

Examining Plasma Instabilities as Ionospheric Turbulence Generation Mechanisms Using Pseudo-Spectral Methods

Chirag Rathod

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Aerospace Engineering

Bhuvana Srinivasan, Co-Chair
Wayne A. Scales, Co-Chair
Gregory D. Earle
Joseph B. H. Baker

February 19, 2021
Blacksburg, Virginia

Keywords: Plasma instabilities, computational physics, space science, turbulence,
pseudo-spectral methods, ionosphere
Copyright 2021, Chirag Rathod

Examining Plasma Instabilities as Ionospheric Turbulence Generation Mechanisms Using Pseudo-Spectral Methods

Chirag Rathod

(ABSTRACT)

Turbulence in the ionosphere is important to understand because it can negatively affect communication signals. This work examines different scenarios in the ionosphere in which turbulence may develop. The two main causes of turbulence considered in this work are the gradient drift instability (GDI) and the Kelvin-Helmholtz instability (KHI).

The likelihood of the development of the GDI during the August 17, 2017 total solar eclipse is studied numerically. This analysis uses the “Sami3 is Also a Model of the Ionosphere” (SAMI3) model to study the effect of the eclipse on the plasma density. The calculated GDI growth rates are small compared to how quickly the eclipse moves over the Earth. Therefore, the GDI is not expected to occur during the solar eclipse.

A novel 2D electrostatic pseudo-spectral fluid model is developed to study the growth of these two instabilities and the problem of ionospheric turbulence in general. To focus on the ionospheric turbulence, a set of perturbed governing equations are derived. The model accurately captures the GDI growth rate in different limits; it is also benchmarked to the evolution of instability development in different collisional regimes of a plasma cloud.

The newly developed model is used to study if the GDI is the cause of density irregularities observed in subauroral polarization streams (SAPS). Data from Global Positioning System (GPS) scintillations and the Super Dual Auroral Radar Network (SuperDARN) are used to examine the latitudinal density and velocity profiles of SAPS. It is found that the GDI is stabilized by velocity shear and therefore will only generate density irregularities in regions of low velocity shear. Furthermore, the density irregularities cannot extend through regions of large velocity shear. In certain cases, the turbulence cascade power laws match observation and theory.

The transition between the KHI and the GDI is studied by understanding the effect of collisions. In low collisionality regimes, the KHI is the dominant instability. In high collisionality regimes, the GDI is the dominant instability. Using nominal ionospheric parameters, a prediction is provided that suggests that there exists an altitude in the upper F region ionosphere above which the turbulence is dominated by the KHI.

This work was funded by NASA under grant number NASAMAG16_2-0050, the NSF CEDAR program under award number AGS-1552188, the Kevin T. Crofton Department of Aerospace and Ocean Engineering at Virginia Tech, and the Bradley Department of Electrical Engineering at Virginia Tech.

Examining Plasma Instabilities as Ionospheric Turbulence Generation Mechanisms Using Pseudo-Spectral Methods

Chirag Rathod

(GENERAL AUDIENCE ABSTRACT)

In the modern day, all wireless communication signals use electromagnetic waves that propagate through the atmosphere. In the upper atmosphere, there exists a region called the ionosphere, which consists of plasma (a mixture of ions, electrons, and neutral particles). Because ions and electrons are charged particles, they interact with the electromagnetic communication signals. A better understanding of ionospheric turbulence will allow for aid in forecasting space weather as well as improve future communication equipment. Communication signals become distorted as they pass through turbulent regions of the ionosphere, which negatively affects the signal quality at the receiving end. For a tangible example, when Global Positioning System (GPS) signals pass through turbulent regions of the ionosphere, the resulting position estimate becomes worse. This work looks at two specific causes of ionospheric turbulence: the gradient drift instability (GDI) and the Kelvin-Helmholtz instability (KHI). Under the correct background conditions, these instabilities have the ability to generate ionospheric turbulence.

To learn more about the GDI and the KHI, a novel simulation model is developed. The model uses a method of splitting the equations such that the focus is on just the development of the turbulence while considering spatially constant realistic background conditions. The model is shown to accurately represent results from previously studied problems in the ionosphere.

This model is applied to an ionospheric phenomenon known as subauroral polarization streams (SAPS) to study the development of the GDI and the KHI. SAPS are regions of the ionosphere with large westward velocity that changes with latitude. The shape of the latitudinal velocity profile depends on many other factors in the ionosphere such as the geomagnetic conditions. It is found that for certain profiles, the GDI will form in SAPS with some of these examples matching observational data. At higher altitudes, the model predicts that the KHI will form instead.

While the model is applied to just the development of the GDI and the KHI in this work, it is written in a general manner such that other causes of ionospheric turbulence can be easily studied in the future.

To my parents, Darshana and Sanjay, for providing me the opportunities to get to where I am today.

To my partner Erin, for supporting me and being my best friend in life.

Acknowledgements

This dissertation represents the culmination of a 5.5 year journey that has, in many ways, been the most transformative of my life. I would never have been able to get to this point without the support of the amazing people in my life.

I would first like to thank Professor Bhuvana Srinivasan for being a great advisor. I came to VT with no idea of what I wanted to do and no initial advisor. In a last minute decision, I decided to email Prof. Srinivasan since her faculty page intrigued me. A few days later, I was in her office, and she pitched plasma physics to me in such a way that I could not resist. I've learned so much from her throughout the years. She's always been there for me to just pop in her office and bounce ideas off of. When I was confused about a topic, she would always know how to respond by asking the perfect question that would make everything click. I also want to thank her for providing me the opportunity to obtain leadership and mentorship experience by giving me two students to mentor. Lastly, I want to thank her for the moral and emotional support she has given me throughout this journey.

A few years into my PhD, I began working on another project and found myself with a second advisor in Professor Wayne Scales. His expertise on ionospheric physics has been invaluable. He would ask the perfect questions that made me think about my simulation assumptions, which has ensured that I can both accurately model the ionosphere and communicate my work to observationalists. His connections in the field have been incredibly useful; he has given me the ability to present my work to several renowned people in the field of space physics. His humorous anecdotes about silly "controversies" in the field has made research meetings entertaining.

I want to also thank the remaining members of my committee: Professors Gregory Earle and Joseph Baker. My first project was an offshoot of work Prof. Earle was conducting on the 2017 solar eclipse. The conclusion of my project was a null result, but it did produce a paper in which he was a co-author. He also was the professor of the first plasma physics course I took, so I credit him with providing a great foundation in plasma physics. For my second project, Prof. Baker provided useful insights based on his experience with SuperDARN observations. He asked many interesting questions on future ideas that my model could be applied to.

I would like to acknowledge Lee Kordella for his help with my first publication, in which he

was also a co-author. He provided data from a global solar eclipse simulation that I was able to use to study the development of ionospheric turbulence. I also want to acknowledge Dr. Bharat Kunduri for providing me with SuperDARN radar and GPS scintillation data. His contribution to my research ensured that my simulations had a firm realistic grounding. He is a co-author on my second publication (accepted pending minor revisions). I also want to mention Professor Mike Ruohoniemi for sitting in on several meetings and providing useful commentary on my work from the observationalist's perspective.

All of the people above directly contributed to the work in this dissertation. But, it would be irresponsible of me to not mention the privileges that have gotten me to the point of even beginning this journey. I will try to say this in words as best I can, but I don't know if these words are enough to express my gratitude. I want to thank my parents, Sanjay and Darshana Rathod, for everything they have done for me. At a young age, they helped develop my love of science and pushed me to do well in school. My parents immigrated from India with few belongings and they worked incredibly hard. I remember my dad working crazy overtime. I remember my mom commuting long hours and still having time to make a home-cooked meal. They saved as much as they could. The culmination of that hard work was to give me (and my sister) the opportunity of a college education. At the time I left for college, I don't think I truly cherished what they had given me. But, without the huge initial push they gave me, I would never be where I am today. And for that, I will be forever grateful.

I would additionally like to thank my mom for pushing me to attend high school at the Bergen County Academies (BCA). I remember being hesitant to leave my friends and go to a new school, but it was one of the best decisions I've ever made. BCA taught classes at a level that prepared me immensely for college. With all of the higher level classes I took, I graduated with an entire year's worth of college credit.

Because of how my high school prepared me, I felt my undergraduate classes were fairly easy. In my junior year of undergrad, I had the opportunity to take my first graduate level class. It was a class on mechanical vibrations taught by Professor Mac Schwager (now at Stanford University). This was the first class that I felt really delved into the mathematics of engineering and academically challenged me. It was the only graduate level class I took in undergrad, but this brief foray left me wanting more. After graduating, I had trouble finding work. I looked to Prof. Schwager since I enjoyed his class so much. I initially only planned on speaking with him for resumé help. What I did not expect was the next morning getting a phone call from someone he knew that was hiring. That afternoon I was hired. The job was a great entry level position, but it did not scratch the itch that I got from the vibrations class. So I decided to apply to graduate school for a PhD in aerospace engineering. After going through the entire application process, I had a phone interview with Professor Lin Ma (when he still worked at Virginia Tech). It was a good enough interview that I was accepted, but the one thing I remember from it is that he told me I was interviewed specifically because of an outstanding recommendation letter. Now, I obviously don't know who exactly would have written that letter since it is anonymous. But, based on my relationship with everyone else who wrote me a recommendation, it must have been Prof. Schwager. So, I want to profusely

thank Prof. Schwager for giving me these opportunities. Without him, I would not have had my initial job out of college. Without him, I would not have had an understanding of what grad school was like. And without his letter, I would not have written this dissertation.

Many other people at VT have helped me in various ways. I thank Professor Colin Adams for hosting the plasma physics journal club and providing a great platform for learning general plasma physics. I thank the research groups of both of my advisors for teaching me all of the interesting aspects of their work. I specifically thank Dr. Petr Cagas, Lujain Almarhabi, Kolter Bradshaw, John Rodman, Robert Masti, Megan McCracken, and Matthew Carrier for providing useful feedback on this dissertation. I thank Professor Todd Lowe for accepting me for a teaching position that provided a small window into what it's like being a professor. I also want to thank Mark Montgomery for being an excellent reference for that position. I thank Kelsey Wall for promptly answering all of my administrative questions as a grad student. I also thank Cory Thompson for being incredibly helpful in the administrative process at the end here as I transition to my next role.

Many friends have made grad school a great experience. I thank Lindsey Jones and Tim Wood for our bi-weekly Dungeons and Dragons sessions. I thank Thomas Edwards for fun times with board games and pro wrestling. I thank Heather Frye for chill nights watching Seinfeld. I thank Petr and Kristyna Cagas for the great barbeques and the many different versions of Warhammer that we've played. I thank José and Jackie Ortega for allowing me to finish my dissertation in their basement while I am transitioning between apartments. I send a special un-thank you to Masahiro Sakurai, who I believe had it out for me when he released Super Smash Bros Ultimate, which probably made my PhD take longer than it should have.

I thank my cats, Xandi and Miko, for providing much needed snuggle sessions. I especially thank Xandi for being the purry kitty that sat in my lap as I worked all these years. [This comic](#) beautifully illustrates how I've gotten work done. I look at my computer with papers and code and equations; I look down at my lap and "It's a kitty!"; I look back up and get back to work. She provided the intermittent joy that has kept me going.

Finally, being a PhD student is difficult and stressful. I would not have made it to the end of my journey without the love and support of my partner, Erin. On multiple occasions when I felt the world was crumbling around me, she was there to keep me from falling apart. She has proofread for spelling and grammar, listened to my practice presentations, and provided me with a steady stream of coffee for those long nights of manic derivations. This would not have been possible without her at my side, and now that I am done writing, I can't wait to be back by her side.

As the final editor of this work, Erin would like to add a last thank you: to Chirag himself, whose courage and perseverance has taken him to lengths not many of us can achieve. My love, I cannot wait to see where you'll go from here.

