture_in_r+
            ↪ tol_r],
        "aperture_theta_lims" : [aperture_theta_min-tol_theta_ap,
            ↪ aperture_theta_max+tol_theta_ap],

        "esa_exit_r_lims" : [esa_exit_r_min-tol_r, esa_exit_r_max+
            ↪ tol_r],
        "esa_exit_theta_lims" : [esa_exit_az_min-tol_theta_esa_exit
            ↪ , esa_exit_az_max+tol_theta_esa_exit],
        "cf_r_lims" : [cf_r_min-tol_r, cf_r_max+tol_r],
        "cf_theta_lims" : [cf_az_min-tol_theta_cf, cf_az_max+
            ↪ tol_theta_cf],

        "source_surface_x" : source_surface_x,
        "tr_az" : {"dist":tr_az_dist,"min":tr_az_min,"max":
            ↪ tr_az_max,"mean":tr_az_mean,"stdev":tr_az_stdev,"
            ↪ steps":tr_az_steps},
        "tr_el" : {"dist":tr_el_dist,"min":tr_el_min,"max":
            ↪ tr_el_max,"mean":tr_el_mean,"stdev":tr_el_stdev,"
            ↪ steps":tr_el_steps},

        "ke_lims" : {"dist":ke_dist,"min":ke_min,"max":ke_max,"mean
```

```
        ↪ ":ke_mean,"stdev":ke_stdev},

      "heat_shield_ap_lims" : {"min_x":heat_shield_ap_min_x,"
        ↪ max_x":heat_shield_ap_max_x,"min_y":
        ↪ heat_shield_ap_min_y,"max_y":heat_shield_ap_max_y,"
        ↪ min_z":heat_shield_ap_min_z,"max_z":
        ↪ heat_shield_ap_max_z}

   }

   return his_config


def params(input_num,input_eV,input_elv,input_azm,D,T,k,model_res
   ↪ ,\
        cf_thickness,cf_transmission,lab_volt_opt):

   # coordinate configuration dictionary
   run_params = {

      "input_num" : input_num,
      "input_eV" : input_eV,
      "input_elv" : input_elv,
      "input_azm" : input_azm,
      "D" : D,
      "T" : T,
      "k" : k,
      "model_res" : model_res,
      "cf_thickness": cf_thickness,
      "cf_transmission": cf_transmission,
      "lab_volt_opt": lab_volt_opt

   }

   return run_params
```

```
# function to import relevant parameters from SimParameters.lua
def import_simparameters(foldername='F:\SO-HIS-
   ↪ SIMION_SARAH_UPDATE\HIS_SIMION',runfile='SimParameters.lua'
   ↪ ):

   # SimParameters.lua file name from .txt filename
   if (runfile != 'SimParameters.lua'):
      runfile = 'SimParameters_'+runfile.split(".")[0]+'.lua'

   # replace slashes
   split_folder = foldername.split('/')
   foldername = os.path.join(*split_folder)

   fullfile = os.path.join(foldername, runfile)
   print("Importing␣SimParameters␣from:",fullfile)

   # read in variables from SimParameters.lua
   df = pd.read_csv(fullfile,header=None,sep="␣=␣|,|{|}",engine='
      ↪ python', \
                  index_col=0,skiprows=1,skipfooter=1,usecols
                     ↪ =[0,1], \
                  skipinitialspace=True)

   # return useful values
   x_center = float(df[1]['x_center'])
   y_center = float(df[1]['y_center'])
   z_center = float(df[1]['z_ap_mid'])
   cf_z = float(df[1]['z_cf'])
   ssd_z = float(df[1]['z_ssd'])
   mcp_start_z = float(df[1]['z_start_mcp'])
   mcp_stop_z = float(df[1]['z_stop_mcp'])
   esa_exit_z = float(df[1]['esa_exit_z'])
   tof_extent_x = float(df[1]['tof_extent_x'])
   tof_extent_y = float(df[1]['tof_extent_y'])
   heat_shield_ap_min_x = float(df[1]['heat_shield_ap_min_x'])
```

```
heat_shield_ap_max_x = float(df[1]['heat_shield_ap_max_x'])
heat_shield_ap_min_y = float(df[1]['heat_shield_ap_min_y'])
heat_shield_ap_max_y = float(df[1]['heat_shield_ap_max_y'])
heat_shield_ap_min_z = float(df[1]['heat_shield_ap_min_z'])
heat_shield_ap_max_z = float(df[1]['heat_shield_ap_max_z'])
z_ap_height = float(df[1]['z_ap_height'])
aperture_out_r = float(df[1]['xy_ap_out_radius'])
aperture_mid_r = float(df[1]['xy_ap_mid_radius'])
aperture_in_r = float(df[1]['xy_ap_in_radius'])
aperture_theta_min = float(df[1]['az_fov_min'])
aperture_theta_max = float(df[1]['az_fov_max'])
ssd_r_min = float(df[1]['ssd_r_min'])
ssd_r_max = float(df[1]['ssd_r_max'])
ssd_az_min = float(df[1]['ssd_az_min'])
ssd_az_max = float(df[1]['ssd_az_max'])
mcp_r_min = float(df[1]['mcp_r_min'])
mcp_r_max = float(df[1]['mcp_r_max'])
mcp_az_min = float(df[1]['mcp_az_min'])
mcp_az_max = float(df[1]['mcp_az_max'])
mcp_r_min = float(df[1]['mcp_r_min'])
mcp_r_max = float(df[1]['mcp_r_max'])
mcp_az_min = float(df[1]['mcp_az_min'])
mcp_az_max = float(df[1]['mcp_az_max'])
esa_exit_r_min = float(df[1]['esa_exit_r_min'])
esa_exit_r_max = float(df[1]['esa_exit_r_max'])
esa_exit_az_min = float(df[1]['esa_exit_az_min'])
esa_exit_az_max = float(df[1]['esa_exit_az_max'])
cf_r_min = float(df[1]['cf_r_min'])
cf_r_max = float(df[1]['cf_r_max'])
cf_az_min = float(df[1]['cf_az_min'])
cf_az_max = float(df[1]['cf_az_max'])
model_res = float(df[1]['model_res'])

source_surface_x = float(df[1]['source_surface_x'])
tr_az_dist = float(df[1]['tr_az_dist'])
tr_az_max = float(df[1]['tr_az_max'])
```

```
tr_az_mean = float(df[1]['tr_az_mean'])
tr_az_min = float(df[1]['tr_az_min'])
tr_az_stdev = float(df[1]['tr_az_stdev'])
tr_az_steps = float(df[1]['tr_az_steps'])
tr_el_dist = float(df[1]['tr_el_dist'])
tr_el_max = float(df[1]['tr_el_max'])
tr_el_mean = float(df[1]['tr_el_mean'])
tr_el_min = float(df[1]['tr_el_min'])
tr_el_stdev = float(df[1]['tr_el_stdev'])
tr_el_steps = float(df[1]['tr_el_steps'])

ke_dist = float(df[1]['ke_dist'])
ke_max = float(df[1]['ke_max'])
ke_mean = float(df[1]['ke_mean'])
ke_min = float(df[1]['ke_min'])
ke_stdev = float(df[1]['ke_stdev'])

his_config = coordinates(x_center, y_center, z_center, cf_z,
   ↪ ssd_z, mcp_start_z,
                   mcp_stop_z, esa_exit_z, tof_extent_x,
                      ↪ tof_extent_y,
                   heat_shield_ap_min_x, heat_shield_ap_max_x,
                   heat_shield_ap_min_y, heat_shield_ap_max_y,
                   heat_shield_ap_min_z, heat_shield_ap_max_z,
                      ↪  z_ap_height,
                   aperture_out_r, aperture_mid_r,
                      ↪ aperture_in_r,
                   aperture_theta_min, aperture_theta_max,
                      ↪ ssd_r_min,
                   ssd_r_max, ssd_az_min, ssd_az_max,
                      ↪ mcp_r_min, mcp_r_max,
                   mcp_az_min, mcp_az_max, esa_exit_r_min,
                   esa_exit_r_max, esa_exit_az_min,
                      ↪ esa_exit_az_max,
                   cf_r_min, cf_r_max, cf_az_min, cf_az_max,
                      ↪ model_res,
                   source_surface_x, tr_az_dist, tr_az_max,
                      ↪ tr_az_mean,
                   tr_az_min, tr_az_stdev, tr_az_steps,
                      ↪ tr_el_dist,
                   tr_el_max, tr_el_mean, tr_el_min,
                      ↪ tr_el_stdev,
                   tr_el_steps, ke_dist, ke_max, ke_mean,
                      ↪ ke_min, ke_stdev)

# run parameters
input_num = float(df[1]['flux'])
if int(df[1]['ke_dist']) == 1:
   input_eV = float(df[1]['ke_mean'])
else:
   input_eV = (float(df[1]['ke_max']) + float(df[1]['ke_min'])
      ↪ )/2
if int(df[1]['tr_el_dist']) == 1:
   input_elv = float(df[1]['tr_el_mean'])
else:
   input_elv = (float(df[1]['tr_el_max']) + float(df[1]['
      ↪ tr_el_min']))/2
if int(df[1]['tr_az_dist']) == 1:
   input_azm = float(df[1]['tr_az_mean'])
else:
   input_azm = (float(df[1]['tr_az_max']) + float(df[1]['
      ↪ tr_az_min']))/2
D = float(df[1]['sim_D'])
T = float(df[1]['sim_T'])
k = float(df[1]['sim_k'])
cf_thickness = float(df[1]['cfoil_thickness'])
cf_transmission = float(df[1]['cfoil_transmission'])
lab_volt_opt = float(df[1]['lab_volt_opt'])


run_params = params(input_num,input_eV,input_elv,input_azm,D,T
   ↪ ,k,model_res,\
                  cf_thickness,cf_transmission,lab_volt_opt)

   return his_config, run_params



# function to read and classify run data
def read_run_data(foldername,runfile,headerlines=8):

   # file i/o
   fullfile = os.path.join(foldername, runfile)
   file = open(fullfile)
   run_contents = file.readlines()[headerlines:]
   columns = ["ion_n","tof","mass","charge","x","y","z","elv","
      ↪ azm", \
           "vx","vy","vz","ke"]

   # his config
   his_config,_ = import_simparameters(foldername,runfile)

   # last ion
   ion_last = get_ion(run_contents, columns, -1)
   # total number of ions in run
   num_ions = ion_last["ion_n"]

   # initialize the final dictionary
   run_data = {

      "gen" : ion_parameters(),
      "aperture" : ion_parameters(),
      "esa_exit" : ion_parameters(),
      "cf_pre" : ion_parameters(),
      "cf_post" : ion_parameters(),
      "mcp_start" : ion_parameters(),
      "mcp_stop" : ion_parameters(),
      "ssd" : ion_parameters(),
      "all_final_splat" : ion_parameters(),
      "non_tof_secondary" : ion_parameters(),
      "dc_gen" : ion_parameters(),
      "tc_gen" : ion_parameters(),
      "start_gen": ion_parameters(),
      "stop_gen": ion_parameters(),
      "ssd_gen": ion_parameters(),
      "cf_post_gen": ion_parameters(),
      "valid_tof": [float('NaN')] * num_ions,
      "actual_tof": [float('NaN')] * num_ions,
      "num_ions": num_ions

      }

   # mark iteration
   ion_n_cur = 0
   ion_counter = 0
   AppendNext = False
   gen_cur = {}
   cf_pre_cur = {}
   cf_post_cur = {}
   start_cur = {}
   stop_cur = {}
   ssd_cur = {}
   DCAppended = False
   TCAppended = False
   for run_count in range(len(run_contents)):
      # each mark as an individual dictionary with names from "
        ↪ columns"
      ion_cur = get_ion(run_contents, columns, run_count)

      # which number mark for the given ion
      if ion_cur["ion_n"] != ion_n_cur:
         ion_n_cur = ion_cur["ion_n"]
         ion_counter = 1
         gen_cur = {}
         cf_pre_cur = {}
         cf_post_cur = {}
         start_cur = {}
         stop_cur = {}
         ssd_cur = {}
         DCAppended = False
         TCAppended = False
      else:
         ion_counter = ion_counter + 1

      # is this a final splat
      FinalSplat = False
      if run_count == len(run_contents)-1:
         FinalSplat = True
      else:
         ion_next = get_ion(run_contents, columns, run_count+1)
         if ion_next["ion_n"] != ion_n_cur:
            FinalSplat = True

      # identify the current mark
      DetectionFound, Splat, detection, ion_cur = \
         identify_mark(ion_cur, ion_counter, FinalSplat,
            ↪ cf_pre_cur, his_config)

      # initialize
      detection_list = []
      if AppendNext:
         detection_list.append("non_tof_secondary")
         AppendNext = False

      # fill in the detection type in all data
      if DetectionFound or Splat == True:

         if DetectionFound and (detection == "gen" \
            or detection == "aperture" \
            or detection == "esa_exit" \
            or detection == "cf_pre" or detection == "cf_post" \
         or detection == "mcp_start" or detection == "mcp_stop
            ↪ " \
         or detection == "ssd") :

         detection_list.append(detection)

         if detection == "gen":
            gen_cur = ion_cur

         if detection == "mcp_start":
            start_cur = ion_cur
            detection_list.append("start_gen")

         if detection == "mcp_stop":
            stop_cur = ion_cur
            detection_list.append("stop_gen")

         if detection == "cf_pre":
            cf_pre_cur = ion_cur

         if detection == "cf_post":
            cf_post_cur = ion_cur
            detection_list.append("cf_post_gen")

         if detection == "ssd":
            ssd_cur = ion_cur
            detection_list.append("ssd_gen")


      # Any last mark in the current ion number (may be
         ↪ various types of marks)
      if FinalSplat:
         detection_list.append("all_final_splat")


      # Undefined marks
```

```
if Splat and not DetectionFound:
   if cf_post_cur: # tof detector miss
      pass
   elif not FinalSplat: # non-tof sescondary
      AppendNext = True

# TC
if start_cur and stop_cur and not DCAppended:
   detection_list.append("dc_gen")
   DCAppended = True
   if ssd_cur and not TCAppended:
      detection_list.append("tc_gen")
      TCAppended = True



# iterate over each parameter to fill into run_data
for detection_count in range(len(detection_list)):

   detection_cur = detection_list[detection_count]

   if detection_cur == "dc_gen" or detection_cur == "
      ↪ tc_gen":
      ion_use = gen_cur
      run_data["valid_tof"][ion_n_cur-1] = \
         2*stop_cur["tof"] - start_cur["tof"] -
            ↪ cf_post_cur["tof"]
      if detection == "tc_gen":
         run_data["actual_tof"][ion_n_cur-1] = \
            ssd_cur["tof"] - cf_post_cur["tof"]

   elif detection_cur == "start_gen" or detection_cur ==
      ↪  "stop_gen" \
      or detection_cur == "ssd_gen" or detection_cur ==
         ↪ "cf_post_gen":
         ion_use = gen_cur
```