Contents

1	Introduction	1
1.1	Plasmas	1
1.2	Ionosphere	3
1.3	Gradient Drift Instability (GDI)	6
1.3.1	Theory and Simulation	8
1.4	Kelvin-Helmholtz Instability (KHI)	11
1.5	Motivation and Objectives	13
2	Total Solar Eclipse	14
3	Model	18
3.1	Mathematical Model	18
3.1.1	Assumptions	18
3.1.2	Velocities	20
3.1.3	Current Closure	24
3.1.4	Continuity Equation	27
3.1.5	Energy Equations	28
3.1.6	Governing Equations	30
3.2	Perturbed Model	30
3.2.1	Governing Equations	33
3.3	Local Linear Theory	33
3.3.1	Assumptions	33

3.3.2	Benchmark Tests	34
3.4	Numerical Methods	37
3.4.1	Pseudo-Spectral or Collocation Method	37
3.4.2	Iterative Method	42
3.4.3	Runge-Kutta Method	44
3.5	Code Verification	45
3.5.1	Order of Accuracy	45
3.5.2	Testing Spatial Derivatives	47
3.5.3	Iterative Solver	50
3.6	Benchmark Problems	51
3.6.1	GDI in Slab Geometry	51
3.6.2	KHI in Slab Geometry	57
3.6.3	Plasma Clouds	62
4	Modeling the Gradient Drift Instability in Subauroral Polarization Streams	69
4.1	Subauroral Polarization Streams (SAPS)	69
4.2	Model Initialization	71
4.3	Results	79
4.4	Summary	91
5	Dominance of the GDI or KHI in Sheared $\mathbf{E} \times \mathbf{B}$ Flows	93
5.1	Motivation	93
5.2	Optimal GDI Growth Direction	93
5.3	Effect of Collisionality	101
5.4	Effect of Velocity Shear Location	106
5.5	Altitude Observation Implications	111
6	Conclusion	115
	Bibliography	118

Appendix A Linear Theory	131
A.1 Geometry	131
A.2 Current Closure	132
A.2.1 Linearized Equation	134
A.3 Continuity	134
A.3.1 Linearized Equation	135
A.4 Matrix Form	135
A.5 Code	137
Appendix B Code Verification Studies	141
B.1 Differentiation	141
B.2 Iterative Solver	147
Appendix C List of Input Files	149
C.1 GDI_tanh2N_noV	149
C.1.1 Initial Conditions	149
C.1.2 Input File	151
C.2 GDI_tanh4N_sech2V	162
C.2.1 Initial Conditions	162
C.2.2 Input File	165
C.3 GDI_KHI_tanh4N_tanhV	177
C.3.1 Initial Conditions	177
C.3.2 Input File	178
C.4 GDI_KHI_tanh2N_tanhV	191
C.4.1 Input File	191
C.5 plasma_cloud	203
C.5.1 Initial Conditions	203
C.5.2 Input File	204
Appendix D Post-Processing Scripts	215

D.1 Best Fit Exponential	215
D.2 Calculating Growth Rate as Function of Wavenumber	220
D.3 Turbulence Energy Cascade	222

List of Figures

1.1	Plots of number densities of neutral particles in the thermosphere (a) and electrons in the ionosphere (b)	5
1.2	A diagram of the optimal geometry for the gradient drift instability (GDI) for the E region (left) and the F region (right)	7
1.3	A diagram of the Kelvin-Helmholtz instability (KHI) growth mechanism	11
2.1	Plots of the electron density (left), GDI growth rate (middle), and eclipse time rate (right)	17
3.1	A plot of the growth rate as a function of the neutral wind in the \hat{x} direction, u_x , and the density gradient scale length, L_N , in the \hat{x} direction	35
3.2	A plot of the growth rate as a function of θ_e and θ_k	36
3.3	A plot of the growth rate as a function of the numerical diffusion constant, D , and the wavenumber	37
3.4	Plots of the norm of the discretization error of the x direction derivatives versus the number of grid points, n	48
3.5	Plots of the norm of the discretization error of the y direction derivatives versus the number of grid points, n	49
3.6	Plots of the norm of the discretization error of the Laplacians versus the number of grid points, n	50
3.7	Plots of the norm of the discretization error of $\partial\phi_p/\partial t$	51
3.8	A diagram of the slab model of the GDI with labeled axes and important vectors	52
3.9	Plots of the plasma density showing the GDI evolution at different times using the parameters from Table 3.1	54
3.10	Plots of the plasma density showing the inertial GDI evolution at different times using the parameters from Table 3.1	55

3.11	The growth rate, γ , vs the wavenumber, k , for a set of single mode GDI simulations whose parameters are found in Table 3.2	57
3.12	A diagram of the slab model of the KHI with labeled axes and important vectors	58
3.13	Plots of the plasma density to show the KHI evolution at different times using the parameters from Table 3.3 with weak collisions	60
3.14	Plots of the plasma density to show the KHI evolution at different times using the parameters from Table 3.3 for stronger collisions	61
3.15	A diagram of a plasma cloud in a background plasma with labeled axes and important vectors	62
3.16	Density plots showing the evolution of a plasma enhancement cloud in a collisional regime	64
3.17	Density plots showing the evolution of a plasma enhancement cloud in an inertial regime	65
3.18	Density plots showing the evolution of a plasma depletion cloud in a collisional regime	66
3.19	Density plots showing the evolution of a plasma depletion cloud in an inertial regime	67
4.1	A diagram of the coordinate system used, where \hat{x} is north, \hat{y} is east, and \hat{z} is down	71
4.2	Latitudinal profiles at 20 MLT of TEC (a) and SAPS velocity (b) of a May 2, 2013 SAPS event	73
4.3	Histograms of different latitudinal TEC and velocity profile parameters for 10 different SAPS events across multiple MLT cuts	76
4.4	Plots of background density and velocity profiles. The functions vary in x from 0 to 750 km	77
4.5	Results of a shearless GDI simulation	80
4.6	Plots of plasma density using Eq. 4.10 for the V_y initial conditions with $V_0^s = -500$ m/s, $L_V^s = 34$ km, and d varying from 75 to 175 km	81
4.7	A plot of power spectra of the perturbed density divided by the total density and perturbed electric potential at $\tilde{t} = 6.45$ from a higher resolution simulation of Figure 4.6(b)	83
4.8	Plots of plasma density later in time for Figures 4.6(a) and (b)	84

4.9	The bottom panels show plots of plasma density for a case using Eq. 4.11 for the background, V_y ; the top panels show the y -averaged density normalized by n_0	85
4.10	The bottom panels show plots of plasma density using Eq. 4.11 for the V_y initial conditions with $V_0^t = -1000$ m/s, $L_V^t = 10$ km, and w varying from 125 to 305 km; the top panels show the y averaged density normalized by n_0	87
4.11	A plot of power spectra of the perturbed density divided by the total density and the perturbed electric potential at late time for all three cases from Figure 4.10 in the turbulent region away from the velocity shear	88
4.12	The bottom panels show plots of the plasma density for the same simulations as in Figures 4.10(b-d), but with an additional zonal component of neutral wind of -300 m/s; the top panels show the y -averaged normalized density profiles	89
4.13	A plot of power spectra of the perturbed density divided by the total density and the perturbed electric potential at late time for all three cases from Figure 4.12 in the turbulent region away from the velocity shear	90
5.1	A diagram of unit vector definitions for $\hat{\mathbf{k}}$, $\hat{\mathbf{e}}$, and $\hat{\mathbf{g}}$ in the xy plane	94
5.2	A plot of f as a function of θ_k and θ_e for $\theta_g = 0$ and $R = 0, 1, \text{ and } 10^6$	95
5.3	Plots of f with $\theta_g = 0$ and $R = 10^{-6}$ for different \mathbf{u} and $\mathbf{V}_{\mathbf{E} \times \mathbf{B}}$ and therefore different θ_e	98
5.4	Plots of plasma density for different θ_e from 0 to $7\pi/4$	100
5.5	Density color plots showing instability development for different collisionalities increasing to the right	104
5.6	Density color plots showing instability development for different collisionalities increasing to the right for a higher velocity case	105
5.7	Density color plots with different velocity shear locations using $R = 4.62 \times 10^{-6}$	108
5.8	Density color plots with different velocity shear locations using $R = 4.92 \times 10^{-6}$ and $u_x = 500$ m/s	109
5.9	Density color plots using the equivalent neutral wind driven electric field to Figures 5.7(b) and 5.8(b)	110
5.10	An example plot of R as a function of altitude	112
5.11	The x direction normalized density (top) and perturbed potential (bottom) power spectra for the cases in Figures 5.5 and 5.6	113

5.12	The y direction normalized density (top) and perturbed potential (bottom) power spectra for the cases in Figures 5.5 and 5.6	114
B.1	Plots of F_1 to F_7 (Eqs. B.1-B.7)	145
B.2	Plots of exact x derivatives of F_1 to F_7 (Eqs. B.8-B.14)	145
B.3	Plots of exact y derivatives of F_1 to F_7 (Eqs. B.15-B.21)	146
B.4	Plots of exact Laplacians of F_1 to F_7 (Eqs. B.22-B.28)	146
B.5	The left panel shows the exact solution for $\partial\phi_p/\partial t$ and the right panel shows the method of manufactured solutions source term	148
C.1	An example of the background density profile as a function of x	150
C.2	An example of the background profiles as functions of x	164
C.3	An example of the background density, velocity, and electric potential profiles as functions of x	178
C.4	An example of the density initial condition using Eq. C.15	204
D.1	Domain-integrated electric field energy vs time for the single mode slab GDI simulation described by Table 3.2 with a y domain length of 7.8125 km . . .	216
D.2	A plot of the domain-integrated electric field energy vs time showing the raw data (blue dots) and the best fit exponential (red line) for the same case as in Figure D.1	218

List of Tables

3.1	A table of the input parameters for slab GDI simulations using Listing C.1 as the input file	53
3.2	A table of the input parameters for several collisional single mode slab GDI simulations using Listing C.1 as the input file	56
3.3	A table of the input parameters for a slab KHI simulation using Listing C.4 as the input file	59
3.4	A table of the input parameters for a plasma cloud simulation using Listing C.5 as the input file	63
4.1	Nominal simulation parameters for modeling SAPS	78
4.2	A table of the input parameters for a control GDI simulation using Listing C.2 as the input file	79
4.3	A table of the input parameters for SAPS profiles using Listing C.2 as the input file and Eq. 4.10 for the background velocity	82
4.4	A table of the input parameters for SAPS profiles using Listing C.3 as the input file and Eq. 4.11 for the background velocity	86
5.1	A table of the input parameters testing expected growth direction using Listing C.1 as the input file	101
5.2	A table of the input parameters examining instability growth for different collisionalities and background velocities using Listing C.4 as the input file	102
5.3	A table of the input parameters testing if the GDI may develop in the low collisional KHI regime using Listing C.4 as the input file	107
B.1	A table of the coefficients used in Eqs. B.1-B.7	144
B.2	A table of the coefficients for different manufactured solutions of the form of Eq. B.29.	147

B.3 A table of the auxiliary parameters for testing the iterative solver. 147

Listings

3.1	Definition of the wavenumber vector in MATLAB.	39
A.1	Mathematica notebook that analytically solves for ω using Eqs. A.35 and A.37.	137
A.2	Matlab function that calculates the wave frequency and growth rate based on the theory derived in Appendix A. The functional forms are found using Listing A.1.	138
C.1	Input file for a GDI simulation that approximates the density with 2 hyperbolic tangent functions and has no background velocity.	151
C.2	Input file for a GDI simulation that approximates the density with 4 hyperbolic tangent functions and approximates the velocity with 2 hyperbolic secant squared functions.	165
C.3	Input file for a GDI and KHI combination simulation that approximates the density with 4 hyperbolic tangent functions and approximates the velocity with 4 hyperbolic tangent functions.	178
C.4	Input file for a GDI and KHI combination simulation that approximates the density with 2 hyperbolic tangent functions and approximates the velocity with 4 hyperbolic tangent functions.	191
C.5	Input file for a plasma cloud simulation that approximates the density with an elliptical Gaussian type function with no background velocity.	204
D.1	Function that finds the best fit exponential within a set of datapoints.	219
D.2	Function that takes the FFT in one direction and integrates in the other.	221
D.3	Function that takes finds a power law fit for density and electric potential data. Note that this makes use of the FFTMat function from Listing D.2.	222

Chapter 1

Introduction

The objective of this dissertation is to develop and apply a model to study ionospheric turbulence. Chapter 1 is an introduction to plasma physics and the ionosphere is provided. The gradient drift and Kelvin-Helmholtz instabilities are discussed in great detail to understand how they generate turbulence in the ionosphere. Chapter 2 discusses a brief project that examined the possibility of ionospheric turbulence during the August 17, 2017 solar eclipse. Chapter 3 discusses the derivation, computational framework, and verification of the developed model. In Chapter 4, the model is applied to better understand the growth of the gradient drift instability within the ionospheric phenomenon known as subauroral polarization streams. In Chapter 5, the physics of the transition between the gradient drift and Kelvin-Helmholtz instabilities, primarily based on collisionality, is explored using this model. Chapter 6 provides concluding remarks as well as directions for future work. Auxiliary derivations, notes, input files, and post-processing scripts are provided in Appendices A through D.

1.1 Plasmas

Plasma is colloquially considered the fourth state of matter. When a gas enters an excited state, either through increasing temperature, ionizing radiation, or a large electric field, some electrons may be pulled away from an atom or a molecule resulting in a mixture of ions, electrons, and neutral particles; this mixture is called a plasma. Because ions and electrons are charged particles, plasmas, unlike neutral gases, interact with electric and magnetic fields through the Lorentz force. There are many practical applications of plasma. A few examples of uses of plasmas are: to etch circuits, as propellants for electric propulsion devices such as Hall thrusters, and as fuel in nuclear fusion devices such as tokamaks. Additionally, plasmas are the most common form of normal matter (i.e., not dark matter) in the universe and exist in stars, the intergalactic medium, and planetary magnetospheres and ionospheres.

The most basic way of modeling a plasma is to understand the Lorentz force acting on each particle and to track each individual particle's motion over time. This is called the Particle-in-Cell (PIC) method [Birdsall and Langdon, 2005]. While the PIC method is useful in many applications, it does not scale well with large numbers of particles. Instead of treating the particles individually, they can be described as a distribution of particles and modeled using the Vlasov equation:

$$\frac{\partial f_s}{\partial t} + \mathbf{V} \cdot \nabla_x f + \frac{q_s}{m_s} (\mathbf{E} + \mathbf{V} \times \mathbf{B}) \cdot \nabla_{\mathbf{v}} f_s = 0, \quad (1.1)$$

where f is the distribution function, q is the charge, m is the mass, \mathbf{E} is the electric field, \mathbf{B} is the magnetic field, \mathbf{V} is the velocity, and the subscript s denotes the species. Note how the velocity is another coordinate, like space, making the plasma description 7-dimensional (3 space, 3 velocity, 1 time) [Chen et al., 1984, Chap. 7]. The Vlasov, or continuum kinetic, approach does not scale well due to the increased dimensionality arising from needing both spatial and velocity coordinates. Therefore, some simplifying assumptions can be made about the plasma. The results will not include all of the physics, but can still provide useful approximations.

First, consider a point test charge embedded in a plasma. The plasma will react by rearranging itself so as to cancel the effect of the charge. Therefore, the test charge does not affect any plasma that is far away. The characteristic length of this phenomenon, called the Debye length, is

$$\lambda_D = \sqrt{\frac{\epsilon_0 k_B T_e}{n_e e^2}}, \quad (1.2)$$

where ϵ_0 is the vacuum permittivity, k_B is the Boltzmann constant, T_e is the electron temperature, n_e is the electron number density, and e is the elementary electric charge. When exploring plasmas at a length scale greater than the Debye length, the plasma can be thought of as quasineutral, i.e., on a large enough scale, plasmas can be considered neutral such that $n_i = n_e = n$.

If the plasma is Maxwellian, i.e., in thermal equilibrium, then it has a Maxwellian distribution function of the form

$$f(\mathbf{V}) = n \pi^{-3/2} V_{Th}^{-3} \exp\left(-\frac{\mathbf{V} \cdot \mathbf{V}}{V_{Th}^2}\right), \quad (1.3)$$

where n is a background density, \mathbf{V} is the velocity, and V_{Th} is the thermal velocity, which is defined as

$$V_{Th} = \sqrt{\frac{2k_B T}{m}}. \quad (1.4)$$

The zeroth, first, and second moments of Eq. 1.3 (when multiplied by the species' mass) yield the important macroscopic quantities of density, momentum, and energy¹, respectively.

¹The choice of closure of the equations will determine whether the energy is in the form of a scalar or a tensor.

Therefore, if the plasma is assumed to be Maxwellian, it can be described by its bulk and macroscopic properties. Through this lens, the plasma can be modeled as a fluid. Other further simplifying assumptions can be made, leading to different fluid models such as the two-fluid model [Shumlak and Loverich, 2003], the magnetohydrodynamic model [Chen et al., 1984, Sec. 5.7], and the ionospheric plasma model described in Chapter 3.

Plasmas exist in a broad range of regimes that can have large impacts on their behavior. A useful dimensionless parameter to provide a basic understanding of the different dynamics is

$$\beta = \frac{P}{P_{mag}} = \frac{nk_B T}{B^2/2\mu_0} = \frac{v_{Th}^2}{V_{Alf}^2}, \quad (1.5)$$

where n is the plasma density, k_B is the Boltzmann constant, T is the plasma temperature, B is the magnetic field magnitude, and μ_0 is the vacuum permeability. Eq. 1.5 is a ratio of the thermal pressure (P) to the magnetic pressure (P_{mag}). An equivalent way of viewing β is as the ratio of the thermal velocity to the Alfvén velocity. Alfvén waves are hydromagnetic waves that travel parallel to the magnetic field and are caused by plasma oscillations with magnetic tension as the restoring force. The Alfvén velocity is

$$V_{Alf} = \frac{B}{\sqrt{\mu_0 n m_i}}. \quad (1.6)$$

For $\beta \gg 1$, the plasma has enough energy to dominate the dynamics and can move the magnetic field lines. For $\beta \ll 1$, the magnetic field has much more energy than the plasma and heavily constrains the motion of the plasma. An example of a high β plasma is the solar wind, which drags the interplanetary, or heliospheric, magnetic field from the corona outward to the solar system. In contrast, the Earth’s ionosphere is a low β plasma whose motion is heavily affected by the geomagnetic field.

1.2 Ionosphere

In the altitude range of about 50 km to 1000 km above sea level is a region of the atmosphere in which there are large amounts of charged particles (ions and electrons). This region describing the plasma is called the ionosphere. The overlapping region, when discussing the neutral particles, is called the thermosphere. Photons from solar radiation impact the atoms and molecules in the upper atmosphere, energizing the neutral particles enough to ionize them. This process is called photoionization. Solar radiation in the extreme ultraviolet (EUV) spectrum is the primary photoionization source.

Figure 1.1 shows an example of the thermospheric neutral density (a) and the ionospheric plasma density (b). The neutral data are obtained from the NRLMSISE-00 model [Picone et al., 2002], and the plasma data are obtained from the IRI model [Bilitza, 2018]. The parameters used are for the date of July 1, 2000 at a geographic latitude and longitude

of 0° and 40° respectively. Daytime values are obtained at 13:00 local time and nighttime values are obtained at 01:00 local time. Even in the altitude ranges of the ionosphere, the neutral density is significantly greater than the plasma density. Because photoionization is the primary source of ionization, the plasma density differs between daytime and nighttime. This phenomenon is depicted in Figure 1.1(b), with the blue line representing the daytime density and the red line representing the nighttime density. Additionally, the ionosphere has 3 distinct regions: the D , E , and F regions [Kelley, 2009], which are labeled in Figure 1.1(b). The D region is the lowest in altitude. This region includes many different ion species (positive and negative) and is dominated by chemical reactions between the many ion and neutral species that alter the composition of the plasma. Because the neutral density is large at this altitude, the D region is also dominated by interactions with neutral particles. All of these different species and interactions cause the D region to be a physically complicated system. At nighttime, the recombination chemistry losses in conjunction with the decreased photoionization causes the D region to disappear. Chemical reactions and interactions with neutral particles also dominate the physics of the E region; however, there are fewer ion species and less complicated chemical reactions, which simplifies the physics. The F region includes the area of highest plasma density and is sometimes split into two peaks, called the F_1 and the F_2 peaks. In this region, collisions with neutral particles and, for higher altitudes, inertial effects are important. The work in this dissertation focuses on the F region. Parameters such as number density, temperature, composition, and velocity can be measured using a wide array of instruments including sounding rockets, satellite measurements, incoherent scatter radar (ISR), and coherent scatter radar, e.g., the Super Dual Auroral Radar Network (SuperDARN) [Chisham et al., 2007].

A simplified understanding of the electrodynamics of the ionosphere arises from considering solely the effects of collisions with neutral particles and the electric field on the plasma dynamics. The result is related to the current through Ohm's law as

$$\mathbf{J} = \sigma \cdot \mathbf{E}, \quad (1.7)$$

where \mathbf{J} is the current, \mathbf{E} is the electric field, and σ is the conductivity tensor. If the coordinate system is defined such that the magnetic field is in the $\hat{\mathbf{z}}$ direction, then the conductivity tensor, σ , is

$$\sigma = \begin{bmatrix} \sigma_P & -\sigma_H & 0 \\ \sigma_H & \sigma_P & 0 \\ 0 & 0 & \sigma_{\parallel} \end{bmatrix}, \quad (1.8)$$

where σ_P is the Pedersen conductivity, σ_H is the Hall conductivity, and σ_{\parallel} is the parallel

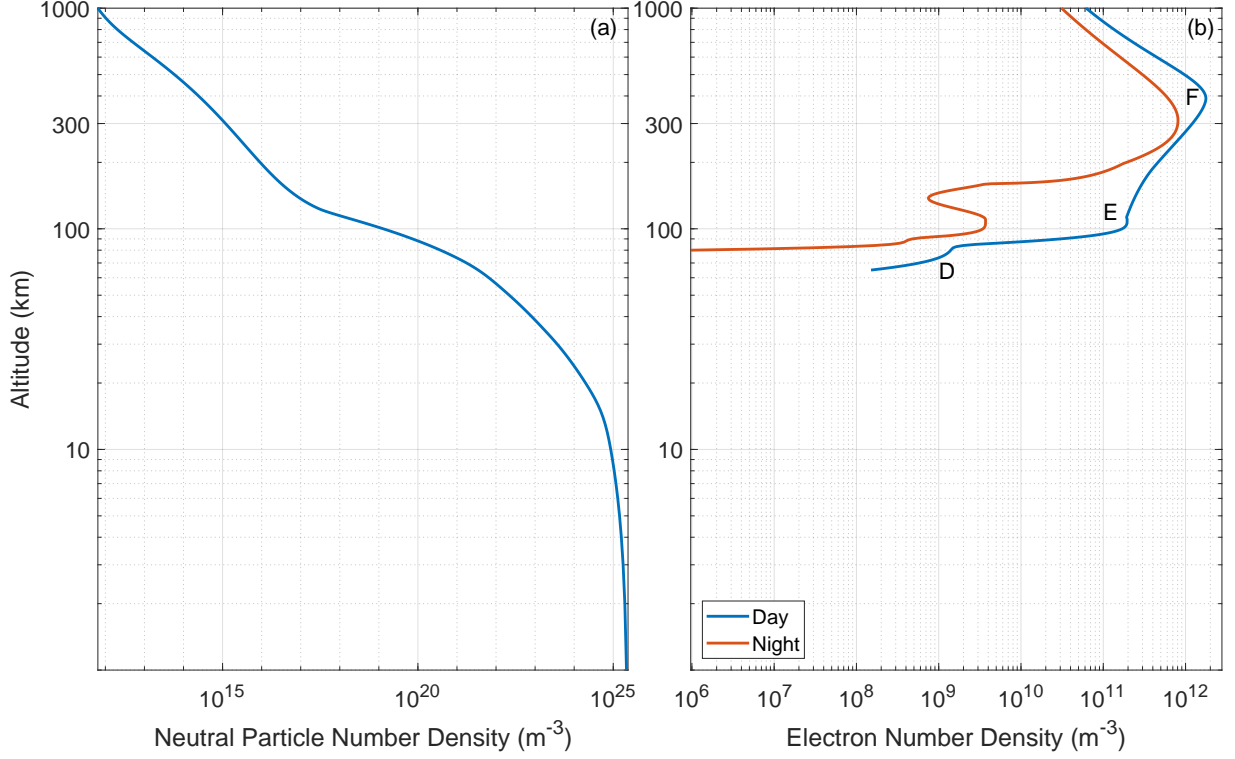


Figure 1.1: Plots of number densities of neutral particles in the thermosphere (a) and electrons in the ionosphere (b). Note how there are significantly more neutral particles than charged particles even in the ionosphere. Panel (b) shows the difference in the plasma density between daytime (blue) and nighttime (red). The *D*, *E*, and *F* regions of the ionosphere [Kelley, 2009] are labeled in Panel (b). Neutral data are obtained from the NRLMSISE-00 model [Picone et al., 2002] and the plasma data are obtained from the IRI model [Bilitza, 2018].

conductivity. Each of these conductivities are

$$\sigma_P = ne^2 \left(\frac{1}{m_i} \frac{\nu_{in}}{\nu_{in}^2 + \Omega_{ci}^2} + \frac{1}{m_e} \frac{\nu_{en}}{\nu_{en}^2 + \Omega_{ce}^2} \right) \quad (1.9)$$

$$\sigma_H = \frac{ne}{B} \left(\frac{\Omega_{ce}^2}{\nu_{en}^2 + \Omega_{ce}^2} - \frac{\Omega_{ci}^2}{\nu_{in}^2 + \Omega_{ci}^2} \right) \quad (1.10)$$

$$\sigma_{\parallel} = ne^2 \left(\frac{1}{m_i \nu_{in}} + \frac{1}{m_e \nu_{en}} \right), \quad (1.11)$$

where n is the electron number density, e is the elementary electric charge, m is the mass, B is the magnetic field magnitude, $\Omega_{c\alpha} = q_{\alpha}B/m_{\alpha}$ is the gyrofrequency, and $\nu_{\alpha n}$ is the collision frequency for a species α (ions or electrons). Note that these conductivities assume only one singly ionized ion species. The Hall conductivity plays a more important role at lower altitudes (below about 130 km). At higher altitudes, the Pedersen conductivity is more

important. The parallel conductivity is the strongest of the three throughout the ionosphere and increases with altitude [Kelley, 2009].