```
                else:
                    ion_use = ion_cur

                run_data[detection_cur]["ion_n"].append(ion_use["
                    ↪ ion_n"])
                run_data[detection_cur]["tof"].append(ion_use["tof"])
                run_data[detection_cur]["mass"].append(ion_use["mass"
                    ↪ ])
                run_data[detection_cur]["charge"].append(ion_use["
                    ↪ charge"])
                run_data[detection_cur]["x"].append(ion_use["x"])
                run_data[detection_cur]["y"].append(ion_use["y"])
                run_data[detection_cur]["z"].append(ion_use["z"])
                run_data[detection_cur]["elv"].append(ion_use["elv"])
                run_data[detection_cur]["azm"].append(ion_use["azm"])
                run_data[detection_cur]["vx"].append(ion_use["vx"])
                run_data[detection_cur]["vy"].append(ion_use["vy"])
                run_data[detection_cur]["vz"].append(ion_use["vz"])
                run_data[detection_cur]["ke"].append(ion_use["ke"])
                run_data[detection_cur]["hit_angle"].append(ion_use["
                    ↪ hit_angle"])
                run_data[detection_cur]["r"].append(ion_use["r"])

        else:
            warn_str = 'Unrecognized␣detection␣in␣run␣file␣line␣'+\
                            str(run_count+headerlines+1)+'!'
            warnings.warn(warn_str)


    return run_data


# function to get a mark as an individual dictionary with names
    ↪ from "columns"
def get_ion(run_contents, columns, idx):
    str_vals = run_contents[idx].rstrip().split(',')
    num_vals = [float(v) for v in str_vals]
    num_vals[0] = int(num_vals[0])
    ion = dict(zip(columns,num_vals))
    # shift elevation back by 180 (due to SIMION coordinates)
    ion["elv"] = 180 - ion["elv"]
    # make sure azimuth and elevation are in the range -180 to 180
    # instead of 0 to 360
    if ion["elv"] > 180:
        ion["elv"] = ion["elv"] - 360
    if ion["azm"] > 180:
        ion["azm"] = ion["azm"] - 360

    return ion


# function to return empty ion parameters dictionary
def ion_parameters():

    return {"ion_n" : [],"tof" : [],"mass" : [],"charge" : [], \
            "x" : [],"y" : [],"z" : [], "elv" : [],"azm" : [], \
            "vx" : [],"vy" : [],"vz" : [],"ke" : [], \
            "hit_angle" : [], "r" : []}


# function to identify type of detection
def identify_mark(ion,counter,FinalSplat,cf_pre,his_config):

    detection = ""

    mark_options = ["gen", "aperture", "esa_exit", "cf_pre", "
        ↪ cf_post", \
                    "ssd", "mcp_stop", "mcp_start"]
    guess = ""
    if counter >= 1 and counter <= len(mark_options):
        guess = mark_options[counter-1]
```

```
DetectionFound = False
if guess:
    DetectionFound, r, theta = detection_polar(his_config,
        ↪ guess, ion, FinalSplat, cf_pre)

if not DetectionFound:
    # first_guess = guess
    detection_count = 1 # skip gen because all marks could look
        ↪  like gen
    while not DetectionFound and detection_count < len(
        ↪ mark_options):
        guess = mark_options[detection_count]
        DetectionFound, r, theta = detection_polar(his_config,
            ↪ guess, ion, FinalSplat, cf_pre)
        # if DetectionFound:
        # print("Incorrect guess of",first_guess,"resolved to",
            ↪ guess)
        detection_count = detection_count + 1

if not DetectionFound:
    detection = ""
else:
    detection = guess

ion["hit_angle"] = theta
ion["r"] = r

Splat = False
if FinalSplat or not DetectionFound:
    Splat = True


return DetectionFound, Splat, detection, ion
```

```
# function to determine if the correct detection has been found
def detection_polar(his_config, detection, ion, FinalSplat,
    ↪ cf_pre):

    # initialize result
    DetectionFound = False
    r = float("NaN")
    theta = float("NaN")

    # detections that can't be splats
    skip_condition = False
    if FinalSplat:
        if detection == "gen" or detection == "aperture" or \
            detection == "esa_exit" or detection == "cf_pre" or \
                detection == "cf_post":
                    skip_condition = True

    # cf_pre previously found or trying to find cf_post but pre
        ↪ not yet found
    if (cf_pre and detection == "cf_pre") or (not cf_pre and
        ↪ detection == "cf_post"):
        skip_condition = True

    if not skip_condition:
        # get bounds
        bounds = switch_detection_bounds(detection,his_config)
        if bounds == '-1':
            error_str = 'Invalid␣detection␣requested:␣' + detection
            raise Exception(error_str)

        # calculate polar coordinates
        dety = ion["y"] - his_config["y_center"]

        # gen
        if detection == "gen":
            detxz = ion["z"] - his_config["z_center"]
            r, theta = get_polar(detxz, dety)
            DetectionFound = True

        # all others
        else:
            # in reasonable cartesian bounds
            if not bounds or \
                ((ion["z"] >= bounds.get("z")[0]) \
                and (ion["z"]<= bounds.get("z")[1]) \
                and (ion["x"] >= bounds.get("x")[0]) \
                and (ion["x"] <= bounds.get("x")[1]) \
                and (ion["y"] >= bounds.get("y")[0]) \
                and (ion["y"] <= bounds.get("y")[1])):

                # polar coordinates
                detxz = ion["x"] - his_config["x_center"]
                r, theta = get_polar(detxz, dety)

                if detection == 'mcp_stop' or detection == 'mcp_start
                    ↪ ' \
                    or detection == 'ssd' or detection == 'aperture' \
                    or detection == 'cf_pre' or detection == 'cf_post'
                        ↪ :

                    condition = (ion["z"] >= bounds.get("z")[0]) \
                        and (ion["z"] <= bounds.get("z")[1]) \
                        and (r >= bounds.get("r")[0]) \
                        and (r <= bounds.get("r")[1]) \
                        and (theta >= bounds.get("theta")[0]) \
                        and (theta <= bounds.get("theta")[1])

                else:
                    condition = True

                if condition:
                    DetectionFound = True
    return DetectionFound, r, theta


# returns the instrument bounds for a given detection type
def switch_detection_bounds(detection, his_config):

    switcher = {
        "gen": {}, # no real limits
        "all_final_splat": {},
        "other_final_splat": {},
        "aperture": {"x": his_config["aperture_x_lims"], "y":
            ↪ his_config["aperture_y_lims"], \
                     "z": his_config["aperture_z_lims"], "r":
                        ↪ his_config["aperture_out_r_lims"], \
                     "theta": his_config["aperture_theta_lims"]},
        "esa_exit": {"x": his_config["tof_x_lims"], "y": his_config
            ↪ ["tof_y_lims"], "z": his_config["esa_exit_z_lims"]},
        "cf_pre": {"x": his_config["tof_x_lims"], "y": his_config["
            ↪ tof_y_lims"], "z": his_config["cf_z_lims"], \
                      "r": his_config["cf_r_lims"], "theta":
                         ↪ his_config["cf_theta_lims"]},
        "cf_post": {"x": his_config["tof_x_lims"], "y": his_config[
            ↪ "tof_y_lims"], "z": his_config["cf_z_lims"], \
                      "r": his_config["cf_r_lims"], "theta":
                         ↪ his_config["cf_theta_lims"]},
        "mcp_stop": {"x": his_config["tof_x_lims"], "y": his_config
            ↪ ["tof_y_lims"], "z": his_config["mcp_stop_z_lims"], \
                      "r": his_config["mcp_r_lims"], "theta":
                         ↪ his_config["mcp_theta_lims"]},
        "mcp_start": {"x": his_config["tof_x_lims"], "y":
            ↪ his_config["tof_y_lims"], "z": his_config["
            ↪ mcp_start_z_lims"], \
                      "r": his_config["mcp_r_lims"], "theta":
                         ↪ his_config["mcp_theta_lims"]},
        "ssd": {"x": his_config["tof_x_lims"], "y": his_config["
        ↪ tof_y_lims"], "z": his_config["ssd_z_lims"], \
                "r": his_config["ssd_r_lims"], "theta":
                    ↪ his_config["ssd_theta_lims"]}
    }
    return switcher.get(detection, '-1')



# returns the conditon for mark in current grid
# for use with geometric_factor
def grid_condition(ke,ke_value,elv,elv_value):

    # conditions for current grid
    Ke1 = ke_value > ke["level"] and ke_value <= ke["upper"]
    Elv1 = elv_value > elv["level"] and elv_value <= elv["upper"]
    # include lowest boundary in first grid
    Ke2 = ke["level"] == ke["min"] and ke_value == ke["min"]
    Elv2 = elv["level"] == elv["min"] and elv_value == elv["min"]

    return (Ke1 and Elv1) or (Ke1 and Elv2) or (Ke2 and Elv1) or (
        ↪ Ke2 and Elv2)



# returns the geometric factor for "tc", "dc", or "ssd"
# uses 10x10 grid over source surface for result_type "
    ↪ optimization" and
# 10x10 grid over detections for results type "characterization"
def geometric_factor(run_data,his_config,run_params,gf_type="tc",
    ↪ method="optimization"):

    # determine detection type
    if gf_type != "tc" and gf_type != "dc" and gf_type != "ssd"
        ↪ and gf_type != "cf_post":
        error_str = 'Invalid␣gf_type␣requested.␣Valid␣inputs␣
            ↪ include␣"dc",␣"tc",␣"ssd",␣or␣"cf_post".'
        raise Exception(error_str)
    else:
```

```
   detection = gf_type+"_gen"

# no deteections
if not run_data[detection]["ke"]:
   return 0, 0



# determine the 10x10 grid on the SS or detection
tot_steps = 10
if method == "characterization": # higher resolution, 10x10
   ↪ grid at detection
   ke_min = min(run_data[detection]["ke"])
   ke_max = max(run_data[detection]["ke"])
   # ke_step = (ke_max - ke_min)/tot_steps
   elv_min = min(run_data[detection]["elv"])
   elv_max = max(run_data[detection]["elv"])
   # elv_step = (elv_max - elv_min)/tot_steps
elif method == "optimization": # better comparability, 10x10
   ↪ grid at SS
   ke_min = min(run_data["gen"]["ke"])
   ke_max = max(run_data["gen"]["ke"])
   # ke_step = (ke_max - ke_min)/tot_steps
   elv_min = min(run_data["gen"]["elv"])
   elv_max = max(run_data["gen"]["elv"])
   # elv_step = (elv_max - elv_min)/tot_steps
else:
   error_str = 'Invalid␣result␣type␣requested.␣Valid␣inputs␣
      ↪ include␣"optimizaton"␣or␣"characterization.'
   raise Exception(error_str)

# each grid boundaries
ke_level_list = np.linspace(ke_min,ke_max,tot_steps+1)
elv_level_list = np.linspace(elv_min,elv_max,tot_steps+1)

gf = 0
gf_adj = 0
```

```
# loop through the grid and add up GF for each grid unit
for ke_count in range(len(ke_level_list)-1):

   ke_level = ke_level_list[ke_count]
   ke_upper = ke_level_list[ke_count+1]

   for elv_count in range(len(elv_level_list)-1):

      elv_level = elv_level_list[elv_count]
      elv_upper = elv_level_list[elv_count+1]

      ke = {"min":ke_min, "max":ke_max, "level":ke_level, "
         ↪ upper":ke_upper}
      elv = {"min":elv_min, "max":elv_max, "level":elv_level,
         ↪ "upper":elv_upper}

      ke_list = []
      elv_list = []
      counts = 0
      # find all relevant detections (ssd, dc, tc) in current
         ↪ grid
      for det_count in range (0,len(run_data[detection]["ion_n
         ↪ "])):

         # conditions for current grid
         condition = grid_condition(ke,run_data[detection]["ke
            ↪ "][det_count], \
                              elv,run_data[detection]["elv"][
                                 ↪ det_count])

         # counts at detection (ssd, dc, or tc) in current
            ↪ grid
         if condition:
              counts = counts + 1
              ke_list.append(run_data[detection]["ke"][
                 ↪ det_count])
              elv_list.append(run_data[detection]["elv"][
                 ↪ det_count])

      # adjust counts for carbon foil efficiency
      if run_params["lab_volt_opt"] == 0:
         counts_adj = counts * run_params["cf_transmission"
            ↪ ]**2 # each of the suppression grid and the CF
            ↪ itself have this efficiency
      else:
         counts_adj = counts

      # leave in if transmission adjustment desired
      counts_adj = counts * run_params["cf_transmission"]**2 #
         ↪  each of the suppression grid and the CF itself
         ↪ have this efficiency


      if counts > 0:
         total_ions = 0
         kegens = []
         elvgens = []
         ygens = []
         zgens = []
         # find all relevant generation events in current grid
         for gen_count in range (0,len(run_data["gen"]["ion_n"
            ↪ ])):

            # conditions for current grid
            condition = grid_condition(ke,run_data["gen"]["ke"
                 ↪ ][gen_count], \
                                      elv,run_data["gen"]["elv"][
                                    ↪ gen_count])

            # gen counts in current grid
            if condition:
                 total_ions = total_ions + 1
                 kegens.append(run_data["gen"]["ke"][
                    ↪ gen_count])
                 elvgens.append(run_data["gen"]["elv"][
                    ↪ gen_count])
                 ygens.append(run_data["gen"]["y"][gen_count
                    ↪ ])
                 zgens.append(run_data["gen"]["z"][gen_count
                    ↪ ])


         # SS (gen) surface area for current detection grid
         yextent = max(ygens) - min(ygens)
         zextent = max(zgens) - min(zgens)

         # values for calculation
         area = yextent * zextent * (0.1**2)
         kemean = np.mean(kegens)
         elvmean = np.mean(elvgens)
         kewidth = max(kegens) - min(kegens)
         elvwidth = max(elvgens) - min(elvgens)
         azwidth = 6 # pixel resolution in degrees taken from
            ↪ Stakhiv code

         kde = sns.kdeplot(kegens)
         lines = kde.get_lines()
         if lines:
            kekde,zkde = kde.lines[0].get_data()
            zpeak = max(zkde)
            peakidx = zkde.tolist().index(zpeak)
            e_mode = kekde[peakidx]
         else:
            print('WARNING:␣KDE␣unsuccessful.␣No␣mode␣found.␣
                 ↪ Using␣mean␣energy␣for␣GF␣calculation.')
            e_mode = np.mean(kegens)

         F = (area * kemean * np.cos(elvmean*(np.pi/180.0))**2
                ↪  \
                 * kewidth * elvwidth * np.pi/180.0 * azwidth*np.
                    ↪ pi/180.0) \
                 / (total_ions * e_mode**2)
            gf = gf + (counts * F)
            gf_adj = gf_adj + (counts_adj * F)

   return gf, gf_adj


# returns a list with every element multiplied by n
def multiply(list1, n):
   return [x*n for x in list1]


# returns the total number of ions flown in the run
def num_particles(ion_n):
   return ion_n[len(ion_n)-1]


# returns the polar coordinates r and theta
def get_polar(detxz, dety):

   r = np.sqrt(detxz**2+dety**2)
   if detxz == 0:
      if dety > 0:
         theta = -90
      elif dety < 0:
         theta = 90
   else:
      theta = np.arctan(dety/detxz)*-180./np.pi

   return r, theta


def intersection_ids(list1_ids, list1_ns, list2_ns):
   _, list1indices = intersection3(list1_ns, list2_ns)
   int_ids = [list1_ids[x] for x in list1indices]
   return int_ids


# returns the intersection of two or three lists and the indices
   ↪ in list1
def intersection3(list1, list2, list3=None):

   if list3 is None:
      list3 = list2

   list_out = [x for x in list1 if x in list2]
   list_out = [x for x in list3 if x in list_out]

   list1_inds = [i for (i, val) in enumerate(list1) if val in
      ↪ list_out]

   return list_out, list1_inds


# returns a list of indices of elements of list1 that are not in
   ↪ list2
def disjoint_idxs(list1_ns, list2_ns):
   return [i for (i, val) in enumerate(list1_ns) if val not in
      ↪ list2_ns]


# returns a list of the elements of list1 that are not in list2
def disjoint(list1, list2):
   return [x for x in list1 if x not in list2]