All modern wireless communication systems work by broadcasting electromagnetic signals, which are typically in the radio or microwave frequency range. Because the ionosphere contains charged particles, the electromagnetic waves interact with the plasma as they propagate. Based on the characteristics of the plasma, the waves will propagate differently. The plasma density strongly affects the angle of electromagnetic wave propagation in the ionosphere. When the ionosphere is undisturbed, i.e., not turbulent, the communication signals reach their intended destination with no issue. However, for a disturbed, i.e., turbulent, ionosphere, the signal becomes disrupted and is thus weaker at the destination. This is increasingly important for the Global Navigation Satellite System (GNSS), where the resolution of the position decreases if the signal travels through turbulent regions in the ionosphere [Kintner and Ledvina, 2005]. The resulting scintillations of the GNSS signals can also be used as diagnostics on the ionosphere.

Ionospheric turbulence is often caused by various plasma instabilities [Fejer and Kelley, 1980, Keskinen and Ossakow, 1983b, Tsunoda, 1988]. Based on the background parameters, different instabilities are known to generate density irregularities. For example, E region structuring is dominated by the Farley-Buneman instability (FBI) [Farley Jr, 1963, Buneman, 1963]. This dissertation focuses primarily on the gradient drift instability (Section 1.3) and secondarily on the Kelvin-Helmholtz instability (Section 1.4). Both of these instabilities generate turbulence in the F region. The goal of this dissertation is to develop a numerical model that is then used to understand how these instabilities grow under different ionospheric conditions. A better understanding of ionospheric turbulence can lead to predictive forecasting models of space weather, which can be used to improve communication signals.

1.3 Gradient Drift Instability (GDI)

The gradient drift instability (GDI) is a plasma interchange instability similar to the well-known hydrodynamic instability, the Rayleigh-Taylor instability [Chandrasekhar, 1961, Chap. 10]. In the literature, the GDI is sometimes called the $\mathbf{E} \times \mathbf{B}$ instability or the cross-field instability. In an inhomogeneous, weakly ionized plasma, when there is a difference in the effect of ion-neutral and electron-neutral collisions and there exists a background electric field or current, a space charge can develop. To provide an intuitive understanding of this, the frictional effects from the collisions with neutral particles act on the ions and the electrons differently, resulting in slightly different speeds for each species; this leads to a net current or a space charge. Figure 1.2 shows a diagram of the GDI growth mechanism. The space charge results in a perturbed electric field, which interacts with the background magnetic field to cause an unstable and growing perturbed $\mathbf{E} \times \mathbf{B}$ drift. Figure 1.2 shows how the optimal direction of the background electric field is different depending on the region of the ionosphere. Because collisions are much stronger in the E region from the increased neutral

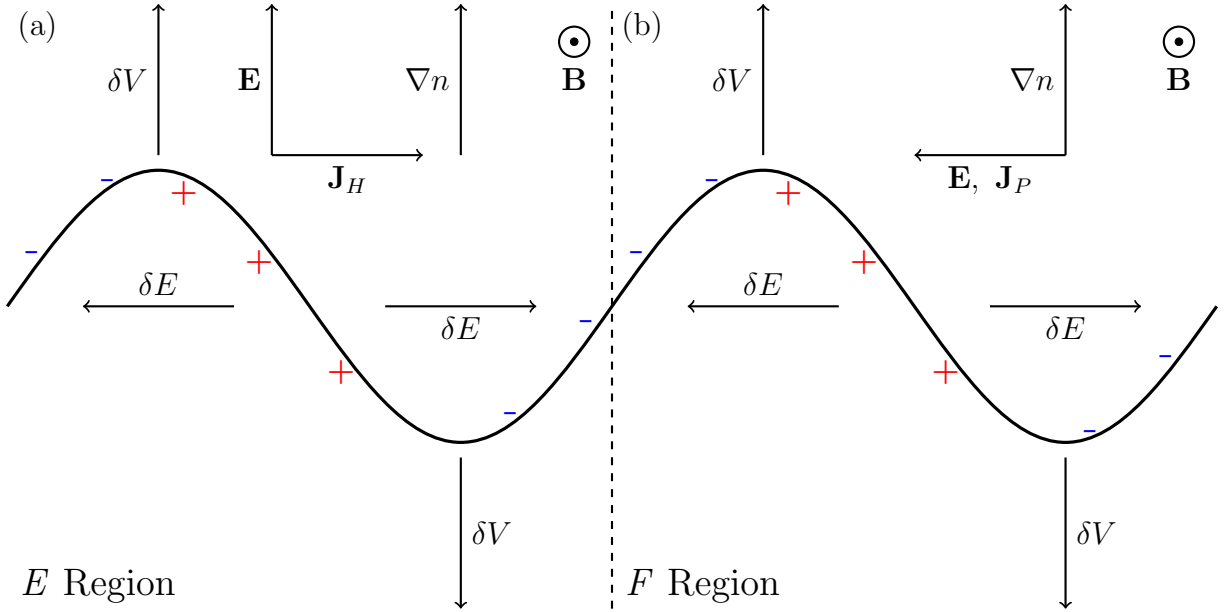


Figure 1.2: A diagram of the optimal geometry for the gradient drift instability (GDI) for the E region (left) and the F region (right). In both cases, the magnetic field, \mathbf{B} , is out of the page, and the density gradient, ∇n , is up with a perturbed interface. In the E and F regions, the charge separation is driven by the Hall and Pedersen currents (\mathbf{J}_H and \mathbf{J}_P), respectively.

density, as shown in Figure 1.1(a), the Hall current, \mathbf{J}_H , is the dominating mechanism that causes the space charge. Thus, the optimal electric field direction for the GDI to grow in the E region is such that $\mathbf{E} \parallel \nabla n$ [Sahr and Fejer, 1996], as shown in Figure 1.2(a). The neutral density is significantly lower in the F region; therefore, the Pedersen current, \mathbf{J}_P , dominates. Thus, the optimal electric field direction for the GDI to grow in the F region is such that $\mathbf{E} \times \mathbf{B} \parallel \nabla n$ [Linson and Workman, 1970], as shown in Figure 1.2(b).

The first observations of the GDI were made in laboratory discharge experiments in a weakly ionized plasma [Simon, 1963, Hoh, 1963]. The instability was then hypothesized to occur in the naturally weakly ionized ionosphere. Initial work attempted to explain density irregularities observed in sporadic E layers [Sato et al., 1968] and the E region in general [Chimonas, 1969]. The instability became studied more frequently to explain structures that develop in barium cloud release experiments [Linson and Workman, 1970, Haerendel and Lüst, 1968, McDonald et al., 1980]. See Section 3.6.3 for more information on the development of GDI-caused structures and striations in plasma clouds. Due to have a geometry similar to that of a plasma cloud, the GDI is thought to cause plasma structuring in the trailing edge of polar cap patches [Weber et al., 1984, Moen et al., 2012, Spicher et al., 2015, Lamarche and Makarevich, 2017]. The GDI, along with the current convective instability, is considered a density irregularity generation mechanism in diffuse auroras [Greenwald, 1974, Tsunoda and

Vickrey, 1982, Keskinen and Ossakow, 1983a, Sahr and Fejer, 1996]. This combination of instabilities also causes the dusk scattering effect seen in the mid-latitude ionospheric trough [Ruohoniemi et al., 1988]. The GDI can work in conjunction with the Rayleigh-Taylor instability to form a combination instability named the Generalized Rayleigh-Taylor instability [Kelley et al., 1981]. This combination instability causes the ionospheric phenomenon known as equatorial spread-F [Kelley et al., 1979, 1981, Hysell et al., 2004]. The turbulence observed in subauroral polarization streams (SAPS) is hypothesized to be caused by the GDI [Mishin et al., 2003, Keskinen et al., 2004, Mishin and Blaunstein, 2008]. This hypothesis is studied in greater detail in Chapter 4. Striations have been observed in the recently discovered optical phenomenon known as strong thermal emission velocity enhancements (STEVE) [MacDonald et al., 2018]; the GDI has been proposed as a possible generation mechanism [Semeter et al., 2020]. Review papers by Fejer and Kelley [1980], Keskinen and Ossakow [1983b], and Tsunoda [1988] are excellent resources for further information on plasma instabilities in the ionosphere in general.

1.3.1 Theory and Simulation

The linear theory has been researched extensively to gain a better understanding of the GDI growth rate under different conditions. Many of the works cited in this section also include simulation results that agree with their theoretical counterparts.

Consider the geometry shown in Figure 1.2(b) for the F region. Linson and Workman [1970] show that the simplified growth rate in this geometry is

$$\gamma_F = \frac{|\nabla n| E}{n B} = \frac{E}{BL} = \gamma_0, \quad (1.12)$$

where E is the electric field magnitude, B is the magnetic field magnitude, n is the plasma number density, and L is the density gradient scale length defined as $L = n|\nabla n|^{-1}$. The growth rate from Eq. 1.12 becomes defined as γ_0 as it is the base of most of the other formulations of the GDI growth rate. Interestingly, this version (and several others) of the GDI growth rate does not depend on the wavenumber. In the lower altitude E region, the GDI growth rate, in the geometry of Figure 1.2(a), is

$$\gamma_E = \frac{1}{(1 + \psi)^2} \frac{\nu_{in} V_{\mathbf{E} \times \mathbf{B}}}{\Omega_{ci} L} = \frac{\nu_{in}}{\Omega_{ci}} \frac{\gamma_0}{(1 + \psi)^2}, \quad (1.13)$$

where $\nu_{\alpha n}$ is the collision frequency between species α and neutrals, $\Omega_{c\alpha}$ is the cyclotron frequency of species α , and $\psi = \nu_{in} \nu_{en} (\Omega_{ci} \Omega_{ce})^{-1}$ [Rogister and d'Angelo, 1970, Sudan et al., 1973]. The value ψ is usually much less than 1 and often negligible. Eqs. 1.12 and 1.13 both neglect the effects of inertia. The E region is heavily dominated by collisions and the lower F region has enough collisions such that inertial effects are negligible. However, higher in the F region, there are significantly less collisions, meaning the effects of inertia become more

important. Ossakow et al. [1978] consider the high-altitude F region limit of the GDI with some simplifying assumptions. Under the condition that $\nu_{in} \ll 4\gamma_0$, the GDI growth rate is

$$\gamma^{\text{high alt}} = \sqrt{\nu_{in}\gamma_0}. \quad (1.14)$$

When the condition $\nu_{in} \ll 4\gamma_0$ is not met (but still in the F region), the growth rate is that of Eq. 1.12. Despite the differences between Eqs. 1.12 to 1.14, the general intuition of the GDI mechanism holds. At low altitudes where $\nu_{in}/\Omega_{ci} > 1$, the increased effect of collision plays an enhancing role in the growth rate as seen in Eq. 1.13. In the lower F region where $\nu_{in}/\Omega_{ci} < 1$, the collisional term does not appear in Eq. 1.12. This viewed as a middle area where the collisions are high enough to cause the GDI but not to substantially increase the growth rate. At extremely high altitudes where $\nu_{in}/\Omega_{ci} \ll 1$, the GDI still grows but at a much slower rate. This is apparent in Eq. 1.14, where the base growth rate γ_0 is taken to the power of 1/2 and multiplied by the now small $\sqrt{\nu_{in}}$.

Several works in the literature have extended the GDI from the simplified geometry in Figure 1.2 to more general geometries for both the E region [Rogister and d'Angelo, 1970, Sahr and Fejer, 1996] and the F region [Keskinen and Ossakow, 1982, 1983a]. Neglecting inertia and parallel transport, Makarevich [2014] develops an altitude independent formulation of the GDI growth rate for an arbitrary geometry. The resulting equation is

$$\gamma = \frac{\gamma_0}{1 + \psi} \frac{\hat{\mathbf{k}} \cdot \hat{\mathbf{b}} \times \hat{\mathbf{g}} \left(\hat{\mathbf{k}} \cdot \hat{\mathbf{e}} - R \hat{\mathbf{k}} \cdot \hat{\mathbf{e}} \times \hat{\mathbf{b}} \right)}{1 + \eta^2 \left(\hat{\mathbf{g}} \cdot \hat{\mathbf{k}} - R \hat{\mathbf{g}} \cdot \hat{\mathbf{k}} \times \hat{\mathbf{b}} \right)^2}, \quad (1.15)$$

where $\psi = \nu_{in}\nu_{en}(\Omega_{ci}\Omega_{ce})^{-1}$, $\eta = (kL)^{-1}$, $R = (\nu_{in}/\Omega_{ci} - \nu_{en}/\Omega_{ce})(1 + \psi)^{-1}$ and $\hat{\mathbf{k}}$, $\hat{\mathbf{b}}$, $\hat{\mathbf{g}}$, and $\hat{\mathbf{e}}$ are the unit vectors for the wavenumber, magnetic field, density gradient, and electric field, respectively. When taken to the limits of the E or F region, Eq. 1.15 becomes Eqs. 1.13 or 1.12, respectively. Note that Eq. 1.15 is pivotal to the work in this dissertation and acts as a foundation for understanding several different aspects of GDI growth. Additional works consider the inclusion of the inertial effects, thermal effects, and parallel transport in an altitude independent dispersion analysis that considers the combined effects of the GDI, FBI, and current convective instability (CCI) [Makarevich, 2016a,b]. The dispersion relation from those papers is difficult to solve analytically without taking some sort of limit that reduces the growth rate to the simpler ones discussed above. Makarevich [2019a] makes some simplifying assumptions to obtain an analytical, albeit unwieldy, solution to the dispersion relation. Assuming purely field-aligned irregularities and a slow growth rate ($|\gamma| \ll \nu_{in}$), the analytical growth rate is found in Eq. 32 from Makarevich [2019a]. In the regime of the F peak considered for much of this dissertation, that equation is effectively the same as Eq. 1.15; the altitudes are too low for inertia to play a role in the linear GDI growth. Makarevich [2019b] uses Eq. 32 from Makarevich [2019a] to further explore the GDI growth rate in the transitional region between the E and F regions. Most recently, a generalized dispersion relation is found for the combination of the GDI, FBI, and the ion-thermal instability (ITI)

to further integrate the different ionospheric instabilities that contribute to the generation of density irregularities [Makarevich, 2020].

All of the growth rates presented thus far have assumed a local approximation, or $1/k \ll L$. The utility of this approximation is that it allows for simplified Fourier differentiation when linearizing the governing equations. However, there are situations in which this approximation is invalid and the results do not accurately capture the physics. The initial work for the nonlocal linear analysis examined barium release clouds in a simplified geometry to better understand their deformation [Perkins et al., 1973]. For many of these types of analyses as well as simulations, the initial conditions should begin in an equilibrium state unperturbed. Just by nature of the equation system, if a density gradient is initialized, at least one of the components of the electric field must be non-constant in space. Since all of the plasma motions are assumed to be caused by the $\mathbf{E} \times \mathbf{B}$ drift, a non-uniform electric field means that there is the presence of a velocity shear. In the local approximations, the velocity shear physics are not considered. Perkins and Doles III [1975] consider the physics of velocity shear in a barium release cloud with a simplified slab geometry. The velocity shear has a stabilizing effect with the condition that the interface is stable to the GDI if

$$\frac{E_x(x_0)}{E_{0y}} > \frac{2}{kL}, \quad (1.16)$$

where $E_x(x)$ is the non-uniform component of the electric field, x_0 is the location of minimum density gradient scale length, and E_{0y} is the constant component of the electric field. This condition shows that the effect of the shear stabilization is stronger on short wavelength modes. The work by Perkins et al. [1973] and Perkins and Doles III [1975] provide useful intuition but are only applicable quantitatively for their simple geometry. Huba et al. [1983] consider the same problem but make no assumptions on the density and electric field profiles, include inertial effects, and make no assumptions on the perturbation wavelength. This results in a complicated differential equation that cannot be solved analytically; no closed form growth rate is provided. Solving the equation numerically yields several results, the most important of which confirms the findings of Perkins and Doles III [1975] for the generalized profiles. The presence of velocity shear has a stabilizing effect on the GDI, acting preferentially on shorter wavelength modes. The long wavelength limit, $kL \ll 1$, of the F region GDI rate is

$$\gamma_{F \text{ long}} = AkV_{\mathbf{E} \times \mathbf{B}} \quad \omega \ll \nu_{in} \quad (1.17)$$

$$\gamma_{\text{high alt long}} = \sqrt{Ak\nu_{in}V_{\mathbf{E} \times \mathbf{B}}} \quad \omega \gg \nu_{in}, \quad (1.18)$$

where $A = (n_1 - n_2)/(n_1 + n_2)$ is the Atwood number [Huba and Zalesak, 1983]. Note that n_1 is the number density of the heavier fluid and n_2 is the density of the lighter fluid. Eqs. 1.17 and 1.18 are the long wavelength analogs of Eqs. 1.12 and 1.14, respectively. Huba and Lee [1983] extend the generalized theory by Huba et al. [1983] from the F region GDI to the E region GDI, showing similar shear stabilization of short wavelength modes. Thus, the theory

indicates that the GDI will be damped preferentially at shorter wavelengths in the presence of velocity shear (or equivalently an inhomogeneous electric field).

All of the works presented thus far have described the linear theory of the GDI. The linear phase of any instability only exists early in the instability development; afterwards, the instability enters the nonlinear phase where the growing perturbation cannot be modeled as an exponentially growing plane wave. Due to the complicated nature of the instability evolution, simplifying assumptions need to be made to have an understanding of the nonlinear theory. The goal of nonlinear theory is to understand the turbulence energy cascade from large scales to small scales. This cascade follows a power law until reaching some smallest diffusive scale. Many simulation results show the nonlinear growth of the GDI with bifurcations cascading energy to smaller scales with the inclusion of varying amounts of physics [Zabusky et al., 1973, Doles III et al., 1976, McDonald et al., 1980, Bernhardt, 1988, Besse et al., 2005]. Through varying assumptions and the help of numerical modeling, the consensus is that the 2D GDI has an isotropic turbulence cascade with a power law of about -2 with homogeneous electric fields [Chaturvedi and Ossakow, 1979, Keskinen and Ossakow, 1981, 1982, 1983a, Keskinen and Huba, 1990] and with inhomogeneous electric fields [Keskinen, 1984]. Additionally, the effect of inertia has been modeled to obtain an understanding of the nonlinear GDI behavior, which typically results in the development of secondary Kelvin-Helmholtz instabilities [Mitchell Jr et al., 1985b, Gondarenko and Guzdar, 1999]. Furthermore, three-dimensional models have attempted to capture the altitudinal variations in GDI growth [Zalesak et al., 1990, Gondarenko and Guzdar, 2006, Besse et al., 2007].

1.4 Kelvin-Helmholtz Instability (KHI)

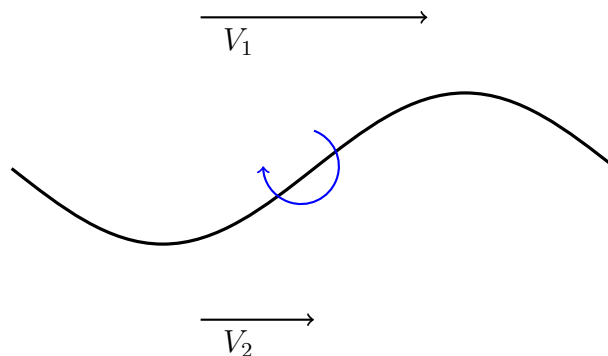


Figure 1.3: A diagram of the Kelvin-Helmholtz instability (KHI) growth mechanism. When there exists a shear flow at an interface, a small perturbation will grow and roll up into a vortex. Based on the velocity vectors shown here, a clockwise vortex will form. If the shear velocity vector is in the opposite direction, a counterclockwise vortex will form.

The KHI is an ubiquitous instability in the universe occurring at many scales. It has been observed in astrophysical flows in the intergalactic medium [Vietri et al., 1997], in atmospheric flows on Earth [Browning, 1971], in quantum fluids [Takeuchi et al., 2010], and the scales in between. It often occurs as a secondary instability to instabilities such as the Rayleigh-Taylor instability [Umeda and Wada, 2016], the Richtmyer-Meshkov instability [Zhang et al., 2020], and, most importantly for this work, the gradient drift instability [Keskinen et al., 1988, Gondarenko and Guzdar, 1999].

The Kelvin-Helmholtz instability (KHI) is a shear-driven hydrodynamic instability. Figure 1.3 provides a diagram of how the KHI grows. At a velocity shear interface, a perturbation will begin to roll up into a vortex. In Figure 1.3, a clockwise vortex would form. If the velocity shear is in the opposite direction, the vortex would be counterclockwise. While not required for growth, the KHI is also typically associated with density gradients.

For an inviscid hydrodynamic flow, the growth rate of the KHI in an inhomogeneous fluid is

$$\gamma = \sqrt{\frac{\rho_1 \rho_2 (V_1 - V_2)^2 k^2}{(\rho_1 + \rho_2)^2}}, \quad (1.19)$$

where ρ is the mass density, V is the velocity, k is the wavenumber, and the subscripts 1 and 2 correspond to the values of the respective side of the interface [Chandrasekhar, 1961, Chap. 11]. In an inviscid magnetohydrodynamic flow, the presence of a magnetic field in the direction of the velocity shear acts to stabilize growth while magnetic fields in the perpendicular directions do not impact KHI growth [Chandrasekhar, 1961, Chap. 11]. Wang et al. [2010] show that the KHI growth rate increases when the velocity gradient scale length tends to 0, or when the density gradient scale length tends to ∞ . Viscosity has a stabilizing effect on the KHI [Kim and Kim, 2017]. Similarly, Keskinen et al. [1988] show that for an $\mathbf{E} \times \mathbf{B}$ driven ionospheric F region plasma with a large density gradient scale length, the KHI becomes damped when collisions with neutral particles are increased. The increased collisions prevent the rollup of the vortices and lead to wavebreaking. However, for small density gradient scale lengths, the effect of collisions can actually increase the growth of the KHI; this regime of the instability has been called the collisional shear instability [Hysell and Kudeki, 2004]. The velocity shear acts as the free energy mechanism that creates charge separations from the collisions, similar to the development of the GDI. The KHI typically develops at higher altitudes because of the lower collisionality and the greater importance of inertia [Ganguli et al., 1994, Spicher et al., 2020]. Of specific importance to this work is understanding the possible KHI development at high altitudes in subauroral polarization streams [Foster and Burke, 2002]. This is discussed in greater detail in Chapter 5.

1.5 Motivation and Objectives

This work is motivated by the need to better understand the cause of ionospheric turbulence in the subauroral and mid-latitude regions and its impact on modern technologies such as communication and navigation systems. While many global ionospheric models exist, their best resolutions (e.g., 0.625° for TIE-GCM [Dang et al., 2020]) are not sufficient to resolve the scales needed to understand the turbulence affecting High Frequency (HF) radar and GPS measurements (10s and 100s of meters, respectively). There is a need for the development and application of local computational models in the community to understand ionospheric disturbances that are hypothesized to impact plasma dynamics at these spatial scales. The computational model and simulation tools developed in this work fill a void to address some of these critical issues. In particular, the developed model is well suited as a foundation to study the GDI and the KHI for a range of subauroral and mid-latitude scenarios.

The model is used to study density irregularities associated with subauroral polarization streams [Foster and Burke, 2002, Mishin and Blaunstein, 2008]. The GDI is hypothesized to be a cause of these irregularities [Mishin et al., 2003, Mishin and Blaunstein, 2008]. This is discussed in greater detail in Chapter 4.

From a theoretical perspective, this model is used to study how GDI and KHI growth are affected by different background conditions that may be present in the ionosphere, specifically the effect of collision frequency (and thus the effect of altitude). This is discussed in greater detail in Chapter 5.

Chapter 2

Total Solar Eclipse

On August 17, 2017, a total solar eclipse passed through the continental United States of America. This generated much interest both within the public (it was the first total solar eclipse to cover the bulk of the continental United States since 1918) and the scientific community. The path of the eclipse was favorable for taking an array of measurements to study the ionospheric and atmospheric response to a total solar eclipse.

One of the primary ionization mechanisms in the ionosphere is photoionization, mainly from extreme ultraviolet (EUV) radiation. As the Moon passes between the Sun and the Earth, it blocks some of the incoming radiation, resulting in a localized decrease in the photoionization rate. This decrease is not accompanied by an equivalent decrease in ion loss mechanisms (dissociative or radiative recombination). Generally, this results in the ion density decreasing locally in the umbra (and less so in the penumbra) [Rishbeth, 1968]. There are many other effects that the eclipse has on the ionosphere and atmosphere, such as causing a bow shock wave due to the shadow's supersonic speed [Chimonas, 1970, Zhang et al., 2017], or lowering the temperature in the umbra [Rishbeth, 1968], to name a few. However, the focus for this work relies on the production of density gradients in the ionosphere.