# function to calculate full width half max
def calc_fwhm(xkde,ykde,maxy):
```

```
halfmax = maxy/2

peaky = max(ykde)
peakidx = ykde.tolist().index(peaky)

failure = False
fwhm = -1
x0 = -1
x1 = -1

# left
leftidx_lo = -1
leftidx_hi = -1
for c in range(peakidx,-1,-1):
   if ykde[c] <= halfmax:
      leftidx_lo = c
      leftidx_hi = c+1
      break

if leftidx_lo == -1 or ykde[leftidx_hi] < halfmax:
   failure = True

# right
rightidx_lo = -1
rightidx_hi = -1
for c in range(peakidx,len(ykde)):
   if ykde[c] <= halfmax:
      rightidx_lo = c
      rightidx_hi = c-1
      break

if rightidx_lo == -1 or ykde[rightidx_hi] < halfmax:
   failure = True

if failure == False:
   x0 = np.interp(x=maxy,xp=[ykde[leftidx_lo],ykde[leftidx_hi
          ↪ ]],fp=[xkde[leftidx_lo],xkde[leftidx_hi]])
        x1 = np.interp(x=maxy,xp=[ykde[rightidx_lo],ykde[
          ↪ rightidx_hi]],fp=[xkde[rightidx_lo],xkde[rightidx_hi
          ↪ ]])
        fwhm = x1-x0
    else:
        x0 = -1
        x1 = -1
        fwhm = -1

    return fwhm, x0, x1



# function to determine the bounds of the source surface
def ss_bounds(his_config):

    if his_config["tr_az"]["dist"] == 1: # gaussian
        az_min = his_config["tr_az"]["mean"] - 3*his_config["tr_az"
          ↪ ]["stdev"]
        az_max = his_config["tr_az"]["mean"] + 3*his_config["tr_az"
          ↪ ]["stdev"]
    else:
        az_min = his_config["tr_az"]["min"]
        az_max = his_config["tr_az"]["max"]

    if his_config["tr_el"]["dist"] == 1: # gaussian
        el_min = his_config["tr_el"]["mean"] - 3*his_config["tr_el"
          ↪ ]["stdev"]
        el_max = his_config["tr_el"]["mean"] + 3*his_config["tr_el"
          ↪ ]["stdev"]
    else:
        el_min = his_config["tr_el"]["min"]
        el_max = his_config["tr_el"]["max"]

    d_ss = his_config["source_surface_x"] - his_config["
      ↪ heat_shield_ap_lims"]["min_x"]
    ymin = his_config["heat_shield_ap_lims"]["min_y"] - d_ss *
        ↪ math.tan(az_max*math.pi/180)
    ymax = his_config["heat_shield_ap_lims"]["max_y"] - d_ss *
        ↪ math.tan(az_min*math.pi/180)
    zmin = his_config["heat_shield_ap_lims"]["min_z"] + d_ss *
        ↪ math.tan(el_min*math.pi/180)
    zmax = his_config["heat_shield_ap_lims"]["max_z"] + d_ss *
        ↪ math.tan(el_max*math.pi/180)

    return ymin, ymax, zmin, zmax



# print to terminal and file
def print_out(s, f):

    print(s)
    f.write(s+'\n')



# function to format file names
def filenames(foldername,filename):

    split_extension = filename.split('.')
    plain_filename = split_extension[0]

    outfolder = os.path.join(foldername,"Outputs",plain_filename)
    if not os.path.isdir(outfolder):
        os.makedirs(outfolder)

    fulloutfile = os.path.join(outfolder, plain_filename)

    text_outfile = os.path.join(outfolder, "TextOutput_"+filename)

    return fulloutfile, text_outfile
```

```
# function to determine plot title informaiton
def plot_title(run_params):

   k_str_p, print_num_p = hrn(run_params["input_num"])
   k_str_ke, print_num_ke = hrn(run_params["input_eV"]/1000)
   k_str_elv, print_num_elv = hrn(run_params["input_elv"])
   k_str_azm, print_num_azm = hrn(run_params["input_azm"])

   return "Input␣particles:␣" +print_num_p+k_str_p+ ",␣Mean␣
      ↪ Energy:␣" + \
         print_num_ke+k_str_ke+ "␣keV,␣Mean␣Elevation:␣" + \
         print_num_elv+k_str_elv+ "␣deg,␣Mean:␣Azimuth:␣" + \
         print_num_azm+k_str_azm+ "␣deg"



# returns an empty run values dictionary
# method = "CH" or "OP"
# gf_type = "tc," "dc," "ssd," or "cf_post"
def run_values(gf_type,method):

   return {"KE":[], "Az":[], "El":[], "El_meas":[], "GF":[], "
      ↪ GF_adj":[], "DCR":[], "TCR":[], "method":method, "
      ↪ gf_type":gf_type}



# sorts a run values dictionary
# either Az or El must contain only one value
def sort_run_values(values):

   # check
   if len(values["Az"]) > 1 and len(values["El"]) == 1:
      variable = "Az"
      constant = "El"
   elif len(values["El"]) > 1 and len(values["Az"]) == 1:
      variable = "El"
```

```
        constant = "Az"
    else:
        error_str = "Invalid␣values␣provided.␣Either␣'Az'␣or␣'El'␣
            ↪ must␣be␣a␣list␣of␣more␣than␣one␣element,␣while␣the␣
            ↪ other␣is␣a␣single␣value."
        raise Exception(error_str)

    values["El_meas"] = [x for _,x in sorted(zip(values[variable],
        ↪ values["El_meas"]))]
    values["GF"] = [x for _,x in sorted(zip(values[variable],
        ↪ values["GF"]))]
    values["GF_adj"] = [x for _,x in sorted(zip(values[variable],
        ↪ values["GF_adj"]))]
    values["DCR"] = [x for _,x in sorted(zip(values[variable],
        ↪ values["DCR"]))]
    values["TCR"] = [x for _,x in sorted(zip(values[variable],
        ↪ values["TCR"]))]
    values[variable] = [x for _,x in sorted(zip(values[variable],
        ↪ values[variable]))]

    return values



# plots the gf and dcr/tcr values calculated from SIMION against
    ↪ the values from the lab at SWRI
# variable can be "El" or "Az"
# values["KE"] in keV
def plot_with_lab_values(values,out_foldername,lab_foldername='Y
    ↪ :\SPACE590␣-␣HIS␣-␣TCP\SO-HIS-SIMION_SARAH_UPDATE\
    ↪ GeomFactor\Data',labfile='gfactors_numbers_rev1_columns.csv
    ↪ '):

    # check
    if len(values["Az"]) > 1 and len(values["El"]) == 1:
        variable = "Az"
        constant = "El"
```

```
   values_plot = values[variable]
   values_comp = values[variable]
   FOVmin = -30
   FOVmax = 66
elif len(values["El"]) > 1 and len(values["Az"]) == 1:
   variable = "El"
   constant = "Az"
   values_plot = values["El_meas"]
   values_comp = values["El"]
   FOVmin = -22.5
   FOVmax = 22.5
else:
   error_str = "Invalid␣values␣provided.␣Either␣'Az'␣or␣'El'␣
      ↪ must␣be␣a␣list␣of␣more␣than␣one␣element,␣while␣the␣
      ↪ other␣is␣a␣single␣value."
   raise Exception(error_str)

# method
if values["method"] == "OP":
   method_str = "(Opimization␣Method)"
elif values["method"] == "CH":
   method_str = "(Characterization␣Method)"
else:
   error_str = "Invalid␣method␣provided.␣Valid␣values␣include␣
      ↪ 'OP'␣(optimation)␣or␣'CH'␣(characterization)."
   raise Exception(error_str)

# gf type
if values["gf_type"] == "tc":
   gf_type_str = "triple␣coincidence"
elif values["gf_type"] == "dc":
   gf_type_str = "double␣coincidence"
elif values["gf_type"] == "ssd":
   gf_type_str = "SSD"
elif values["gf_type"] == "cf_post":
   gf_type_str = "post␣carbon␣foil"
```

```
else:
   error_str = 'Invalid␣gf␣type␣requested.␣Valid␣inputs␣
      ↪ include␣"dc",␣"tc",␣"ssd",␣or␣"cf_post".'
   raise Exception(error_str)


# replace slashes
split_folder = lab_foldername.split('/')
lab_foldername = os.path.join(*split_folder)

lab_fullfile = os.path.join(lab_foldername, labfile)
print("Importing␣lab␣values␣from:",lab_fullfile)

# Check if GF was adjusted for CF transmission
if values["GF"] == values["GF_adj"]:
   use_adj_values = False
else:
    use_adj_values = True

# lab values input
labin = open(lab_fullfile,'r')
reader = csv.DictReader(labin)
lab_list = []
for line in reader:
   for k, v in line.items():
      line[k] = float(v)
   line["Az"] = line["Az"] * -1 # lab used opposite
      ↪ orientation
   line["El"] = line["El"] * -1 # lab used opposite
      ↪ orientation
   lab_list.append(line)

# close file
labin.close()

ke_vals = values["KE"][0]
```

```
const_vals = values[constant][0]

# get the relevant plotting data
lab_plot_data = {variable:[],"GF":[],"DCR":[],"TCR":[]}
for lab_count in range(len(lab_list)):
   if lab_list[lab_count]["KE"] == ke_vals and \
      lab_list[lab_count][constant] == const_vals:
         lab_plot_data[variable].append(lab_list[lab_count][
            ↪ variable])
         lab_plot_data["GF"].append(lab_list[lab_count]["GF"])
         lab_plot_data["DCR"].append(lab_list[lab_count]["DCR"
            ↪ ])
         lab_plot_data["TCR"].append(lab_list[lab_count]["TCR"
            ↪ ])

if not lab_plot_data[variable]:
   print("No␣matching␣lab␣data␣found␣for␣KE␣=␣" + str(ke_vals)
      ↪  + "␣keV,␣" + \
       constant + "␣=␣" + str(const_vals) + "␣deg.")
   return

# print ratios
f_out = open(os.path.join(out_foldername,"GF_Ratios.txt"),"w")
constant_str = "KE:␣" + str(ke_vals) + ",␣" + constant + ":␣"
   ↪ + str(const_vals)
print("Ratios␣at␣"+constant_str)
for val_count in range(len(values_comp)):
    comp_cur = values_comp[val_count]
    meas_cur = values_plot[val_count]
    gf_cur = values["GF"][val_count]
    gf_adj_cur = values["GF_adj"][val_count]
    lab_idx_l = [i for i,x in enumerate(lab_plot_data[variable
       ↪ ]) if x == comp_cur]
    for lab_idx_count in range(len(lab_idx_l)):
       lab_idx = lab_idx_l[lab_idx_count]
       lab_gf_cur = lab_plot_data["GF"][lab_idx]
```

```
        print_out("SIMION␣GF␣at␣" +variable+"=␣"+str(round(
            ↪ meas_cur,2))+":␣"+str(round(gf_cur,3)),f_out)
        print_out("SIMION␣adjusted␣GF␣at␣" +variable+"=␣"+str(
            ↪ round(meas_cur,2))+":␣"+str(round(gf_adj_cur,3)),
            ↪ f_out)
        print_out("SWRI␣GF␣at␣" +variable+"=␣"+str(round(
            ↪ comp_cur,2))+":␣"+str(round(lab_gf_cur,3)),f_out)
        print_out("GF␣Ratio␣to␣SWRI:␣"+str(round(gf_cur/
            ↪ lab_gf_cur,2)),f_out)
        print_out("GF␣Adjusted␣Ratio␣to␣SWRI:␣"+str(round(
            ↪ gf_adj_cur/lab_gf_cur,2)),f_out)
f_out.close()

# plotting
if variable == "Az":
    xstr = "Azimuth"
elif variable == "El":
    xstr = "Elevation"
else:
    xstr = variable

# GF plot with linear axis
fig_str = str(round(ke_vals*1000)) + "eV_" + str(round(
    ↪ const_vals)) + constant
figurename = os.path.join(out_foldername, fig_str)

fig = plt.figure()
fig.suptitle("Geometric␣Factor␣Comparison␣at␣"+constant_str+\
            '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
                ↪ ",fontsize=20,y=1.4)
plt.plot(values_plot,values["GF"],'o',color="darkturquoise")
miny = np.min(values["GF"])
maxy = np.max(values["GF"])
if use_adj_values:
    plt.plot(values_plot,values["GF_adj"],'o',color="fuchsia")
    miny = np.min([miny, np.min(values["GF_adj"])])
    maxy = np.max([maxy, np.max(values["GF_adj"])])
plt.plot(lab_plot_data[variable],lab_plot_data["GF"],'o',color
    ↪ ="grey")
miny = np.min([miny, np.min(lab_plot_data["GF"])])
maxy = np.max([maxy, np.max(lab_plot_data["GF"])])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel(xstr+"␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$[cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV]$",fontsize=16)
if use_adj_values:
    legend_str = ['SIMION␣values␣'+method_str,\
                 'CF␣Transmission␣adjusted␣SIMION␣values␣'+
                    ↪ method_str,\
                 'SWRI␣lab␣values']
else:
    legend_str = ['SIMION␣values␣'+method_str,'SWRI␣lab␣values'
       ↪ ]
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.2,1.45),
    ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_lab_comparison_GF.png'
fig.savefig(fullfigurename)

# GF plot with log y-axis
fig = plt.figure()
fig.suptitle("Geometric␣Factor␣Comparison␣at␣"+constant_str+\
             '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
                ↪ ",fontsize=20,y=1.4)
plt.semilogy(values[variable],values["GF"],'o',color="
    ↪ darkturquoise")
miny = np.min(values["GF"])
maxy = np.max(values["GF"])
if use_adj_values:
    plt.semilogy(values[variable],values["GF_adj"],'o',color="
       ↪ fuchsia")
    miny = np.min([miny, np.min(values["GF_adj"])])
```

```
   maxy = np.max([maxy, np.max(values["GF_adj"])])
plt.semilogy(lab_plot_data[variable],lab_plot_data["GF"],'o',
   ↪ color="grey")
miny = np.min([miny, np.min(lab_plot_data["GF"])])
maxy = np.max([maxy, np.max(lab_plot_data["GF"])])
plt.xlim(FOVmin,FOVmax)
# plt.ylim(5e-8,5e-3)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel(xstr+"␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$[cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV]$",fontsize=16)
if use_adj_values:
   legend_str = ['SIMION␣values␣'+method_str,\
            'CF␣Transmission␣adjusted␣SIMION␣values␣'+
               ↪ method_str,\
            'SWRI␣lab␣values']
else:
   legend_str = ['SIMION␣values␣'+method_str,'SWRI␣lab␣values'
      ↪ ]
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.2,1.45),
   ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_lab_comparison_GFlogy.png'
fig.savefig(fullfigurename)

# DCR plot
fig = plt.figure()
fig.suptitle("DCR␣Comparison␣at␣"+constant_str,fontsize=20,y
   ↪ =1.2)
plt.plot(values[variable],values["DCR"],'o',color="gold")
plt.plot(lab_plot_data[variable],lab_plot_data["DCR"],'o',
   ↪ color="cornflowerblue")
miny = np.min([np.min(values["DCR"]), np.min(lab_plot_data["
   ↪ DCR"])])
maxy = np.max([np.max(values["DCR"]), np.max(lab_plot_data["
   ↪ DCR"])])
plt.xlim(FOVmin,FOVmax)
```

```
    plt.ylim(miny-maxy*0.15,maxy*1.15)
    plt.xlabel(xstr+"␣$(degrees)$",fontsize=16)
    plt.ylabel("DCR",fontsize=16)
    plt.legend(['SIMION␣values','SWRI␣lab␣values'],fontsize=14,\
            bbox_to_anchor=(0,1.3),loc="upper␣left")
    plt.show()
    fullfigurename = figurename + '_lab_comparison_DCR.png'
    fig.savefig(fullfigurename)