Section 1.3 described how the GDI grows in the presence of electric fields, magnetic fields, and density gradients under certain geometries. Since the density gradients caused by the solar eclipse result in a generally circular geometry, it is hypothesized that there may be regions which are unstable to the GDI. Neglecting the collisional terms, considering primarily the F region, and taking the short wavelength limit ($k \rightarrow \infty$), Eq. 1.15 is simplified to obtain

$$\gamma = \frac{\nabla n}{n} \frac{E}{B} \left[\hat{\mathbf{k}} \cdot (\hat{\mathbf{b}} \times \hat{\mathbf{g}}) \right] \left[\hat{\mathbf{k}} \cdot \hat{\mathbf{e}} \right], \quad (2.1)$$

where E is the magnitude of the electric field, B is the magnitude of the magnetic field, n is the plasma density, $\hat{\mathbf{k}}$ is the wavenumber unit vector, $\hat{\mathbf{b}}$ is the magnetic field unit vector, $\hat{\mathbf{g}}$ is the density gradient unit vector, and $\hat{\mathbf{e}}$ is the electric field unit vector. Since it is unknown what sorts of perturbations might exist, the perturbation is assumed to be

white noise, i.e. every possible wavenumber and wavenumber direction is excited. Eq. 2.1 is independent of the wavenumber magnitude but is dependent on its direction. Geometrically, the wavenumber vector that optimizes Eq. 2.1 bisects the vectors $\hat{\mathbf{b}} \times \hat{\mathbf{g}}$ and $\hat{\mathbf{e}}$ [Makarevich, 2014].

The objective is to calculate this growth rate using Eq. 2.1 as a function of time and space through the use of several global empirical and physics-based models. The Earth’s geomagnetic field is found through the International Geomagnetic Reference Field (IGRF) model. This is an empirical model that is continually updated (presently on 12th generation) to account for temporal changes in the geomagnetic field using magnetic observatories located worldwide [Thébault et al., 2015]. The electric field is assumed to be purely neutral wind driven as $\mathbf{E} = \mathbf{u} \times \mathbf{B}$. The neutral wind in the radial, or up/down, direction is neglected; therefore, only the horizontal wind remains. This wind is found using the Horizontal Wind Model (HWM14), which is an empirical model that uses data collected from many instruments across the planet [Drob et al., 2015].

The remaining values needed to calculate Eq. 2.1 are all functions of the plasma density. The density is obtained using the model Sami3 is Also a Model of the Ionosphere (SAMI3)¹ developed by the Naval Research Laboratory (NRL). SAMI3 is a fully 3D model that couples the ionosphere and the plasmasphere, and includes many different plasma and neutral species with appropriate collisions and reaction rates for the plasma chemistry. The model uses inputs from HWM14 (described above) [Drob et al., 2015] to obtain the neutral winds. The thermospheric parameters are obtained from the empirical NRL Mass Spectrometer Incoherent Scatter Radar-00 (NRLMSISE-00) model [Picone et al., 2002]. The magnetic field is obtained using the Richmond apex model [Richmond, 1995]. The EUV flux is obtained from the empirical EUV flux model for aeronomic calculations (EUVAC) [Richards et al., 1994]. With these inputs, SAMI3 evolves a set of continuity, momentum, and energy equations for different ion species, different neutral species, and electrons [Huba et al., 2000, SAMI3 is based on SAMI2]. Recent work on the model has relaxed some of the assumptions and introduces several metallic ion species (Mg^+ and Fe^+) to facilitate extensions into lower altitudes [Huba et al., 2019]. SAMI3 has been used to study equatorial plasma bubbles [Huba and Joyce, 2010], the plasmasphere [Huba and Krall, 2013], and most importantly for the this work, the August 17, 2017 solar eclipse [Huba and Drob, 2017].

Huba and Drob [2017] modeled the eclipse by using a mask to inhibit the incoming EUV radiation flux. The obscuration factor is calculated using Naval Observatory Vector Astronomy Software (NOVAS) [Kaplan et al., 2012] to get the positions of the Sun and Moon. While the visible and near UV photon flux is nearly constant across the Sun’s surface, X-ray and EUV radiation, which are more important for photoionization, vary across the Sun’s surface and extend into the corona [Marriott et al., 1971]. Effective obscuration factors for light at 171 Å are found using the Solar Dynamics Observatory (SDO) Atmospheric Imaging Assembly (AIA) [Lemen et al., 2011]. The SAMI3 results predicted a decrease in the F

¹Yes, this is a humorous self-referential model name.

region density by 50% and a time lag in the TEC drop, as well as some conjugate effects in the southern hemisphere². Some of these predictions, such as the density drop and the time lag, have been verified by observations [Mrak et al., 2018, Yang et al., 2018].

For this study, a simulation is run using SAMI3 using essentially the same method as Huba and Drob [2017], but with updated parameters for the August 17, 2017 solar eclipse. The solar flux indicator, F10.7, is set to 90 SFU and the magnetic field Ap index is set to 4, indicating quiet time conditions. The ephemeris calculator is improved to consider the angle of the incoming photon flux to create a mask that affects the entire simulation domain. The simulation is run to gain an understanding of how the electron density is affected by the solar eclipse, shown in the left panels of Figure 2.1. The different outputs from all the models used (SAMI3, HWM14, and IGRF) are transformed into a spherical coordinate system defined such that r is the radial direction, ϕ is the polar angle, and θ is the azimuthal angle. The density gradient is calculated using a second order central difference approximation. Since the goal of this study is to examine the effect of the horizontal density gradients caused by the eclipse, the derivative with respect to altitude, $\partial n/\partial r$, is neglected, leaving ∇n in the $\phi\theta$ “plane”. The GDI growth rate can be calculated using Eq. 2.1 with the optimal wavenumber direction. To properly understand the possible effect the GDI may have during the eclipse, it is compared to the eclipse time rate, $1/\tau$, which is defined as

$$\frac{1}{\tau} = \frac{1}{n} \frac{\partial n}{\partial t}. \quad (2.2)$$

This is analogous to the inverse density gradient scale length, but for time. The temporal density derivative is also calculated using a second order central difference approximation.

Figure 2.1 shows the results over time with the plasma density on the left, the maximum GDI growth rate in the middle, and the eclipse time rate on the right. The effect of the eclipse is clearly observed as a significant decrease in the electron density, which can be seen passing over the continental United States. The resulting horizontal density gradients create hotspots in the GDI growth rate approximately co-located with the trailing edge of the density depletion generated by the eclipse. Intuitively, the density is expected to be decreasing at the leading edge of the eclipse and be increasing at the trailing edge. This is precisely what is shown in the eclipse time rate on the right plots in Figure 2.1. Qualitatively, it would seem that the GDI has a higher chance of occurring during a solar eclipse. While the growth rate is certainly higher than that of an undisturbed ionosphere, it is still about an order of magnitude lower than the eclipse time rate. This suggests that any density gradients that might cause the GDI are not expected to exist long enough for any actual instability growth. The eclipse occurs over a few hours, but the effects of the GDI are expected to be observed over approximately one day. By the time the GDI might begin to grow, the ionosphere will have already returned to normal conditions. Therefore, the GDI is not expected to grow and be observable during a total solar eclipse [Rathod et al., 2020].

²Note that the study by Huba and Drob [2017] was conducted before the eclipse occurred and therefore was making several predictions.

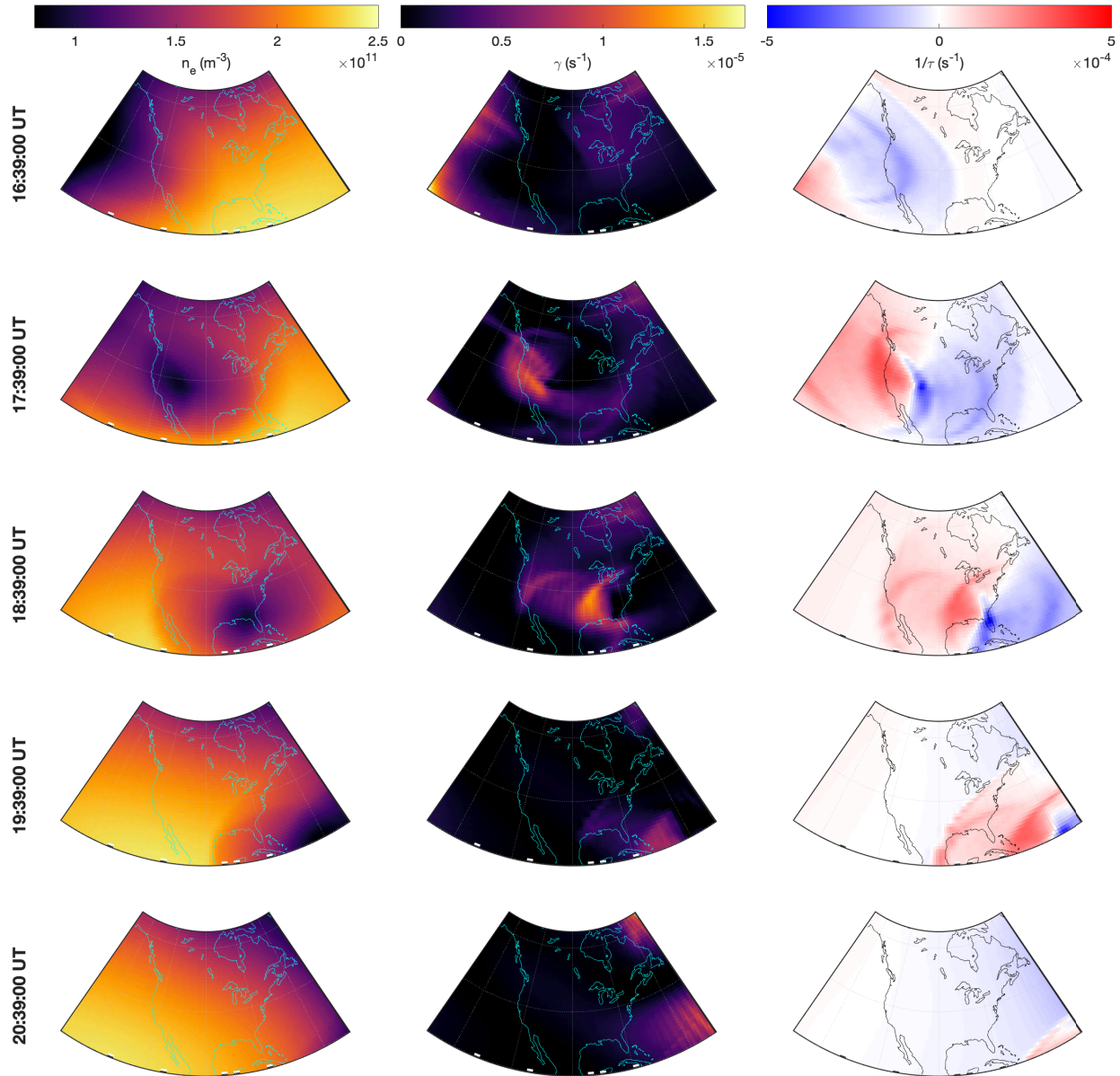


Figure 2.1: Plots of the electron density (left), GDI growth rate (middle), and eclipse time rate (right). The eclipse can be seen as an approximately circular decrease in the density moving across the continental United States. In the eclipse region, there is a hot spot for possible GDI growth. However, comparing the magnitudes of the growth rate ($\approx 10^{-5}$) and the eclipse time rate ($\approx 10^{-4}$) suggests that the GDI does not cause any ionospheric turbulence. The effect of the eclipse occurs significantly faster than any possible GDI growth.

Chapter 3

Model

This chapter discusses the derivation of the mathematical model that becomes the foundation for the numerical work in Chapters 4 and 5. The mathematical model is verified by conducting a local linear analysis, which is compared to known results in the literature. The computational model is verified by showing that the components follow the expected order of accuracy. Finally, several benchmark problems are run to demonstrate the model's capabilities.

3.1 Mathematical Model

3.1.1 Assumptions

The primary application of the model is to study F region electrostatic instability development and consequential turbulence generation surrounding large scale ionospheric phenomena. Some examples are studying irregularities in plasma clouds (Section 3.6.3), polar cap patches [Weber et al., 1984], and subauroral polarization streams (Chapter 4). This model is presently most suited to studying the gradient drift and Kelvin-Helmholtz instabilities.

The mean free path, λ_{MFP} , is the average distance between collisions for a particle. In the case where λ_{MFP} is much smaller than the smallest resolvable scale size in the model, it can be assumed that the plasma undergoes sufficient collisions to be thermalized, i.e., to follow a Maxwellian distribution. Under these conditions, the plasma can then be modeled as a fluid considering only the bulk properties.

If the smallest resolvable scale length in the model is much larger than the Debye length (Eq. 1.2), then the plasma can be considered quasineutral, i.e., $n_i = n_e = n$. For problems involving low frequency waves with no magnetic field perturbations (electrostatic), then the electric field can be found using the equations of motion of the plasma [Chen et al., 1984].

In this case, the electric field can be found using a current continuity equation. If high frequency waves and magnetic perturbations are considered, then Maxwell's equations need to be solved to obtain the electric and magnetic fields. Furthermore, the ionosphere is a low β plasma and therefore does not affect the geomagnetic field. For example, Alfvén waves propagate quickly in low β plasmas at time scales much shorter than the time scales of the plasma instabilities this model is developed to study. Furthermore, some plasma waves, such as the Alfvén wave, propagate parallel to the magnetic field, and therefore cannot be captured by this model, which considers only the motion perpendicular to the magnetic field.

With appropriate domain sizes and number of grid points (and perhaps parallelization of the code), the model is expected to resolve scales of 10s to 100s of meters. At the smallest scales, additional terms may need to be added to consider the kinetic effects.

The model assumes a 2D planar Cartesian geometry. For large domain sizes, effects involving the curvature of the planet would need to be considered. The model only considers motion perpendicular to the magnetic field. Therefore, it is most applicable for use in mid to high latitudes to study an altitudinal slice of the ionosphere. For equatorial latitudes, the perpendicular 2D domain would instead be a latitudinal slice of the ionosphere. To appropriately model this, the effects of gravity and altitudinal variations¹ would need to be considered. The model is currently being extended to 3D, however, novel investigations of subauroral ionospheric turbulence can be made with the current 2D version of the model (see Chapters 4 and 5).

This model does not evolve the neutral particles and only considers them as a source term. The neutral wind is assumed to be constant spatially and temporally. This simplifying assumption is useful for studying the GDI and the KHI from a fundamental physics perspective. The model can be easily modified to consider a spatially and temporally varying neutral wind. A spatially and temporally varying neutral wind can cause spatial and temporal variations in the growth of the GDI and the KHI; future work can consider these effects applied to subauroral polarization streams [Zhang et al., 2015].

This model is an adaptation of the model developed by Keskinen et al. [2004]. To fully study varying altitudes and associated collisionality in F region, inertial effects are considered. Inertial effects are important to allow the development of the KHI, to examine secondary and nonlinear motion of the GDI, and to study the turbulence cascade.

¹Altitudinal variations may need a different discretization method since the Fourier basis used in this model (described in Section 3.4.1) requires periodic boundary conditions.

The continuity, momentum, and energy equations are

$$\frac{\partial n_\alpha}{\partial t} + \nabla \cdot (n_\alpha \mathbf{V}_\alpha) = D \nabla^2 n_\alpha \quad (3.1)$$

$$n_\alpha \left(\frac{\partial}{\partial t} + \mathbf{V}_\alpha \cdot \nabla \right) \mathbf{V}_\alpha = \frac{q_\alpha n_\alpha}{m_\alpha} (\mathbf{E} + \mathbf{V}_\alpha \times \mathbf{B}) - \frac{\nabla P_\alpha}{m_\alpha} - n_\alpha \nu_{\alpha n} (\mathbf{V}_\alpha - \mathbf{u}) - n_\alpha \nu_{\alpha \xi} (\mathbf{V}_\alpha - \mathbf{V}_\xi) \quad (3.2)$$

$$\frac{3}{2} n_\alpha k_B \frac{\partial T_\alpha}{\partial t} + \frac{3}{2} n_\alpha k_B \mathbf{V}_\alpha \cdot \nabla T_\alpha + n_\alpha k_B T_\alpha \nabla \cdot \mathbf{V}_\alpha = -\nabla \cdot \mathbf{q}_\alpha + Q_\alpha, \quad (3.3)$$

where n is number density, \mathbf{V} is velocity, D is the numerical diffusion constant, q is electric charge, m is mass, \mathbf{E} is electric field, \mathbf{B} is magnetic field, P is pressure, ν is collision frequency, T is temperature, k_B is the Boltzmann constant, \mathbf{q} is heat flux, Q is frictional heating, the subscript α denotes the species (ions or electrons), the subscript ξ denotes the other species, and the subscript n denotes neutrals. The equation of state is the ideal gas law,

$$P_\alpha = n_\alpha k_B T_\alpha. \quad (3.4)$$

The collision frequencies are

$$\nu_{in} = n_n V_{Th_i} \sigma_{in} \quad (3.5)$$

$$\nu_{en} = n_n V_{Th_e} \sigma_n \quad (3.6)$$

$$\nu_{ee} = \frac{n e^4}{2\pi \epsilon_0^2 m_e^2 V_{Th_e}^3} \ln \frac{\Lambda}{3} \quad (3.7)$$

$$\nu_{ii} = \nu_{ee} \sqrt{\frac{m_e}{m_i}} \quad (3.8)$$

$$\nu_{ie} = \nu_{ee} \frac{m_e}{m_i} \quad (3.9)$$

$$\nu_{ei} = \nu_{ee}, \quad (3.10)$$

where σ is collision cross section, V_{Th_α} is thermal velocity (Eq. 1.4), and Λ is the plasma parameter ($12\pi n \lambda_D^3$, where λ_D is the Debye length from Eq. 1.2). These frequencies are self-consistently calculated in the model based on the density and temperatures.

3.1.2 Velocities

The velocities can be split into ordered terms of the form ν/Ω_c or $(\partial/\partial t + \mathbf{V} \cdot \nabla)/\Omega_c$, where Ω_c is the gyrofrequency. In the F region, these terms are considered to be small and therefore orders greater than 1 can be neglected. Consider the momentum equation, Eq. 3.2, for the electrons with $q_e = -e$. Taking the right cross product with \mathbf{B} yields

$$n_e \left(\frac{\partial}{\partial t} + \mathbf{V}_e \cdot \nabla \right) \mathbf{V}_e \times \mathbf{B} = \frac{-en_e}{m_e} (\mathbf{E} + \mathbf{V}_e \times \mathbf{B}) \times \mathbf{B} - \frac{\nabla P_e}{m_e} \times \mathbf{B} - n_e \nu_{en} (\mathbf{V}_e - \mathbf{u}) \times \mathbf{B} - n_e \nu_{ei} (\mathbf{V}_e - \mathbf{V}_i) \times \mathbf{B}. \quad (3.11)$$

The vector identity for the triple cross product,

$$(\mathbf{a} \times \mathbf{b}) \times \mathbf{c} = -(\mathbf{c} \cdot \mathbf{b})\mathbf{a} + (\mathbf{c} \cdot \mathbf{a})\mathbf{b}, \quad (3.12)$$

is used frequently in this derivation. Since this model only considers motion perpendicular to \mathbf{B} , any other vector dotted with \mathbf{B} is 0. The term $\mathbf{V}_e \times \mathbf{B} \times \mathbf{B}$ is

$$\mathbf{V}_e \times \mathbf{B} \times \mathbf{B} = \left[-(\mathbf{B} \cdot \mathbf{B})\mathbf{V}_e + (\mathbf{V}_e \cdot \mathbf{B})\mathbf{B} \right] = -B^2\mathbf{V}_e. \quad (3.13)$$

Therefore, solving Eq. 3.11 for \mathbf{V}_e yields

$$\begin{aligned} \mathbf{V}_e = & \frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} + \frac{m_e}{eB^2} \nu_{en} (\mathbf{V}_e - \mathbf{u}) \times \mathbf{B} \\ & + \frac{m_e}{eB^2} \nu_{ei} (\mathbf{V}_e - \mathbf{V}_i) \times \mathbf{B} + \frac{m_e}{eB^2} \left(\frac{\partial}{\partial t} + \mathbf{V}_e \cdot \nabla \right) \mathbf{V}_e \times \mathbf{B}. \end{aligned} \quad (3.14)$$

The substitutions $\mathbf{V}_e = \mathbf{V}_e^0 + \mathbf{V}_e^1$ and $m_e/eB = 1/\Omega_{ce}$ are used to rewrite the electron velocity in ordered terms resulting in

$$\begin{aligned} \mathbf{V}_e^0 + \mathbf{V}_e^1 = & \frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} + \frac{\nu_{en}}{B\Omega_{ce}} (\mathbf{V}_e - \mathbf{u}) \times \mathbf{B} \\ & + \frac{\nu_{ei}}{B\Omega_{ce}} (\mathbf{V}_e - \mathbf{V}_i) \times \mathbf{B} + \frac{1}{B\Omega_{ce}} \left(\frac{\partial}{\partial t} + \mathbf{V}_e \cdot \nabla \right) \mathbf{V}_e \times \mathbf{B}. \end{aligned} \quad (3.15)$$

The zeroth order velocity is found by neglecting the terms of first order and higher, resulting in

$$\mathbf{V}_e^0 = \frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2}, \quad (3.16)$$

which is the summation of two fundamental plasma drifts: the $\mathbf{E} \times \mathbf{B}$ drift and the diamagnetic drift. The first order velocity is found by neglecting the terms of second order and higher. Splitting the ion and electron velocities into zeroth and first order terms yields

$$\begin{aligned} \mathbf{V}_e^1 = & \frac{\nu_{en}}{B\Omega_{ce}} (\mathbf{V}_e^0 + \mathcal{V}_e^1 - \mathbf{u}) \times \mathbf{B} + \frac{\nu_{ei}}{B\Omega_{ce}} (\mathbf{V}_e^0 + \mathcal{V}_e^1 - \mathbf{V}_i^0 - \mathcal{V}_i^1) \times \mathbf{B} \\ & + \frac{1}{B\Omega_{ce}} \left(\frac{\partial}{\partial t} + [\mathbf{V}_e^0 + \mathcal{V}_e^1] \cdot \nabla \right) (\mathbf{V}_e^0 + \mathcal{V}_e^1) \times \mathbf{B}. \end{aligned} \quad (3.17)$$

Note that all first order velocities on the right hand side can be neglected because they are multiplied by first order terms. The resulting first order electron velocity is

$$\mathbf{V}_e^1 = \frac{\nu_{en}}{B\Omega_{ce}} (\mathbf{V}_e^0 - \mathbf{u}) \times \mathbf{B} + \frac{\nu_{ei}}{B\Omega_{ce}} (\mathbf{V}_e^0 - \mathbf{V}_i^0) \times \mathbf{B} + \frac{1}{B\Omega_{ce}} \left(\frac{\partial}{\partial t} + \mathbf{V}_e^0 \cdot \nabla \right) \mathbf{V}_e^0 \times \mathbf{B}. \quad (3.18)$$

However, in order to evaluate the right hand side, the zeroth order ion velocity needs to be determined. Therefore, consider the momentum equation, Eq. 3.2, for ions with $q_i = e$. Taking the right cross product with \mathbf{B} yields

$$n_i \left(\frac{\partial}{\partial t} + \mathbf{V}_i \cdot \nabla \right) \mathbf{V}_i \times \mathbf{B} = \frac{en_i}{m_i} (\mathbf{E} + \mathbf{V}_i \times \mathbf{B}) \times \mathbf{B} - \frac{\nabla P_i}{m_i} \times \mathbf{B} \\ - n_i \nu_{in} (\mathbf{V}_i - \mathbf{u}) \times \mathbf{B} - n_i \nu_{ie} (\mathbf{V}_i - \mathbf{V}_e) \times \mathbf{B}. \quad (3.19)$$

Using Eq. 3.12,

$$\mathbf{V}_i \times \mathbf{B} \times \mathbf{B} = \left[-(\mathbf{B} \cdot \mathbf{B}) \mathbf{V}_i + (\mathbf{V}_i \cdot \mathbf{B}) \mathbf{B} \right] = -B^2 \mathbf{V}_i. \quad (3.20)$$

Therefore, solving Eq. 3.19 for \mathbf{V}_i gives

$$\mathbf{V}_i = \frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} - \frac{m_i}{e B^2} \nu_{in} (\mathbf{V}_i - \mathbf{u}) \times \mathbf{B} \\ - \frac{m_i}{e B^2} \nu_{ie} (\mathbf{V}_i - \mathbf{V}_e) \times \mathbf{B} - \frac{m_i}{e B^2} \left(\frac{\partial}{\partial t} + \mathbf{V}_i \cdot \nabla \right) \mathbf{V}_i \times \mathbf{B}. \quad (3.21)$$