    # TCR plot
    fig = plt.figure()
    fig.suptitle("TCR␣Comparison␣at␣"+constant_str,fontsize=20,y
       ↪ =1.2)
    plt.plot(values[variable],values["TCR"],'o',color="limegreen")
    plt.plot(lab_plot_data[variable],lab_plot_data["TCR"],'o',
       ↪ color="mediumslateblue")
    miny = np.min([np.min(values["TCR"]), np.min(lab_plot_data["
       ↪ TCR"])])
    maxy = np.max([np.max(values["TCR"]), np.max(lab_plot_data["
       ↪ TCR"])])
    plt.xlim(FOVmin,FOVmax)
    plt.ylim(miny-maxy*0.15,maxy*1.15)
    plt.xlabel(xstr+"␣$(degrees)$",fontsize=16)
    plt.ylabel("TCR",fontsize=16)
    plt.legend(['SIMION␣values','SWRI␣lab␣values'],fontsize=14,\
            bbox_to_anchor=(0,1.3),loc="upper␣left")
    plt.show()
    fullfigurename = figurename + '_lab_comparison_TCR.png'
    fig.savefig(fullfigurename)



# function to create mark panel plots
def mark_panel(run_data,his_config,run_params,foldername,filename
    ↪ ,f_out):

    fulloutfile, _ = filenames(foldername,filename)
```

```
title = plot_title(run_params)



fig = plt.figure(figsize=(8.5,11)) # (w,h)
fig.suptitle('Mark␣Hit␣Plots:\n'+title)
#make outer gridspec
outer = gridspec.GridSpec(8, 1)
#make nested gridspecs
gs1 = gridspec.GridSpecFromSubplotSpec(1, 1, subplot_spec =
   ↪ outer[0])
gs2 = gridspec.GridSpecFromSubplotSpec(7, 1, subplot_spec =
   ↪ outer[1:], hspace = 0)
ax0 = plt.subplot(gs1[0, 0:])
ax1 = plt.subplot(gs2[0, 0:])
ax2 = plt.subplot(gs2[1, 0:])
ax3 = plt.subplot(gs2[2, 0:])
ax4 = plt.subplot(gs2[3, 0:])
ax5 = plt.subplot(gs2[4, 0:])
ax6 = plt.subplot(gs2[5, 0:])
ax7 = plt.subplot(gs2[6, 0:])
fig.tight_layout(rect=[0,0,1,0.99],pad=3) # don't overlap,
   ↪ rect=[lbrt]

# Hit angle range for all plots

t0 = his_config["esa_exit_theta_lims"][0]
t1 = his_config["esa_exit_theta_lims"][1]
t_range = [t0-5,t1+5]

# generation (cartesian)
y_gen = run_data["gen"]["y"]
z_gen = run_data["gen"]["z"]
im0 = ax0.hist2d(y_gen, z_gen, cmin=1, bins=[100,10], \
             cmap=plt.get_cmap('plasma'))
ax0.set_title('Generation␣marks')
fig.colorbar(im0[3], ax=ax0)
```

```
ax0.set_ylabel('Z', size=10)
ax0.set_xlabel('Y', size=10)

        # Outline-----------
# this outline is the minimum SS to cover the back plane of
    ↪ the heat shield aperture area
ymin, ymax, zmin, zmax = ss_bounds(his_config)

ax0.plot([ymin,ymax],[zmin,zmin], color='black', ls='--', lw
    ↪ =2)
ax0.plot([ymin,ymax],[zmax,zmax], color='black', ls='--', lw
    ↪ =2)
ax0.plot([ymin,ymin],[zmin,zmax], color='black', ls='--', lw
    ↪ =2)
ax0.plot([ymax,ymax],[zmin,zmax], color='black', ls='--', lw
    ↪ =2)

ax0.set_xlim([min(ymin,min(y_gen))-2,max(ymax,max(y_gen))+2])
ax0.set_ylim([min(zmin,min(z_gen))-2,max(zmax,max(z_gen))+2])



# aperture (cylindrical)
t0 = his_config["aperture_theta_lims"][0]
t1 = his_config["aperture_theta_lims"][1]
z0 = his_config["aperture_z_lims"][0]
z1 = his_config["aperture_z_lims"][1]

if his_config["tr_az"]["dist"] == 1: # gaussian
    tmin = his_config["tr_az"]["mean"] - 3*his_config["tr_az"][
        ↪ "stdev"]
    tmax = his_config["tr_az"]["mean"] + 3*his_config["tr_az"][
        ↪ "stdev"]
else:
    tmin = his_config["tr_az"]["min"]
    tmax = his_config["tr_az"]["max"]
```

```
# penetrated angles on the aperture will range from min-90 to
   ↪ max+90
tmin = tmin - 90
tmax = tmax + 90
# only care about the angles within range of the aperture
tmin = max(tmin,t0)
tmax = min(tmax,t1)


z_aperture = run_data["aperture"]["z"]
aperture_hit_angles = run_data["aperture"]["hit_angle"]
im1 = ax1.hist2d(aperture_hit_angles, z_aperture, cmin=1, bins
   ↪ =[100,10], \
             range=[t_range,[z0-1,z1+1]],cmap=plt.get_cmap('
                ↪ plasma'))
fig.colorbar(im1[3], ax=ax1)
ax1.set_ylabel('Z␣Aperture', size=10)
ax1.get_xaxis().set_visible(False)


   # Full Outline-----------
ax1.plot([t0,t1],[z0,z0], color='grey', ls='--', lw=2)
ax1.plot([t0,t1],[z1,z1], color='grey', ls='--', lw=2)
ax1.plot([t0,t0],[z0,z1], color='grey', ls='--', lw=2)
ax1.plot([t1,t1],[z0,z1], color='grey', ls='--', lw=2)
   # Expected Outline-----------
ax1.plot([tmin,tmax],[z0,z0], color='black', ls='--', lw=2)
ax1.plot([tmin,tmax],[z1,z1], color='black', ls='--', lw=2)
ax1.plot([tmin,tmin],[z0,z1], color='black', ls='--', lw=2)
ax1.plot([tmax,tmax],[z0,z1], color='black', ls='--', lw=2)


# ESA exit (polar)
t0 = his_config["esa_exit_theta_lims"][0]
t1 = his_config["esa_exit_theta_lims"][1]
r0 = his_config["esa_exit_r_lims"][0]
```

```
r1 = his_config["esa_exit_r_lims"][1]


esa_exit_rs = run_data["esa_exit"]["r"]
esa_exit_hit_angles = run_data["esa_exit"]["hit_angle"]
im2 = ax2.hist2d(esa_exit_hit_angles, esa_exit_rs, cmin=1,
   ↪ bins=[100,10], \
             range=[t_range,[r0-0.5,r1+0.5]],cmap=plt.get_cmap
                ↪ ('plasma'))
fig.colorbar(im2[3], ax=ax2)
ax2.set_ylabel('R␣ESA␣Exit', size=10)
ax2.get_xaxis().set_visible(False)

   # Outline-----------
ax2.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax2.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax2.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax2.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)


# CF pre (polar)
t0 = his_config["cf_theta_lims"][0]
t1 = his_config["cf_theta_lims"][1]
r0 = his_config["cf_r_lims"][0]
r1 = his_config["cf_r_lims"][1]


cf_pre_rs = run_data["cf_pre"]["r"]
cf_pre_hit_angles = run_data["cf_pre"]["hit_angle"]
im3 = ax3.hist2d(cf_pre_hit_angles, cf_pre_rs, cmin=1, bins
   ↪ =[100,10], \
             range=[t_range,[r0-1,r1+1]],cmap=plt.get_cmap('
                ↪ plasma'))
fig.colorbar(im3[3], ax=ax3)
ax3.set_ylabel('R␣CF␣Pre', size=10)
ax3.get_xaxis().set_visible(False)
```

```
    # Outline-----------
ax3.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax3.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax3.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax3.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)



# CF post (polar)
t0 = his_config["cf_theta_lims"][0]
t1 = his_config["cf_theta_lims"][1]
r0 = his_config["cf_r_lims"][0]
r1 = his_config["cf_r_lims"][1]



cf_post_rs = run_data["cf_post"]["r"]
cf_post_hit_angles = run_data["cf_post"]["hit_angle"]
im4 = ax4.hist2d(cf_post_hit_angles, cf_post_rs, cmin=1, bins
    ↪ =[100,10], \
             range=[t_range,[r0-1,r1+1]],cmap=plt.get_cmap('
                ↪ plasma'))
fig.colorbar(im4[3], ax=ax4)
ax4.set_ylabel('R␣CF␣Post', size=10)
ax4.get_xaxis().set_visible(False)

    # Outline-----------
ax4.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax4.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax4.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax4.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)



# SSD (polar)
t0 = his_config["ssd_theta_lims"][0]
t1 = his_config["ssd_theta_lims"][1]
r0 = his_config["ssd_r_lims"][0]
```

```
r1 = his_config["ssd_r_lims"][1]


ssd_rs = run_data["ssd"]["r"]
ssd_hit_angles = run_data["ssd"]["hit_angle"]
im5 = ax5.hist2d(ssd_hit_angles, ssd_rs, cmin=1, bins
   ↪ =[100,10], \
             range=[t_range,[r0-1,r1+1]],cmap=plt.get_cmap('
                ↪ plasma'))
fig.colorbar(im5[3], ax=ax5)
ax5.set_ylabel('R␣SSD', size=10)
ax5.get_xaxis().set_visible(False)

   # Outline-----------
ax5.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax5.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax5.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax5.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)


# start (polar)
t0 = his_config["mcp_theta_lims"][0]
t1 = his_config["mcp_theta_lims"][1]
r0 = his_config["mcp_r_lims"][0]
r1 = his_config["mcp_r_lims"][1]


mcp_start_rs = run_data["mcp_start"]["r"]
mcp_start_hit_angles = run_data["mcp_start"]["hit_angle"]
im6 = ax6.hist2d(mcp_start_hit_angles, mcp_start_rs, cmin=1,
   ↪ bins=[100,10], \
             range=[t_range,[r0-3,r1+3]],cmap=plt.get_cmap('
                ↪ plasma'))
fig.colorbar(im6[3], ax=ax6)
ax6.set_ylabel('R␣MCP␣Start', size=10)
ax6.get_xaxis().set_visible(False)
```

```
    # Outline-----------
ax6.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax6.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax6.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax6.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)



# stop (polar)
t0 = his_config["mcp_theta_lims"][0]
t1 = his_config["mcp_theta_lims"][1]
r0 = his_config["mcp_r_lims"][0]
r1 = his_config["mcp_r_lims"][1]



mcp_stop_rs = run_data["mcp_stop"]["r"]
mcp_stop_hit_angles = run_data["mcp_stop"]["hit_angle"]
im7 = ax7.hist2d(mcp_stop_hit_angles, mcp_stop_rs, cmin=1,
   ↪ bins=[100,10], \
             range=[t_range,[r0-3,r1+3]],cmap=plt.get_cmap('
                ↪ plasma'))
fig.colorbar(im7[3], ax=ax7)
ax7.set_ylabel('R␣MCP␣Stop', size=10)
ax7.set_xlabel('Hit␣Angle', size=10)

    # Outline-----------
ax7.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax7.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax7.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax7.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)



# display
plt.show()

# save
```

```
    fullfigurename = fulloutfile + '_HPanel.png'
    fig.savefig(fullfigurename)



# function to plot the final panel plot for voltage analysis
def voltage_analysis_panel(his_config,run_params,num_flown,\
                           ssd_keV,ssd_elv,input_keV,ssd_rs,
                              ↪ ssd_hit_angles,\
                           foldername,filename,f_out):

    fulloutfile, _ = filenames(foldername,filename)
    title = plot_title(run_params)

    fig = plt.figure(figsize=(8.5,11)) # (w,h)
    fig.suptitle('Voltage␣Analysis␣Plots:\n'+title)
    grid = gridspec.GridSpec(5,4)
    ax0 = plt.subplot(grid[0, 0:])
    ax1 = plt.subplot(grid[1, 0:])
    ax2 = plt.subplot(grid[2, 0:])
    ax3 = plt.subplot(grid[3, 0:3])
    ax4 = plt.subplot(grid[3, 3])
    ax5 = plt.subplot(grid[4, 0:3])
    fig.tight_layout(rect=[0,0,1,0.99],pad=3) # don't overlap,
       ↪ rect=[lbrt]


    # 2D Elevation-Energy Histogram
    im0 = ax0.hist2d(ssd_keV, ssd_elv, cmin=1, bins=10, \
                    cmap=plt.get_cmap('plasma'))


    # calculate features
    tot_transmission = np.nansum(im0[0])
    peak_transmission = np.nanmax(im0[0])
    # plot features
    # text plotted later after E0 and T0 computed, axes adjusted
    # labels
    ax0.set_title('SSD␣Generation␣Elevation-Energy␣Plot')
    fig.colorbar(im0[3], ax=ax0)
    ax0.set_ylabel('Elevation', size=10)
    ax0.set_xlabel('Energy', size=10)

    # energy histogram
    im1 = ax1.hist(ssd_keV, bins='auto')
    ax1.set_title('SSD␣generation␣energies␣[keV]')
    ax1.set_xlim([his_config["ke_lims"]["min"]/1000,his_config["
       ↪ ke_lims"]["max"]/1000])
    x1,x2 = ax1.get_xlim()
    y1,y2 = ax1.get_ylim()
    x = x1+(x2-x1)*0.02
    y = y2-(y2-y1)*0.15
    ydec = (y2-y1)*0.15
    E0, sigma = stats.norm.fit(ssd_keV)
    if sigma > 0:
       xfit = np.linspace(im1[1][0], im1[1][-1], 100)
       yfit = multiply(stats.norm.pdf(xfit,E0,sigma),max(im1[0])*\
                      math.sqrt(2*math.pi)*sigma)
       ax1.plot(xfit,yfit,color='orange',linewidth=2)
       ax1.plot([E0,E0],[0,max(yfit)],color='orange',linewidth=2)
       ax1.text(x,y,'E0='+str(round(E0,2))+'␣keV',fontsize=12)
       spline = UnivariateSpline(xfit, yfit-max(yfit)/2, s=0)
       try:
          r1, r2 = spline.roots()
          ax1.plot([r1,r2],[max(yfit)/2,max(yfit)/2],color='orange
             ↪ ',linewidth=2)
          fwhm_E0 = r2 - r1
       except:
          print('WARNING:␣Insufficient␣data␣to␣determine␣FWHM.')
          fwhm_E0 = float('NaN')
       y = y - ydec
       ax1.text(x,y,'fwhm='+str(round(fwhm_E0,2))+'␣keV',fontsize
          ↪ =12)
```

```
    fwhm_perc_E0 = fwhm_E0*100/E0
    y = y - ydec
    ax1.text(x,y,str(round(fwhm_perc_E0,1))+'%␣fwhm␣perc.␣E0')
else:
    print_out("Inadequate␣statistics␣for␣fitting␣energy␣
       ↪ histogram.", f_out)
    fwhm_perc_E0 = np.nan



# elevation histogram
im2 = ax2.hist(ssd_elv, bins='auto')
ax2.set_title('SSD␣generation␣elevation␣angles␣[deg]')
ax2.set_xlim([his_config["tr_el"]["min"],his_config["tr_el"]["
    ↪ max"]])
x1,x2 = ax2.get_xlim()
y1,y2 = ax2.get_ylim()
x = x1+(x2-x1)*0.02
y = y2-(y2-y1)*0.15
ydec = (y2-y1)*0.15
T0, sigma = stats.norm.fit(ssd_elv)
if sigma > 0:
    xfit = np.linspace(im2[1][0], im2[1][-1], 100)
    yfit = multiply(stats.norm.pdf(xfit,T0,sigma),max(im2[0])*
                   math.sqrt(2*math.pi)*sigma)
    ax2.plot(xfit,yfit,color='orange',linewidth=2)
    ax2.plot([T0,T0],[0,max(yfit)],color='orange',linewidth=2)
    ax2.text(x,y,'Theta0='+str(round(T0,4))+DEGREE_SIGN,
       ↪ fontsize=12)
    spline = UnivariateSpline(xfit, yfit-max(yfit)/2, s=0)
    try:
        r1, r2 = spline.roots()
        ax2.plot([r1,r2],[max(yfit)/2,max(yfit)/2],color='orange
           ↪ ',linewidth=2)
        fwhm_T0 = r2 - r1
    except:
        print('WARNING:␣Insufficient␣data␣to␣determine␣FWHM.')
```

```
            fwhm_T0 = float('NaN')
        y = y - ydec
        ax2.text(x,y,'fwhm='+str(round(fwhm_T0,4))+DEGREE_SIGN,
            ↪ fontsize=12)
    else:
        print_out("Inadequate␣statistics␣for␣fitting␣elevation␣
            ↪ histogram.", f_out)
        fwhm_T0 = np.nan


    # 2D Elevation-Energy Histogram Center and Distance
    dist = distance.euclidean((E0,T0),(input_keV,run_params["
        ↪ input_elv"]))
    ax0.plot(E0, T0, 'rx', markersize=12)
    ax0.plot(input_keV, run_params["input_elv"], 'ro', fillstyle='
        ↪ none', markersize=12)
    x1,x2 = ax0.get_xlim()
    y1,y2 = ax0.get_ylim()
    ax0.set_xlim([min(x1,his_config["ke_lims"]["min"]/1000), max(
        ↪ x2,his_config["ke_lims"]["max"]/1000)])
    ax0.set_ylim([min(y1,his_config["tr_el"]["min"]), max(y2,
        ↪ his_config["tr_el"]["max"])])
    # text
    fish_dt_text = 'D=' + str(run_params["D"]) + ',␣T=' + str(
        ↪ run_params["T"])
    transmission_text = 'Peak=' + str(peak_transmission) + \
                        '␣Total=' + str(tot_transmission)
    transmission_perc_text = 'Peak=' + str(round(peak_transmission
        ↪ *100/num_flown,3))\
                          + '%' + '␣Total=' \
                          + str(round(tot_transmission*100/num_flown
                              ↪ ,3)) + '%'
    dist_text = 'distance=' + str(round(dist,3))
    center_text = 'center=('+str(round(E0,2))+',␣'+str(round(T0,2)
        ↪ )+')'
    x1,x2 = ax0.get_xlim()
```

```
y1,y2 = ax0.get_ylim()
x = x1+(x2-x1)*0.02
y = y2-(y2-y1)*0.15
ydec = (y2-y1)*0.15
ax0.text(x,y,fish_dt_text,color='orange')
y = y - ydec
ax0.text(x,y,transmission_text,color='black')
y = y - ydec
ax0.text(x,y,transmission_perc_text,color='black')
y = y - ydec
ax0.text(x,y,dist_text,color='red')
y = y - ydec
ax0.text(x,y,center_text,color='red')


# SSD hit plot
t0 = his_config["ssd_theta_lims"][0]
t1 = his_config["ssd_theta_lims"][1]
r0 = his_config["ssd_r_lims"][0]
r1 = his_config["ssd_r_lims"][1]
im3 = ax3.hist2d(ssd_hit_angles,ssd_rs,bins=[100,10],cmin=1,\
             range=[[t0-5,t1+5],[r0-5,r1+5]],cmap=plt.get_cmap
                ↪ ('cividis'))
# SSD outline-----------
ax3.plot([t0,t1],[r0,r0], color='black', ls='--', lw=2)
ax3.plot([t0,t1],[r1,r1], color='black', ls='--', lw=2)
ax3.plot([t0,t0],[r0,r1], color='black', ls='--', lw=2)
ax3.plot([t1,t1],[r0,r1], color='black', ls='--', lw=2)
ax3.set_ylabel('R', size=10)
ax3.set_xlabel('hit␣angle', size=10)
# labels
ax3.set_title('SSD␣Hit␣Plot')
# horizontal sum
r_lo = r0-5
r_hi = r1+5
im4 = ax4.hist(ssd_rs,bins='auto',range=[r_lo,r_hi],
   ↪ orientation='horizontal',density=True)
ax4.set_title('Horizontal␣Sum')
ssd_rs_range = [i for i in ssd_rs if i >= r_lo and i <= r_hi]
maxy = max(im4[0])
midx = np.mean(ssd_rs_range)
ax4.plot([0,maxy],[midx,midx],color='red',linewidth=2)
r_kde = sns.kdeplot(y=ssd_rs, ax=ax4,clip=[r_lo, r_hi],
   ↪ linewidth=2)

lines = r_kde.get_lines()
fwhm_r = float('NaN')
x0_fwhm = float('NaN')
x1_fwhm = float('NaN')
if len(lines)>1:
   ykde,xkde = lines[1].get_data()
   fwhm_r,x0_fwhm,x1_fwhm = calc_fwhm(xkde,ykde,maxy)

x1,x2 = ax4.get_xlim()
y1,y2 = ax4.get_ylim()
x = x1+(x2-x1)*0.02
y = y2-(y2-y1)*0.15
ydec = (y2-y1)*0.15
ax4.plot([maxy/2,maxy/2],[x0_fwhm,x1_fwhm],color='red',
   ↪ linewidth=2)
ax4.text(x,y,'r0='+str(round(midx,2))+'␣mm',fontsize=12)
y = y - ydec
ax4.text(x,y,'fwhm='+str(round(fwhm_r,2))+'␣mm',fontsize=12)

# vertical sum
t_lo = t0-5
t_hi = t1+5
im5 = ax5.hist(ssd_hit_angles,bins='auto',range=[t_lo,t_hi],
   ↪ density=True)
ax5.set_title('Vertical␣Sum')
ssd_ts_range = [i for i in ssd_hit_angles if i >= t_lo and i
   ↪ <= t_hi]
```

```
maxy = max(im5[0])
midx = np.mean(ssd_ts_range)
t_kde = sns.kdeplot(ssd_hit_angles,ax=ax5,clip=[t_lo, t_hi],
   ↪ linewidth=2)

lines = t_kde.get_lines()
fwhm_t = float('NaN')
x0_fwhm = float('NaN')
x1_fwhm = float('NaN')
if lines:
   xkde,ykde = t_kde.get_lines()[0].get_data()
   fwhm_t,x0_fwhm,x1_fwhm = calc_fwhm(xkde,ykde,maxy)
x1,x2 = ax5.get_xlim()
y1,y2 = ax5.get_ylim()
x = x1+(x2-x1)*0.02
y = y2-(y2-y1)*0.15
ydec = (y2-y1)*0.15
ax5.plot([midx,midx],[0,maxy],color='red',linewidth=2)
ax5.text(x,y,'hit0='+str(round(midx,2))+DEGREE_SIGN,fontsize
   ↪ =12)
ax5.plot([x0_fwhm,x1_fwhm],[maxy/2,maxy/2],color='red',
   ↪ linewidth=2)
y = y - ydec
ax5.text(x,y,'fwhm='+str(round(fwhm_t,2))+DEGREE_SIGN,fontsize
   ↪ =12)



# display
plt.show()

# save
fullfigurename = fulloutfile + '_VPanel.png'
fig.savefig(fullfigurename)

# return the measured elevation center
return E0,T0,peak_transmission,tot_transmission,fwhm_perc_E0,
      ↪ fwhm_T0,fwhm_r,fwhm_t



# function to create a third degree polynomial for curve fitting
   ↪ the GF data
def poly2(x,a,b,c):
   return a*x**2 + b*x + c

# function to create a third degree polynomial for curve fitting
   ↪ the GF data
def poly3(x,a,b,c,d):
   return a*x**3 + b*x**2 + c*x + d

# function to create a fourth degree polynomial for curve fitting
   ↪  the GF data
def poly4(x,a,b,c,d,e):
   return a*x**4 + b*x**3 + c*x**2 + d*x + e



# plots GFs against elevation at a given azimuth, scales by the
   ↪ SIMION to lab ratio,
# and finds a curve fitting
# values["KE"] in keV
def plot_and_fit_geometric_factors(values,factor,out_foldername):

   # check
   if not all(x == values["KE"][0] for x in values["KE"]):
      error_str = "Exactly␣one␣unique␣KE␣value␣must␣be␣provided."
      raise Exception(error_str)
   else:
      ke = values["KE"][0]

   if not all(x == values["Az"][0] for x in values["Az"]):
      error_str = "Exactly␣one␣unique␣Azimuth␣value␣must␣be␣
         ↪ provided."
      raise Exception(error_str)
```

```
else:
    az = values["Az"][0]

if len(values["El_meas"]) != len(values["GF"]) or len(values["
    ↪ El_meas"]) != len(values["GF_adj"]):
    error_str = "Number␣of␣GF␣and␣adjusted␣GF␣values␣must␣match
        ↪ ␣number␣of␣provided␣measured␣elevations."

# method
if values["method"] == "OP":
    method_str = "(Opimization␣Method)"
elif values["method"] == "CH":
    method_str = "(Characterization␣Method)"
else:
    error_str = "Invalid␣method␣provided.␣Valid␣values␣include␣
        ↪ 'OP'␣(optimation)␣or␣'CH'␣(characterization)."
    raise Exception(error_str)



# gf type
if values["gf_type"] == "tc":
    gf_type_str = "triple␣coincidence"
elif values["gf_type"] == "dc":
    gf_type_str = "double␣coincidence"
elif values["gf_type"] == "ssd":
    gf_type_str = "SSD"
elif values["gf_type"] == "cf_post":
    gf_type_str = "post␣carbon␣foil"
else:
    error_str = 'Invalid␣gf␣type␣requested.␣Valid␣inputs␣
        ↪ include␣"dc",␣"tc",␣"ssd",␣or␣"cf_post".'
    raise Exception(error_str)



# Check if GF was adjusted for CF transmission
if values["GF"] == values["GF_adj"]:
    use_adj_values = False
else:
     use_adj_values = True



# SWRI factor adjusted numbers
gf_adj_factor = [x/factor for x in values["GF_adj"]]



f_out = open(os.path.join(out_foldername,"Fit_Parameters.txt")
   ↪ ,"w")
# curve fit

# 4th degree polynomial
popt, pcov = curve_fit(poly4,values["El_meas"],gf_adj_factor)
a, b, c, d, e = popt
perr = np.sqrt(np.diag(pcov))
residuals = gf_adj_factor - poly4(np.array(values["El_meas"])
   ↪ ,*popt)
ss_res = np.sum(residuals**2)
ss_tot = np.sum((gf_adj_factor-np.mean(gf_adj_factor))**2)
r_squared = 1 - (ss_res / ss_tot)
eq_str = 'y␣=␣(' + str("{:.3e}".format(a)) + ')␣*␣x^4␣+␣(' +
   ↪ str("{:.3e}".format(b)) + \
       ')␣*␣x^3␣+␣(' + str("{:.3e}".format(c)) + ')␣*␣x^2␣+␣('
          ↪  + str("{:.3e}".format(d)) + \
       ')␣*␣x␣+␣(' + str("{:.3e}".format(e)) + ')'
err_str = 'Fitting␣Error:␣a:␣' + str("{:.3e}".format(perr[0]))
   ↪  + ',␣b:␣' + \
        str("{:.3e}".format(perr[1])) + ',␣c:␣' + str("{:.3e}"
           ↪ .format(perr[2])) + \
        ',␣d:␣' + str("{:.3e}".format(perr[3])) + ',␣e:␣' +
           ↪ str("{:.3e}".format(perr[4]))
rsq_str = 'R␣Squared:␣' + str(r_squared)
print_out('Fourth␣Degree␣Polynomial␣Fit:',f_out)
print_out(eq_str, f_out)
print_out(err_str, f_out)
print_out(rsq_str, f_out)
fitx4 = np.linspace(np.min(values["El_meas"]),np.max(values["
    ↪ El_meas"]),num=1000,endpoint=True)
fity4 = poly4(fitx4,a,b,c,d,e)

# 3rd degree polynomial
popt, pcov = curve_fit(poly3,values["El_meas"],gf_adj_factor)
a, b, c, d = popt
perr = np.sqrt(np.diag(pcov))
residuals = gf_adj_factor - poly3(np.array(values["El_meas"])
    ↪ ,*popt)
ss_res = np.sum(residuals**2)
ss_tot = np.sum((gf_adj_factor-np.mean(gf_adj_factor))**2)
r_squared = 1 - (ss_res / ss_tot)
eq_str = 'y␣=␣(' + str("{:.3e}".format(a)) + ')␣*␣x^3␣+␣(' +
    ↪ str("{:.3e}".format(b)) + \
        ')␣*␣x^2␣+␣(' + str("{:.3e}".format(c)) + ')␣*␣x␣+␣(' +
            ↪ str("{:.3e}".format(d)) + ')'
err_str = 'Fitting␣Error:␣a:␣' + str("{:.3e}".format(perr[0]))
    ↪ + ',␣b:␣' + \
        str("{:.3e}".format(perr[1])) + ',␣c:␣' + str("{:.3e}"
            ↪ .format(perr[2])) + \
        ',␣d:␣' + str("{:.3e}".format(perr[3]))
rsq_str = 'R␣Squared:␣' + str(r_squared)
print_out('\nThird␣Degree␣Polynomial␣Fit:',f_out)
print_out(eq_str, f_out)
print_out(err_str, f_out)
print_out(rsq_str, f_out)
fitx3 = np.linspace(np.min(values["El_meas"]),np.max(values["
    ↪ El_meas"]),num=1000,endpoint=True)
fity3 = poly3(fitx3,a,b,c,d)

# 2nd degree polynomial
popt, pcov = curve_fit(poly2,values["El_meas"],gf_adj_factor)
a, b, c = popt
```

```
perr = np.sqrt(np.diag(pcov))
residuals = gf_adj_factor - poly2(np.array(values["El_meas"])
    ↪ ,*popt)
ss_res = np.sum(residuals**2)
ss_tot = np.sum((gf_adj_factor-np.mean(gf_adj_factor))**2)
r_squared = 1 - (ss_res / ss_tot)
eq_str = 'y␣=␣(' + str("{:.3e}".format(a)) + ')␣*␣x^2␣+␣(' +
    ↪ str("{:.3e}".format(b)) + \
        ')␣*␣x␣+␣(' + str("{:.3e}".format(c)) + ')'
err_str = 'Fitting␣Error:␣a:␣' + str("{:.3e}".format(perr[0]))
    ↪  + ',␣b:␣' + \
        str("{:.3e}".format(perr[1])) + ',␣c:␣' + str("{:.3e}"
            ↪ .format(perr[2]))
rsq_str = 'R␣Squared:␣' + str(r_squared)
print_out('\nSecond␣Degree␣Polynomial␣Fit:',f_out)
print_out(eq_str, f_out)
print_out(err_str, f_out)
print_out(rsq_str, f_out)
fitx2 = np.linspace(np.min(values["El_meas"]),np.max(values["
    ↪ El_meas"]),num=1000,endpoint=True)
fity2 = poly2(fitx2,a,b,c)

f_out.close()

# GF plot

# elevation plotting parameters
FOVmin = -22.5
FOVmax = 22.5

constant_str = "KE:␣" + str(ke) + "␣keV,␣Azimuth:␣" + str(az)
    ↪ + "␣deg"

fig_str = str(round(ke*1000)) + "eV_" + str(round(az)) + "Az"
figurename = os.path.join(out_foldername, fig_str)
```

```
# comparison plot, 4th degree polynomial fit
fig = plt.figure()
fig.suptitle("Geometric␣Factors␣at␣"+constant_str+\
            '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
               ↪ ",fontsize=20,y=1.4)
plt.plot(values["El_meas"],values["GF"],'o',color="
   ↪ lightseagreen")
miny = np.min(values["GF"])
maxy = np.max(values["GF"])
if use_adj_values:
   plt.plot(values["El_meas"],values["GF_adj"],'o',color="
      ↪ mediumvioletred")
   miny = np.min([miny, np.min(values["GF_adj"])])
   maxy = np.max([maxy, np.max(values["GF_adj"])])
plt.plot(values["El_meas"],gf_adj_factor,'o',color="goldenrod"
   ↪ )
miny = np.min([miny,np.min(gf_adj_factor)])
maxy = np.max([maxy,np.max(gf_adj_factor)])
plt.plot(fitx4,fity4,color="darkgoldenrod")
miny = np.min([miny,np.min(fity4)])
maxy = np.max([maxy,np.max(fity4)])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel("Elevation␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$(cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV)$",fontsize=16)
if use_adj_values:
   legend_str = ['SIMION␣values␣'+method_str,\
                'SIMION␣values␣adjusted␣for␣CF␣transmission␣'+
                   ↪ method_str,\
                'SIMION␣values␣adjusted␣for␣CF␣transmission␣and␣
                   ↪ lab␣factor␣'+method_str,\
                'Fourth␣degree␣polynomial␣fit']
else:
   legend_str = ['SIMION␣values␣'+method_str,'SIMION␣values␣
      ↪ adjusted␣for␣lab␣factor␣'+method_str,'Fourth␣degree␣
      ↪ polynomial␣fit']
```

```
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.5,1.46),
   ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_GF_plot4.png'
fig.savefig(fullfigurename)


# Only fitted data plot, 4th degree polynomial fit
fig = plt.figure()
fig.suptitle("Adjusted␣Geometric␣Factors␣at␣"+constant_str+\
          '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
             ↪ ",fontsize=20,y=1.4)
plt.plot(values["El_meas"],gf_adj_factor,'o',color="goldenrod"
   ↪ )
miny = np.min(gf_adj_factor)
maxy = np.max(gf_adj_factor)
plt.plot(fitx4,fity4,color="darkgoldenrod")
miny = np.min([miny,np.min(fity4)])
maxy = np.max([maxy,np.max(fity4)])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel("Elevation␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$(cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV)$",fontsize=16)
if use_adj_values:
   legend_str = ['SIMION␣values␣adjusted␣for␣CF␣transmission␣
      ↪ and␣lab␣factor␣'+method_str,\
             'Fourth␣degree␣polynomial␣fit']
else:
   legend_str = ['SIMION␣values␣adjusted␣for␣lab␣factor␣'+
      ↪ method_str,'Fourth␣degree␣polynomial␣fit']
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.5,1.45),
   ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_GF_fit_plot4.png'
fig.savefig(fullfigurename)
```

```
# comparison plot, 3rd degree polynomial fit
fig = plt.figure()
fig.suptitle("Geometric␣Factors␣at␣"+constant_str+\
          '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
             ↪ ",fontsize=20,y=1.4)
plt.plot(values["El_meas"],values["GF"],'o',color="
   ↪ lightseagreen")
miny = np.min(values["GF"])
maxy = np.max(values["GF"])
if use_adj_values:
   plt.plot(values["El_meas"],values["GF_adj"],'o',color="
      ↪ mediumvioletred")
   miny = np.min([miny, np.min(values["GF_adj"])])
   maxy = np.max([maxy, np.max(values["GF_adj"])])
plt.plot(values["El_meas"],gf_adj_factor,'o',color="goldenrod"
   ↪ )
miny = np.min([miny,np.min(gf_adj_factor)])
maxy = np.max([maxy,np.max(gf_adj_factor)])
plt.plot(fitx3,fity3,color="darkgoldenrod")
miny = np.min([miny,np.min(fity3)])
maxy = np.max([maxy,np.max(fity3)])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel("Elevation␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$(cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV)$",fontsize=16)
if use_adj_values:
   legend_str = ['SIMION␣values␣'+method_str,\
             'SIMION␣values␣adjusted␣for␣CF␣transmission␣'+
                ↪ method_str,\
             'SIMION␣values␣adjusted␣for␣CF␣transmission␣and␣
                ↪ lab␣factor␣'+method_str,\
             'Third␣degree␣polynomial␣fit']
else:
   legend_str = ['SIMION␣values␣'+method_str,'SIMION␣values␣
      ↪ adjusted␣for␣lab␣factor␣'+method_str,'Third␣degree␣
```

```
        ↪ polynomial␣fit']
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.5,1.46),
    ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_GF_plot3.png'
fig.savefig(fullfigurename)

# Only fitted data plot, 3rd degree polynomial fit
fig = plt.figure()
fig.suptitle("Adjusted␣Geometric␣Factors␣at␣"+constant_str+\
          '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
            ↪ ",fontsize=20,y=1.4)
plt.plot(values["El_meas"],gf_adj_factor,'o',color="goldenrod"
    ↪ )
miny = np.min(gf_adj_factor)
maxy = np.max(gf_adj_factor)
plt.plot(fitx3,fity3,color="darkgoldenrod")
miny = np.min([miny,np.min(fity3)])
maxy = np.max([maxy,np.max(fity3)])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel("Elevation␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$(cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV)$",fontsize=16)
if use_adj_values:
    legend_str = ['SIMION␣values␣adjusted␣for␣CF␣transmission␣
        ↪ and␣lab␣factor␣'+method_str,\
             'Third␣degree␣polynomial␣fit']
else:
    legend_str = ['SIMION␣values␣adjusted␣for␣lab␣factor␣'+
        ↪ method_str,'Third␣degree␣polynomial␣fit']
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.5,1.45),
    ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_GF_fit_plot3.png'
fig.savefig(fullfigurename)
```

```
# comparison plot, 2nd degree polynomial fit
fig = plt.figure()
fig.suptitle("Geometric␣Factors␣at␣"+constant_str+\
          '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
             ↪ ",fontsize=20,y=1.4)
plt.plot(values["El_meas"],values["GF"],'o',color="
   ↪ lightseagreen")
miny = np.min(values["GF"])
maxy = np.max(values["GF"])
if use_adj_values:
   plt.plot(values["El_meas"],values["GF_adj"],'o',color="
      ↪ mediumvioletred")
   miny = np.min([miny, np.min(values["GF_adj"])])
   maxy = np.max([maxy, np.max(values["GF_adj"])])
plt.plot(values["El_meas"],gf_adj_factor,'o',color="goldenrod"
   ↪ )
miny = np.min([miny,np.min(gf_adj_factor)])
maxy = np.max([maxy,np.max(gf_adj_factor)])
plt.plot(fitx2,fity2,color="darkgoldenrod")
miny = np.min([miny,np.min(fity2)])
maxy = np.max([maxy,np.max(fity2)])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel("Elevation␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$(cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV)$",fontsize=16)
if use_adj_values:
   legend_str = ['SIMION␣values␣'+method_str,\
             'SIMION␣values␣adjusted␣for␣CF␣transmission␣'+
                ↪ method_str,\
             'SIMION␣values␣adjusted␣for␣CF␣transmission␣and␣
                ↪ lab␣factor␣'+method_str,\
             'Second␣degree␣polynomial␣fit']
else:
   legend_str = ['SIMION␣values␣'+method_str,'SIMION␣values␣
      ↪ adjusted␣for␣lab␣factor␣'+method_str,'Second␣degree␣
```

```
        ↪ polynomial␣fit']
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.5,1.46),
    ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_GF_plot2.png'
fig.savefig(fullfigurename)

# Only fitted data plot, 2nd degree polynomial fit
fig = plt.figure()
fig.suptitle("Adjusted␣Geometric␣Factors␣at␣"+constant_str+\
             '\nSIMION␣GF␣measured␣for␣'+gf_type_str+"␣detections
                ↪ ",fontsize=20,y=1.4)
plt.plot(values["El_meas"],gf_adj_factor,'o',color="goldenrod"
    ↪ )
miny = np.min(gf_adj_factor)
maxy = np.max(gf_adj_factor)
plt.plot(fitx2,fity2,color="darkgoldenrod")
miny = np.min([miny,np.min(fity2)])
maxy = np.max([maxy,np.max(fity2)])
plt.xlim(FOVmin,FOVmax)
plt.ylim(miny-maxy*0.15,maxy*1.15)
plt.xlabel("Elevation␣$(degrees)$",fontsize=16)
plt.ylabel("GF␣$(cm^2␣\cdot␣sr␣\cdot␣eV␣/␣eV)$",fontsize=16)
if use_adj_values:
    legend_str = ['SIMION␣values␣adjusted␣for␣CF␣transmission␣
        ↪ and␣lab␣factor␣'+method_str,\
                 'Second␣degree␣polynomial␣fit']
else:
    legend_str = ['SIMION␣values␣adjusted␣for␣lab␣factor␣'+
        ↪ method_str,'Second␣degree␣polynomial␣fit']
plt.legend(legend_str,fontsize=14,bbox_to_anchor=(-0.5,1.45),
    ↪ loc="upper␣left")
plt.show()
fullfigurename = figurename + '_GF_fit_plot2.png'
fig.savefig(fullfigurename)
```

APPENDIX G

HIS Python Plotting and Analysis Code

The SO-HIS post-processing analysis and plotting Python code is included below.

Listing G.1: Code used for post-processing plotting and analysis of SO-HIS SIMION runs.

```
 # -*- coding: utf-8 -*-
"""
Created on Thu Mar 19 11:52:06 2020

@author: saraylet
"""

__author__ = 'Sarah␣Spitzer;␣saraylet@umich.edu'



# import pandas as pd
import numpy as np
# import matplotlib
import matplotlib.pyplot as plt
import os
import HISdatalib as his
import csv
import time
# DEBUG
import pdb

# METHOD = "OP" # optimization
METHOD = "CH" # characterization

```

```
# leave blank to use all .txt files in FOLDERNAME
FILENAMES = []

# For user input foldername, do not create this variable
# FOLDERNAME = os.path.join('')

LAB_FILENAME = 'gfactors_numbers_rev1_columns.csv'
LAB_FOLDERNAME = os.path.join('..','GeomFactor','Data')

# main
def main():


   if 'FOLDERNAME' not in globals():
      FOLDERNAME = input ('Enter␣foldername:␣')

   print('folder:', FOLDERNAME)
   if FILENAMES:
      list_names = FILENAMES
   else:
      list_names = os.listdir(FOLDERNAME)

   # create csv to store comvined output analysis results
   output_foldername = os.path.join(FOLDERNAME,"Outputs")
   if not os.path.isdir(output_foldername):
      os.makedirs(output_foldername)
   csv_filename = os.path.join(output_foldername,"
      ↪ combined_analysis_values.csv")
   csv_out = open(csv_filename,"w",newline='')
   csv_writer = csv.writer(csv_out)

   # process each file/run
   run_results = {"ke_list":[], "run_values_list_cf":[], "
      ↪ run_values_list_tc":[]}
   for filename in list_names:
      start_time = time.time()
      if filename.endswith('.txt'):
         print('file:', filename)

         # process data
         his_config,run_params = his.import_simparameters(
             ↪ FOLDERNAME,filename)
         cf_post_gf, cf_post_gf_adj, tc_gf_ch, tc_gf_ch_adj, dcr,
             ↪  tcr, T0 = \
            process_single(his_config,run_params,FOLDERNAME,
                ↪ filename,csv_writer,start_time)

         # store results
         ke = run_params["input_eV"]/1000
         azm = run_params["input_azm"]
         elv = run_params["input_elv"]

         run_results = append_run_results(run_results,ke,azm,elv,
            ↪ T0,dcr,tcr,\
                                  cf_post_gf,cf_post_gf_adj,\
                                  tc_gf_ch,tc_gf_ch_adj)

         print_time(time.time()-start_time)

   # close csv file
   csv_out.close()

   # create combined GF plot for all processed runs if taken at
      ↪ the same energy and azimuth
   if len(run_results["run_values_list_cf"]) == 1: # only one KE
      run_values_cf_cur = run_results["run_values_list_cf"][0]
      ke_list_cur = run_values_cf_cur["KE"]
      az_list_cur = run_values_cf_cur["Az"]
      if all(x == ke_list_cur[0] for x in ke_list_cur) and \
         all(x == az_list_cur[0] for x in az_list_cur):
            his.plot_and_fit_geometric_factors(run_values_cf_cur,
              ↪ his.SIMION_TO_SWRI_FACTOR,output_foldername)

   # compare results to lab data
   for idx_ke in range(len(run_results["ke_list"])):

      run_values_cf_cur = run_results["run_values_list_cf"][
         ↪ idx_ke]
      run_values_tc_cur = run_results["run_values_list_tc"][
         ↪ idx_ke]
      ke = run_results["ke_list"][idx_ke]


      # constant elevation
      elv_list = set(run_values_cf_cur["El"])
      for elv in elv_list:
         if run_values_cf_cur["El"].count(elv) > 1:
           run_plot_with_lab_values(run_values_cf_cur,ke,elv,"El
                ↪ ","Az",output_foldername)


      print_time(time.time()-start_time)

      # constant azimuth
      azm_list = set(run_values_cf_cur["Az"])
      for azm in azm_list:
         if run_values_cf_cur["Az"].count(azm) > 1:
           run_plot_with_lab_values(run_values_cf_cur,ke,azm,"Az
                ↪ ","El",output_foldername)

      print_time(time.time()-start_time)


# function to add values to a run_result
def append_run_results(run_results,ke,azm,elv,T0,dcr,tcr,
   ↪ cf_post_gf,cf_post_gf_adj,\
                 tc_gf_ch,tc_gf_ch_adj):
```

```
    if ke not in run_results["ke_list"]:
        run_results["ke_list"].append(ke)
        run_results["run_values_list_cf"].append(his.run_values("
            ↪ cf_post",METHOD))
        run_results["run_values_list_tc"].append(his.run_values("tc
            ↪ ","CH"))
        idx = -1
    else:
        idx = run_results["ke_list"].index(ke)

    run_results = fill_results_list(run_results,idx,ke,azm,elv,T0,
       ↪ dcr,tcr,\
                                    cf_post_gf,cf_post_gf_adj,"
                                       ↪ run_values_list_cf")
    run_results = fill_results_list(run_results,idx,ke,azm,elv,T0,
       ↪ dcr,tcr,\
                                    tc_gf_ch,tc_gf_ch_adj,"
                                       ↪ run_values_list_tc")

    return run_results

# helper function for above
def fill_results_list(run_results,idx,ke,azm,elv,T0,dcr,tcr,gf,
    ↪ gf_adj,list_name):

    run_results[list_name][idx]["KE"].append(ke)
    run_results[list_name][idx]["Az"].append(azm)
    run_results[list_name][idx]["El"].append(elv)
    run_results[list_name][idx]["El_meas"].append(T0)
    run_results[list_name][idx]["DCR"].append(dcr)
    run_results[list_name][idx]["TCR"].append(tcr)
    run_results[list_name][idx]["GF"].append(gf)
    run_results[list_name][idx]["GF_adj"].append(gf_adj)

    return run_results
```

```
# function to set up and run his.plot_with_lab_values for either
    ↪ Az or El varying
def run_plot_with_lab_values(run_values,ke,var,var_str,const_str,
    ↪ output_foldername):

    print("Checking␣Lab␣comparison␣for␣KE␣=␣" + str(ke) + "keV,␣"
        ↪ + \
          var_str + "␣=␣" + str(var) + "␣deg.")
    run_values_comp = his.run_values(run_values["gf_type"],
        ↪ run_values["method"])
    run_values_comp["KE"].append(ke)
    run_values_comp[var_str].append(var)
    for idx in range(len(run_values[var_str])):
        if run_values[var_str][idx] == var:
            run_values_comp["El_meas"].append(run_values["El_meas"][
                ↪ idx])
            run_values_comp[const_str].append(run_values[const_str][
                ↪ idx])
            run_values_comp["DCR"].append(run_values["DCR"][idx])
            run_values_comp["TCR"].append(run_values["TCR"][idx])
            run_values_comp["GF"].append(run_values["GF"][idx])
            run_values_comp["GF_adj"].append(run_values["GF_adj"][
                ↪ idx])
    run_values_comp = his.sort_run_values(run_values_comp)
    his.plot_with_lab_values(run_values_comp,output_foldername,
        ↪ LAB_FOLDERNAME,LAB_FILENAME)


# process each text file/run in the folder/set of runs
def process_single(his_config,run_params,foldername,filename,
    ↪ csv_writer,start_time):

    fulloutfile, text_outfile = his.filenames(foldername,filename)
    # open text outfile
```

```
f_out = open(text_outfile,"w")

# Read in data
print("Reading␣in␣run␣data.")
run_data = his.read_run_data(foldername,filename)
print("Completed␣reading␣in␣run␣data.")
print_time(time.time()-start_time)

# get various parameters

# number of particles
num_flown = run_data["num_ions"]
num_k_str,num_print_num = his.hrn(num_flown)
input_k_str,input_print_num = his.hrn(run_params["input_num"])
his.print_out('Number␣of␣particles␣flown:␣' + num_print_num +
    ↪ num_k_str + \
     '␣(' + input_print_num + input_k_str + '␣input)', f_out)

      # total splats
splat_ns = run_data["all_final_splat"]["ion_n"]
splat_k_str,splat_print_num = his.hrn(len(splat_ns))
his.print_out('Number␣of␣total␣final␣splats:␣' +
    ↪ splat_print_num + splat_k_str, \
       f_out)



# print mean of energies at generation
eV_gen = run_data["gen"]["ke"]
keV_gen = his.multiply(eV_gen,1/1000)
keV_gen_mean = round(np.mean(keV_gen), 3)
his.print_out('Mean␣of␣particle␣energies␣at␣generation:␣' +
     str(keV_gen_mean) + '␣[keV]' + \
     '␣(' + str(run_params["input_eV"]/1000) + '␣[keV]␣input)'
       ↪ , f_out)
```

```
# create a histogram of energies at generation
plt.hist(keV_gen, bins='auto')
plt.title('Particle␣Energies␣at␣Generation␣[keV]')
fullfigurename = fulloutfile + '_eVgen.png'
plt.savefig(fullfigurename)
plt.show()
# create a histogram of elevations at generation
elv_gen = run_data["gen"]["elv"]
plt.hist(elv_gen, bins='auto')
plt.title('Particle␣Elevations␣at␣Generation␣[deg]')
fullfigurename = fulloutfile + '_elvgen.png'
plt.savefig(fullfigurename)
plt.show()
# create a 2D histogram of energy and elevation at generation
plt.hist2d(keV_gen, elv_gen, cmin=1, bins=10, \
        cmap=plt.get_cmap('plasma'))
plt.title('Particle␣Elevations␣vs.␣Energy␣at␣Generation␣[deg,␣
   ↪ keV]')
fullfigurename = fulloutfile + '_fishgen.png'
plt.savefig(fullfigurename)
plt.show()
# create a histogram of azimuths at generation
azm_gen = run_data["gen"]["azm"]
plt.hist(azm_gen, bins='auto')
plt.title('Particle␣Azimuths␣at␣Generation␣[deg]')
fullfigurename = fulloutfile + '_azgen.png'
plt.savefig(fullfigurename)
plt.show()

# print mean elevation angle at generation
elv_gen_mean = round(np.mean(elv_gen), 3)
his.print_out('Mean␣of␣particle␣elevation␣angles␣at␣generation
   ↪ :␣' +
    str(elv_gen_mean) + '␣[deg]' + \
    '␣(' + str(run_params["input_elv"]) + '␣[deg]␣input)',
       ↪ f_out)
```

```
# print mean azimuth angle at generation
azm_gen_mean = round(np.mean(azm_gen), 3)
his.print_out('Mean␣of␣particle␣azimuth␣angles␣at␣generation:␣
    ↪ ' +
      str(azm_gen_mean) + '␣[deg]'+ \
      '␣(' + str(run_params["input_azm"]) + '␣[deg]␣input)',
        ↪ f_out)


# print the number of particles that mark at the aperture
num_aperture = len(run_data["aperture"]["ion_n"])
his.print_out('Number␣of␣particles␣that␣cross␣the␣aperture:␣'
    ↪ + \
        str(num_aperture), f_out)


# Print how many particles survive the ESA / reach cf_pre
num_cf_pre = len(run_data["cf_pre"]["ion_n"])
his.print_out('Number␣of␣ions␣that␣survive␣the␣ESA␣/␣reach␣the
    ↪ ␣carbon␣foil:␣' +
      str(num_cf_pre), f_out)
his.print_out('Percentage␣of␣ions␣that␣survive␣the␣ESA␣/␣reach
    ↪ ␣the␣carbon␣foil:␣' +
      str((num_cf_pre/num_flown)*100) + '␣%', f_out)


# print number of particles to survive carbon foil
num_cf_post = len(run_data["cf_post"]["ion_n"])
cf_survival_ratio = 0
if num_cf_pre > 0:
    cf_survival_ratio = num_cf_post/num_cf_pre
his.print_out('Number␣of␣ions␣that␣survive␣the␣carbon␣foil:␣'
    ↪ +
      str(num_cf_post), f_out)
```

```
his.print_out('Percentage␣of␣ions␣that␣survive␣the␣carbon␣foil
   ↪ :␣' +
     str((num_cf_post/num_flown)*100) + '␣%', f_out)
his.print_out('Percentage␣of␣ions␣that␣reach␣the␣carbon␣foil␣
   ↪ to␣survive␣the␣' + \
     'carbon␣foil:␣' + str((cf_survival_ratio)*100) + '␣%',
        ↪ f_out)

# physical double coincidences (MCP start and stop with or
   ↪ without SSD)
# and triple coincidences (MCP start and stop plus SSD)
num_dc = len(run_data["dc_gen"]["ion_n"])
his.print_out('Number␣of␣Double␣Coincidences:␣'+str(num_dc),
   ↪ f_out)
num_tc = len(run_data["tc_gen"]["ion_n"])
his.print_out('Number␣of␣Triple␣Coincidences:␣'+str(num_tc),
   ↪ f_out)



# all SSD triggers
ssd_elv = run_data["ssd_gen"]["elv"] # elevation at generation
ssd_eV = run_data["ssd_gen"]["ke"] # energy at generation
ssd_rs = run_data["ssd"]["r"] # r at SSD
ssd_hit_angles = run_data["ssd"]["hit_angle"] # hit angle at
   ↪ SSD
num_ssd = len(ssd_elv)
ssd_keV = his.multiply(ssd_eV, 1/1000)
input_keV = run_params["input_eV"] / 1000
his.print_out('Number␣of␣ions␣that␣reach␣the␣SSD:␣' +
     str(num_ssd), f_out)
his.print_out('Percentage␣of␣ions␣that␣reach␣the␣SSD:␣' +
     str((num_ssd/num_flown)*100) + '␣%', f_out)
if len(run_data["ssd"]["ion_n"])>0:
   ssd_spot_height = np.max(run_data["ssd"]["r"]) - np.min(
      ↪ run_data["ssd"]["r"])
   ssd_spot_angle_width = np.max(run_data["ssd"]["hit_angle"])
```

```
        ↪  - np.min(run_data["ssd"]["hit_angle"])
    r_mid = (np.max(run_data["ssd"]["r"])+np.min(run_data["ssd"
        ↪ ]["r"]))/2
    ssd_spot_width = (ssd_spot_angle_width/360) * (2*np.pi*
        ↪ r_mid)
    cell_width = (r_mid*2*np.pi*(29+65-6)/360 - (0.25*29))/30
    num_cells = np.floor(ssd_spot_width/cell_width)
    if num_cells == 0:
        num_cells = 1
    affected_area_perc_lo = 100 * ((num_cells-1)*0.25)/(
        ↪ ssd_spot_width)
    affected_area_perc_hi = 100 * (num_cells*0.25)/(
        ↪ ssd_spot_width)
    print('SSD␣SPOT␣INFO␣FOR␣APPROXIMATION␣USE␣ONLY:')
    print('SSD␣spot␣height:␣' + str(ssd_spot_height) + 'mm')
    print('SSD␣spot␣width:␣' + str(ssd_spot_width) + 'mm')
    print('Approximate␣lower␣percentage␣of␣SSD␣spot␣affected␣by
        ↪ ␣0.25mm␣gaps:␣' + \
          str(affected_area_perc_lo) + '␣%')
    print('Approximate␣upper␣percentage␣of␣SSD␣spot␣affected␣by
        ↪ ␣0.25mm␣gaps:␣' + \
          str(affected_area_perc_hi) + '␣%')


# SSD GF
ssd_gf_op,ssd_gf_op_adj = his.geometric_factor(run_data,
    ↪ his_config,run_params,"ssd","optimization")
his.print_out('SSD␣GF␣(optimization␣method):␣' + str(ssd_gf_op
    ↪ ), f_out)
his.print_out('SSD␣GF␣adjusted␣(optimization␣method):␣' + str(
    ↪ ssd_gf_op_adj), f_out)
ssd_gf_ch, ssd_gf_ch_adj = his.geometric_factor(run_data,
    ↪ his_config,run_params,"ssd","characterization")
his.print_out('SSD␣GF␣(characterization␣method):␣' + str(
    ↪ ssd_gf_ch), f_out)
his.print_out('SSD␣GF␣adjusted␣(characterization␣method):␣' +
    ↪ str(ssd_gf_ch_adj), f_out)

# TC GF
tc_gf_op, tc_gf_op_adj = his.geometric_factor(run_data,
    ↪ his_config,run_params,"tc","optimization")
his.print_out('TC␣GF␣(optimization␣method):␣' + str(tc_gf_op),
    ↪  f_out)
his.print_out('TC␣GF␣adjusted␣(optimization␣method):␣' + str(
    ↪ tc_gf_op_adj), f_out)
tc_gf_ch, tc_gf_ch_adj = his.geometric_factor(run_data,
    ↪ his_config,run_params,"tc","characterization")
his.print_out('TC␣GF␣(characterization␣method):␣' + str(
    ↪ tc_gf_ch), f_out)
his.print_out('TC␣GF␣adjusted␣(characterization␣method):␣' +
    ↪ str(tc_gf_ch_adj), f_out)



# CF Post GF
cf_post_gf_op, cf_post_gf_op_adj = his.geometric_factor(
    ↪ run_data,his_config,run_params,"cf_post","optimization")
his.print_out('CF␣Post␣GF␣(optimization␣method):␣' + str(
    ↪ cf_post_gf_op), f_out)
his.print_out('CF␣Post␣GF␣adjusted␣(optimization␣method):␣' +
    ↪ str(cf_post_gf_op_adj), f_out)
cf_post_gf_ch, cf_post_gf_ch_adj = his.geometric_factor(
    ↪ run_data,his_config,run_params,"cf_post","
    ↪ characterization")
his.print_out('CF␣Post␣GF␣(characterization␣method):␣' + str(
    ↪ cf_post_gf_ch), f_out)
his.print_out('CF␣Post␣GF␣adjusted␣(characterization␣method):␣
    ↪ ' + str(cf_post_gf_ch_adj), f_out)

if METHOD == "OP":
   cf_post_gf = cf_post_gf_op
   cf_post_gf_adj = cf_post_gf_op_adj
else:
    cf_post_gf = cf_post_gf_ch
    cf_post_gf_adj = cf_post_gf_ch_adj


# start
num_start = len(run_data["mcp_start"]["ion_n"])
his.print_out('Number␣of␣ions␣that␣reach␣the␣Start␣Detector:␣'
   ↪  +
      str(num_start), f_out)
# stop
num_stop = len(run_data["mcp_stop"]["ion_n"])
his.print_out('Number␣of␣ions␣that␣reach␣the␣Stop␣Detector:␣'
   ↪ +
      str(num_stop), f_out)

# DCR
if num_cf_post == 0:
    dcr = 0
else:
    dcr = (num_start * num_stop) / num_cf_post
his.print_out('DCR␣estimation␣(start*stop/I0)):␣' +
      str(dcr), f_out)

# TCR
if num_cf_post == 0:
    tcr = 0
else:
    tcr = (num_start * num_stop * num_ssd) / (num_cf_post**2)
his.print_out('TCR␣estimation␣(start*stop*ssd/I0^2)):␣' +
      str(tcr), f_out)


# timing
print_time(time.time()-start_time)
```

```
print("Starting␣voltage␣calibration␣panel␣plot.")
# voltage calibration panel plot
T0 = np.nan
if not (len(ssd_keV)==0 or len(ssd_elv)==0):
    E0,T0,peak_transmission,tot_transmission,fwhm_perc_E0,
        ↪ fwhm_T0,fwhm_r,fwhm_t \
        = his.voltage_analysis_panel(his_config,run_params,
            ↪ num_flown,
                            ssd_keV,ssd_elv,input_keV,ssd_rs,
                            ssd_hit_angles,foldername,filename,
                                ↪ f_out)
else:
    his.print_out('No␣particles␣hit␣the␣SSD', f_out)


# timing
print_time(time.time()-start_time)


print("starting␣mark␣panel␣plot.")
# marks panel plot
his.mark_panel(run_data,his_config,run_params,foldername,
    ↪ filename,f_out)


# write results to csv file
# target E/q (keV, 0.2f), target elv (0.1f), target azm (0.1f)
    ↪ , D (0.3f), T (0.3f),
# measured energy (keV, 0.2f), measured elevation (0.2f), peak
    ↪  trasnmission,
# total transmission, E/q acceptance (0.1f), elv acceptance
    ↪ (0.2f),
# spot r fwhm (0.2f), spot th fwhm (0.2f)
if len(run_data["ssd"]["ion_n"])>0:
    csv_writer.writerow([round(run_params["input_eV"]/1000,2),
        ↪ \
                          round(run_params["input_elv"],1), \
                          round(run_params["input_azm"],1), \
                          round(run_params["D"],3), \
                          round(run_params["T"],3), \
                          round(E0,2), round(T0,2), peak_transmission
                             ↪ , \
                          tot_transmission, round(fwhm_perc_E0,1), \
                          round(fwhm_T0,2), round(fwhm_r,2), round(
                             ↪ fwhm_t,2)])
    else:
        csv_writer.writerow([round(run_params["input_eV"]/1000,2),
           ↪ \
                          round(run_params["input_elv"],1), \
                          round(run_params["input_azm"],1), \
                          round(run_params["D"],3), \
                          round(run_params["T"],3), \
                          np.nan, np.nan, np.nan, np.nan, \
                          np.nan, np.nan, np.nan, np.nan])



    # close text outfile
    f_out.close()

    return cf_post_gf, cf_post_gf_adj, tc_gf_ch, tc_gf_ch_adj, dcr
        ↪ , tcr, T0



# print timing information in hr format
def print_time(timing):

    if timing > 60*60*24:
        print("Elapsed␣time:",str(round(timing/60/60/24,2)),"days")
    elif timing > 60*60:
        print("Elapsed␣time:",str(round(timing/60/60,2)),"hr")
    elif timing > 60:
        print("Elapsed␣time:",str(round(timing/60,2)),"min")
```

```
   else:
      print("Elapsed␣time:",str(round(timing,2)),"sec")


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

BIBLIOGRAPHY

Ajello, J. M. 1978, Astrophysical Journal, 222, 1068, doi: 10.1086/156224

Alfvén, H. 1942, Nature, 150, 405, doi: 10.1038/150405d0

Alfven, H. 1977, Reviews of Geophysics and Space Physics, 15, 271, doi: 10.1029/RG015i003p00271

Allegrini, F., Ebert, R. W., & Funsten, H. O. 2016, Journal of Geophysical Research: Space Physics, 121, 3931, doi: 10.1002/2016JA022570

Andrews, G. B., Zurbuchen, T. H., Mauk, B. H., et al. 2007, Space Science Reviews, 131, 523, doi: 10.1007/s11214-007-9272-5

Bame, S. J., Hundhausen, A. J., Asbridge, J. R., & Strong, I. B. 1968, The Astronomical Journal Supplement, 73, 55

Bertaux, J. L., & Blamont, J. E. 1971, Astronomy and Astrophysics, 11, 200

Bower, J. S., Moebius, E., Berger, L., et al. 2019, Journal of Geophysical Research (Space Physics), 124, 6418, doi: 10.1029/2019JA026781

Bzowski, M., Kubiak, M. A., Hłond, M., et al. 2014, Astronomy and Astrophysics, 569, 1, doi: 10.1051/0004-6361/201424127

Bzowski, M., Swaczyna, P., Kubiak, M. A., et al. 2015, Astrophysical Journal, Supplement Series, 220, doi: 10.1088/0067-0049/220/2/28

Cameron, A. E., & Eggers, D. F., J. 1948, Review of Scientific Instruments, 19, 605, doi: 10.1063/1.1741336

Cane, H. V., & Richardson, I. G. 2003, Journal of Geophysical Research (Space Physics), 108, 1156, doi: 10.1029/2002JA009817

Chalov, S. V. 2014, Monthly Notices of the Royal Astronomical Society: Letters, 443, L25, doi: 10.1093/mnrasl/slu074

Chalov, S. V., & Fahr, H. J. 1998, Astronomy and Astrophysics, 335, 746

—. 1999, Solar Physics, 187, 123, doi: 10.1023/A:1005176229228

Collinson, G. A., Dorelli, J. C., Avanov, L. A., et al. 2012, Review of Scientific Instruments, 83, doi: `10.1063/1.3687021`

Dalaudier, F., Bertaux, J. L., Kurt, V. G., & Mironova, E. N. 1984, Astronomy and Astrophysics, 134, 171

Drews, C., Berger, L., Taut, A., Peleikis, T., & Wimmer-Schweingruber, R. F. 2015, Astronomy and Astrophysics, 575, A97, doi: `10.1051/0004-6361/201425271`

Drews, C., Berger, L., Wimmer-Schweingruber, R. F., et al. 2012, Journal of Geophysical Research (Space Physics), 117, A09106, doi: `10.1029/2012JA017746`

ESA. 2019, Solar Orbiter at IABG. `https://www.esa.int/ESA_Multimedia/Images/2019/10/Solar_Orbiter_at_IABG`

Fahr, H. J. 1968, Astrophysics and Space Science, 2, 474, doi: `10.1007/BF02175923`

Fisk, L. A., & Gloeckler, G. 2006, Astrophysical Journal, 640, L79, doi: `10.1086/503293`

Frisch, P. C., Bzowski, M., Livadiotis, G., et al. 2013, Science, 341, 1080, doi: `10.1126/science.1239925`

Frisch, P. C., Bzowski, M., Drews, C., et al. 2015, Astrophysical Journal, 801, doi: `10.1088/0004-637X/801/1/61`

Galvin, A. B., Kistler, L. M., Popecki, M. A., et al. 2008, Space Science Reviews, 136, 437, doi: `10.1007/s11214-007-9296-x`

Gershman, D. J., Fisk, L. A., Gloeckler, G., et al. 2014, Astrophysical Journal, 788, 124, doi: `10.1088/0004-637X/788/2/124`

Gershman, D. J., Gloeckler, G., Gilbert, J. A., et al. 2013, Journal of Geophysical Research (Space Physics), 118, 1389, doi: `10.1002/jgra.50227`

Gilbert, J. A. 2008, PhD thesis, University of Michigan

Gilbert, J. A., Gershman, D. J., Gloeckler, G., et al. 2014, Review of Scientific Instruments, 85, 91301, doi: `10.1063/1.4894694`

Gilbert, J. A., Zurbuchen, T. H., & Battel, S. 2016, Journal of Geophysical Research (Space Physics), 121, 5097, doi: `10.1002/2016JA022381`

Gloeckler, G., & Geiss, J. 1996, Nature, 381, 210, doi: `10.1038/381210a0`

—. 1998, Space Science Reviews, 86, 127, doi: `10.1023/A:1005019628054`

Gloeckler, G., & Hsieh, K. C. 1979, Nuclear Instruments and Methods, 165, 537, doi: `10.1016/0029-554X(79)90636-0`

Gloeckler, G., Schwadron, N. A., Fisk, L. A., & Geiss, J. 1995, Geophysical Research Letters, 22, 2665, doi: `10.1029/95GL02480`

Gloeckler, G., Ipavich, F. M., Studemann, W., et al. 1985, IEEE Transactions on Geoscience and Remote Sensing, 23, 234, doi: 10.1109/TGRS.1985.289519

Gloeckler, G., Geiss, J., Balsiger, H., et al. 1992, Astronomy and Astrophysics, 92, 267

Gloeckler, G., Cain, J., Ipavich, F. M., et al. 1998, Space Science Reviews, 86, 497, doi: 10.1023/A:1005036131689

Gloeckler, G., Möbius, E., Geiss, J., et al. 2004, Astronomy and Astrophysics, 426, 845, doi: 10.1051/0004-6361:20035768

Hansteen, V. H. 2009a, in Heliophysics: Plasma Physics of the Local Cosmos, ed. C. J. Schrijver & G. L. Siscoe (Cambridge University Press), 195–224

—. 2009b, in Heliophysics: Plasma Physics of the Local Cosmos, ed. C. J. Schrijver & G. L. Siscoe (Cambridge University Press), 225–255

Heber, B., & Potgieter, M. S. 2006, Space Science Reviews, 127, 117, doi: 10.1007/s11214-006-9085-y

HelioWeb, N. 2017, Heliocentric Trajectories for Selected Spacecraft, Planets, and Comets. https://omniweb.gsfc.nasa.gov/coho/helios/planet.html

Ipavich, F. M., Lundgren, R. A., Lambird, B. A., & Gloeckler, G. 1978, Nuclear Instruments and Methods, 154, 291, doi: 10.1016/0029-554X(78)90412-3

Isenberg, P. A. 1997, Journal of Geophysical Research: Space Physics, 102, 4719, doi: 10.1029/96JA03671

Jian, L., Russell, C. T., Luhmann, J. G., & Skoug, R. M. 2006, Solar Physics, 239, 393, doi: 10.1007/s11207-006-0133-2

Kasper, J. C., Klein, K. G., Lichko, E., et al. 2021, Physical Review Letters, 127, 255101, doi: 10.1103/PhysRevLett.127.255101

Kivelson, M. G., & Russell, C. T. 1995, Introduction to Space Physics (Cambridge University Press)

Lallement, R., & Bertaux, J. L. 2014, Astronomy and Astrophysics, 565, A41, doi: 10.1051/0004-6361/201323216

Lallement, R., & Bertin, P. 1992, Astronomy and Astrophysics, 266, 479

Lallement, R., Raymond, J. C., Vallerga, J., et al. 2004, Astronomy and Astrophysics, 426, 875, doi: 10.1051/0004-6361:20035929

Leonard, T. W., Möbius, E., Bzowski, M., et al. 2015, Astrophysical Journal, 804, 42, doi: 10.1088/0004-637X/804/1/42

Manura, D. J. S. I. S. I., & Dahl, D. A. I. N. L. 2007, SIMION 8.0 user manual (Ringoes, NJ: Scientific Instrument Services)

McComas, D. J., Angold, N., Elliott, H. A., et al. 2013, Astrophysical Journal, 779, 2, doi: 10.1088/0004-637X/779/1/2

McComas, D. J., Alexashov, D., Bzowski, M., et al. 2012, Science, 336, 1291, doi: 10.1126/science.1221054

McComas, D. J., Bzowski, M., Frisch, P., et al. 2015a, Astrophysical Journal, 801, doi: 10.1088/0004-637X/801/1/28

McComas, D. J., Bzowski, M., Fuselier, S. A., et al. 2015b, Astrophysical Journal, Supplement Series, 220, 22, doi: 10.1088/0067-0049/220/2/22

Möbius, E., Hovestadt, D., Klecker, B., et al. 1985, Nature, 318, 426, doi: 10.1038/318426a0

Möbius, E., Lee, M. A., & Drews, C. 2015a, Astrophysical Journal, 815, 20, doi: 10.1088/0004-637X/815/1/20

Möbius, E., Lee, M. A., Drews, C., & Gloeckler, G. 2016a, in SOLAR WIND 14: Proceedings of the Fourteenth International Solar Wind Conference, Vol. 090002, 090002, doi: 10.1063/1.4943854

Möbius, E., Lee, M. A., Gloeckler, G., Drews, C., & Keilbach, D. 2016b, Journal of Physics: Conference Series, 767, 012017, doi: 10.1088/1742-6596/767/1/012017

Möbius, E., Rucinski, D., Isenberg, P. A., & Lee, M. A. 1996, Annales Geophysicae, 14, 492, doi: 10.1007/s00585-996-0492-x

Möbius, E., Rucinski, D., Lee, M. A., & Isenberg, P. A. 1998, Journal of Geophysical Research: Space Physics, 103, 257, doi: 10.1029/97JA02771

Möbius, E., Litvinenko, Y., Grüwaldt, H., et al. 1999, Geophysical Research Letters, 26, 3181, doi: 10.1029/1999GL003644

Möbius, E., Klecker, B., Bochsler, P., et al. 2010, in American Institute of Physics Conference Series, Vol. 1302, Pickup Ions Throughout the Heliosphere and Beyond, ed. J. Le Roux, G. P. Zank, A. J. Coates, & V. Florinski, 37–43, doi: 10.1063/1.3529987

Möbius, E., Bochsler, P., Bzowski, M., et al. 2012, The Astrophysical Journal Supplement Series, 198, 11, doi: 10.1088/0067-0049/198/2/11

Möbius, E., Bzowski, M., Frisch, P. C., et al. 2015b, Astrophysical Journal, Supplement Series, 220, doi: 10.1088/0067-0049/220/2/24

Moebius, E., Ruciński, D., Hovestadt, D., & k. 1995, Astronomy and Astrophysics, 304, 505. https://ui.adsabs.harvard.edu/abs/1995A&A...304..505M

Moebius, E., Hovestadt, D., Klecker, B., et al. 1985, IEEE Transactions on Geoscience and Remote Sensing, 23, 274, doi: 10.1109/TGRS.1985.289527

Müller, D., St. Cyr, O. C., Zouganelis, I., et al. 2020, Astronomy & Astrophysics, 642, A1, doi: 10.1051/0004-6361/202038467

Nagy, A. F., Brace, L. H., Carignan, G. R., & Kanal, M. 1963, Journal of Geophysical Research, 68, 6401, doi: 10.1029/JZ068i024p06401

NASA. 2016, NASA Helps Power Grids Weather Geomagnetic Storms. https://www.nasa.gov/feature/goddard/2016/nasa-helps-power-grids-weather-geomagnetic-storms

—. 2017, Sun Emits a Solstice Flare and CME. https://solarsystem.nasa.gov/resources/386/sun-emits-a-solstice-flare-and-cme/?category=solar-system_sun

Neugebauer, M., & Snyder, C. W. 1962, Science, 138, 1095, doi: 10.1126/science.138.3545.1095-a

Owen, C. J., Bruno, R., Livi, S., et al. 2020, Astronomy & Astrophysics, 642, A16, doi: 10.1051/0004-6361/201937259

Phillips, J. A. 1955, Physical Review, 97, 404, doi: 10.1103/PhysRev.97.404

Pizzo, V. J. 1985, Washington DC American Geophysical Union Geophysical Monograph Series, 35, 51, doi: 10.1029/GM035p0051

Quinn, P. R., Schwadron, N. A., & Möbius, E. 2016, Astrophysical Journal, 824, 142, doi: 10.3847/0004-637X/824/2/142

Reifman, A., & Dow, W. G. 1949, Physical Review, 76, 987, doi: 10.1103/PhysRev.76.987

Richardson, I. G., & Cane, H. V. 2010, Solar Physics, 264, 189, doi: 10.1007/s11207-010-9568-6

Schwadron, N. A., Moebius, E., Kucharek, H., et al. 2013, Astrophysical Journal, 775, 86, doi: 10.1088/0004-637X/775/2/86

Schwadron, N. A., Möbius, E., Leonard, T., et al. 2015, Astrophysical Journal, Supplement Series, 220, doi: 10.1088/0067-0049/220/2/25

Schwenn, R., & Marsch, E., eds. 1990, Physics of the Inner Heliosphere I (Berlin, Heidelberg: Springer Berlin Heidelberg), doi: 10.1007/978-3-642-75361-9

Sokół, J. M., Bzowski, M., Kubiak, M. A., & Möbius, E. 2016, Monthly Notices of the Royal Astronomical Society, 458, 3691, doi: 10.1093/mnras/stw515

Stakhiv, M. 2016, PhD thesis, University of Michigan, United States

Stone, E. C., Frandsen, A. M., Mewaldt, R. A., et al. 1998, Space Science Reviews, 86, 1, doi: 10.1023/A:1005082526237

Swaczyna, P., McComas, D. J., & Schwadron, N. A. 2019, The Astrophysical Journal, 871, 254, doi: 10.3847/1538-4357/aafa78

Swaczyna, P., Bzowski, M., Kubiak, M. A., et al. 2018, Astrophysical Journal, 854, 119, doi: 10.3847/1538-4357/aaabbf

Swaczyna, P., Kubiak, M. A., Bzowski, M., et al. 2022, arXiv e-prints, arXiv:2201.05463

Taut, A., Berger, L., Möbius, E., et al. 2018, Astronomy and Astrophysics, 611, 1, doi: 10.1051/0004-6361/201731796

Thomas, G. E., & Krassa, R. F. 1971, Astronomy and Astrophysics, 11, 218

University of New South Wales, S. o. P. 2009, Rolling: The Physics of Wheels, https://www.animations.physics.unsw.edu.au//jw/rolling.htm

Vallerga, J., Lallement, R., Lemoine, M., Dalaudier, F., & McMullin, D. 2004, Astronomy and Astrophysics, 426, 855, doi: 10.1051/0004-6361:20035887

Vasyliunas, V. M., & Siscoe, G. L. 1976, Journal of Geophysical Research, 81, 1247, doi: 10.1029/JA081i007p01247

Weller, C. S., & Meier, R. R. 1974, Astrophysical Journal, 193, 471, doi: 10.1086/153182

Witte, M. 2004, Astronomy and Astrophysics, 426, 835, doi: 10.1051/0004-6361:20035956

Wood, B. E., Müller, H.-R., & Witte, M. 2015, Astrophysical Journal, 801, 62, doi: 10.1088/0004-637X/801/1/62

Wüest, M., Evans, D. S., & von Steiger, R. 2007, Calibration of Particle Instruments in Space Physics Calibration of Particle Instruments in Space Physics (International Space Science Institute)

Young, D. T., Berthelier, J. J., Blanc, M., et al. 2004, Space Science Reviews, 114, 1, doi: 10.1007/s11214-004-1406-4

Young, H. D., Freedman, R. A., & Ford, A. L. 2012, University Physics with Modern Physics (Pearson Education). https://books.google.com/books?id=5fgsAAAAQBAJ

Zouganelis, I., De Groof, A., Walsh, A. P., et al. 2020, Astronomy & Astrophysics, 642, A3, doi: 10.1051/0004-6361/202038445