The substitutions $\mathbf{V}_i = \mathbf{V}_i^0 + \mathbf{V}_i^1$ and $m_i/eB = 1/\Omega_{ci}$ are used to write the ion velocity in ordered terms, resulting in

$$\mathbf{V}_i^0 + \mathbf{V}_i^1 = \frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} - \frac{\nu_{in}}{B \Omega_{ci}} (\mathbf{V}_i - \mathbf{u}) \times \mathbf{B} \\ - \frac{\nu_{ie}}{B \Omega_{ci}} (\mathbf{V}_i - \mathbf{V}_e) \times \mathbf{B} - \frac{1}{B \Omega_{ci}} \left(\frac{\partial}{\partial t} + \mathbf{V}_i \cdot \nabla \right) \mathbf{V}_i \times \mathbf{B}. \quad (3.22)$$

The zeroth order velocity is found by neglecting the terms of first order and higher, resulting in

$$\mathbf{V}_i^0 = \frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2}, \quad (3.23)$$

which is also a summation of the $\mathbf{E} \times \mathbf{B}$ and diamagnetic drifts. Now that the zeroth order ion velocity is known, the first order electron velocity can be found. Substituting Eqs. 3.16 and 3.23 into Eq. 3.18 yields

$$\mathbf{V}_e^1 = \frac{\nu_{en}}{B \Omega_{ce}} \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} - \mathbf{u} \right) \times \mathbf{B} \\ + \frac{\nu_{ei}}{B \Omega_{ce}} \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} - \frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} \right) \times \mathbf{B} \\ + \frac{1}{B \Omega_{ce}} \left(\frac{\partial}{\partial t} + \left[\frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} \right] \cdot \nabla \right) \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} \right) \times \mathbf{B}. \quad (3.24)$$

For the F region phenomena considered, the diamagnetic drift is much smaller than the $\mathbf{E} \times \mathbf{B}$ drift, and is therefore neglected in the convective derivative term. The $\mathbf{E} \times \mathbf{B}$ drift is

defined as $\mathbf{V}_{\mathbf{E} \times \mathbf{B}}$. With prodigious use of Eq. 3.12, the first order electron velocity is

$$\mathbf{V}_e^1 = \frac{\nu_{en}}{B\Omega_{ce}} \left(-\mathbf{E} - \frac{\nabla P_e}{en_e} - \mathbf{u} \times \mathbf{B} \right) - \frac{\nu_{ei}}{B\Omega_{ce}} \left(\frac{\nabla P_e}{en} + \frac{\nabla P_i}{en_i} \right) - \frac{1}{B\Omega_{ce}} \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E}. \quad (3.25)$$

This is simplified to

$$\mathbf{V}_e^1 = -\frac{1}{B\Omega_{ce}} \left[\left(\nu_{en} + \nu_{ei} \right) \frac{\nabla P_e}{en_e} + \nu_{ei} \frac{\nabla P_i}{en_i} + \nu_{en} \left(\mathbf{E} + \mathbf{u} \times \mathbf{B} \right) + \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E} \right]. \quad (3.26)$$

The first order ion velocity is found by expanding the velocities in Eq. 3.22 and neglecting higher order terms, resulting in

$$\begin{aligned} \mathbf{V}_i^1 = & -\frac{\nu_{in}}{B\Omega_{ci}} \left(\mathbf{V}_i^0 + \mathcal{Y}_i^1 - \mathbf{u} \right) \times \mathbf{B} - \frac{\nu_{ie}}{B\Omega_{ci}} \left(\mathbf{V}_i^0 + \mathcal{Y}_i^1 - \mathbf{V}_e^0 - \mathcal{Y}_e^1 \right) \times \mathbf{B} \\ & - \frac{1}{B\Omega_{ci}} \left(\frac{\partial}{\partial t} + [\mathbf{V}_i^0 + \mathcal{Y}_i^1] \cdot \nabla \right) \left(\mathbf{V}_i^0 + \mathcal{Y}_i^1 \right) \times \mathbf{B}. \end{aligned} \quad (3.27)$$

Substituting in Eqs. 3.16 and 3.23 yields

$$\begin{aligned} \mathbf{V}_i^1 = & -\frac{\nu_{in}}{B\Omega_{ci}} \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} - \mathbf{u} \right) \times \mathbf{B} \\ & - \frac{\nu_{ie}}{B\Omega_{ci}} \left(\frac{\mathbf{E} \times \mathcal{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} - \frac{\mathbf{E} \times \mathcal{B}}{B^2} - \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} \right) \times \mathbf{B} \\ & - \frac{1}{B\Omega_{ci}} \left(\frac{\partial}{\partial t} + \left[\frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathcal{B}}{en_i B^2} \right] \cdot \nabla \right) \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathcal{B}}{en_i B^2} \right) \times \mathbf{B}. \end{aligned} \quad (3.28)$$

Using Eq. 3.12, the first order ion velocity is

$$\mathbf{V}_i^1 = \frac{\nu_{in}}{B\Omega_{ci}} \left(\mathbf{E} - \frac{\nabla P_i}{en_i} + \mathbf{u} \times \mathbf{B} \right) - \frac{\nu_{ie}}{B\Omega_{ci}} \left(\frac{\nabla P_i}{en_i} + \frac{\nabla P_e}{en_e} \right) + \frac{1}{B\Omega_{ci}} \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E}. \quad (3.29)$$

This is simplified to

$$\mathbf{V}_i^1 = \frac{1}{B\Omega_{ci}} \left[-\nu_{ie} \frac{\nabla P_e}{en_e} - \left(\nu_{in} + \nu_{ie} \right) \frac{\nabla P_i}{en_i} + \nu_{in} \left(\mathbf{E} + \mathbf{u} \times \mathbf{B} \right) + \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E} \right]. \quad (3.30)$$

The zeroth and first order velocities, rewritten here for convenience, are

$$\mathbf{V}_e^0 = \frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} \quad (3.31)$$

$$\mathbf{V}_e^1 = -\frac{1}{B\Omega_{ce}} \left[\left(\nu_{en} + \nu_{ei} \right) \frac{\nabla P_e}{en_e} + \nu_{ei} \frac{\nabla P_i}{en_i} + \nu_{en} (\mathbf{E} + \mathbf{u} \times \mathbf{B}) + \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E} \right] \quad (3.32)$$

$$\mathbf{V}_i^0 = \frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} \quad (3.33)$$

$$\mathbf{V}_i^1 = \frac{1}{B\Omega_{ci}} \left[-\nu_{ie} \frac{\nabla P_e}{en_e} - \left(\nu_{in} + \nu_{ie} \right) \frac{\nabla P_i}{en_i} + \nu_{in} (\mathbf{E} + \mathbf{u} \times \mathbf{B}) + \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E} \right]. \quad (3.34)$$

3.1.3 Current Closure

The continuity, momentum, and energy equations (Eqs. 3.1 to 3.3), due to quasineutrality, are closed using a current continuity equation,

$$\nabla \cdot \mathbf{J} = 0, \quad (3.35)$$

where the current density, \mathbf{J} , is defined as

$$\mathbf{J} = n_i q_i \mathbf{V}_i + n_e q_e \mathbf{V}_e = ne (\mathbf{V}_i - \mathbf{V}_e). \quad (3.36)$$

The zeroth and first order velocities are considered in obtaining \mathbf{J} . Therefore,

$$\mathbf{J} = ne (\mathbf{V}_i^0 + \mathbf{V}_i^1 - \mathbf{V}_e^0 - \mathbf{V}_e^1). \quad (3.37)$$

This derivation involves many terms, so it is broken down into smaller parts and put back together at the end. First, the zeroth order current density is

$$\begin{aligned} \mathbf{J}^0 &= ne (\mathbf{V}_i^0 - \mathbf{V}_e^0) \\ &= ne \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{ne B^2} - \frac{\mathbf{E} \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{ne B^2} \right) \\ &= -\frac{(\nabla P_i + \nabla P_e) \times \mathbf{B}}{B^2}. \end{aligned} \quad (3.38)$$

Next, the first order current density is found. This is further split into smaller terms. Based on Eqs. 3.9 and 3.10, the useful identity of $\nu_{ie}/\Omega_{ci} = \nu_{ei}/\Omega_{ce}$ is found, which can be used to simplify the current density terms. The first order current density term with ∇P_i is

$$\begin{aligned} \mathbf{J}_{\nabla P_i}^1 &= \frac{ne}{B} \left(-\frac{\cancel{\nu_{ie}} + \nu_{in}}{\Omega_{ci}} \frac{\nabla P_i}{en} + \frac{\cancel{\nu_{ei}}}{\Omega_{ce}} \frac{\nabla P_i}{en} \right) \\ &= -\frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla P_i}{B}. \end{aligned} \quad (3.39)$$

Using the same identity of the cross species collision frequencies, the first order term with ∇P_e is

$$\begin{aligned}\mathbf{J}_{\nabla P_e}^1 &= \frac{ne}{B} \left(-\frac{\nu_{ie}}{\Omega_{ci}} \frac{\nabla P_e}{en} + \frac{\nu_{en} + \nu_{ei}}{\Omega_{ce}} \frac{\nabla P_e}{en} \right) \\ &= \frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla P_e}{B}.\end{aligned}\quad (3.40)$$

The first order term with the $\mathbf{E} + \mathbf{u} \times \mathbf{B}$ is

$$\mathbf{J}_{\mathbf{E}}^1 = \frac{ne}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) (\mathbf{E} + \mathbf{u} \times \mathbf{B}). \quad (3.41)$$

The first order term with the convective derivative is

$$\mathbf{J}_{conv}^1 = \frac{ne}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E}. \quad (3.42)$$

After combining Eqs. 3.38 to 3.42, the total current density is

$$\begin{aligned}\mathbf{J} &= -\frac{(\nabla P_i + \nabla P_e) \times \mathbf{B}}{B^2} - \frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla P_i}{B} + \frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla P_e}{B} \\ &\quad + \frac{ne}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) (\mathbf{E} + \mathbf{u} \times \mathbf{B}) + \frac{ne}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \mathbf{E}.\end{aligned}\quad (3.43)$$

Since the model is intended to study low frequency electrostatic instabilities that do not have magnetic field perturbations, the electric field can be defined with an electric potential as

$$\mathbf{E} = -\nabla \phi. \quad (3.44)$$

Substituting Eq. 3.44 into Eq. 3.43 yields

$$\begin{aligned}\mathbf{J} &= -\frac{(\nabla P_i + \nabla P_e) \times \mathbf{B}}{B^2} - \frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla P_i}{B} + \frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla P_e}{B} \\ &\quad + \frac{ne}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) (-\nabla \phi + \mathbf{u} \times \mathbf{B}) - \frac{ne}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \nabla \phi.\end{aligned}\quad (3.45)$$

The current closure, found by substituting Eq. 3.45 into Eq. 3.35, is

$$\begin{aligned}\nabla \cdot \mathbf{J} &= \nabla \cdot \left[-\frac{(\nabla P_i + \nabla P_e) \times \mathbf{B}}{B^2} - \frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla P_i}{B} + \frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla P_e}{B} \right. \\ &\quad \left. + \frac{ne}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) (-\nabla \phi + \mathbf{u} \times \mathbf{B}) - \frac{ne}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \nabla \phi \right].\end{aligned}\quad (3.46)$$

Similar to how the current density is found, the current closure is found by looking at each term individually. The vector identity for the divergence of a cross product is

$$\nabla \cdot (\mathbf{a} \times \mathbf{b}) = \mathbf{b} \cdot (\nabla \times \mathbf{a}) - \mathbf{a} \cdot (\nabla \times \mathbf{b}). \quad (3.47)$$

Using this identity, the zeroth order current closure is

$$\begin{aligned}
\nabla \cdot \mathbf{J}^0 &= \nabla \cdot \left[-\frac{(\nabla P_i + \nabla P_e) \times \mathbf{B}}{B^2} \right] \\
&= -\frac{1}{B^2} \nabla \cdot \left[\nabla (P_i + P_e) \times \mathbf{B} \right] \\
&= -\frac{1}{B^2} \left[\mathbf{B} \cdot (\nabla \times \nabla [P_i + P_e]) - \nabla (P_i + P_e) \cdot (\nabla \times \mathbf{B}) \right] \\
&= 0.
\end{aligned} \tag{3.48}$$

Note that because the system is electrostatic, \mathbf{B} is constant everywhere. Additionally, the curl of a gradient is by definition zero. Therefore, the zeroth order current density plays no role in the current closure. The divergence of the ∇P_i term is

$$\nabla \cdot \mathbf{J}_{\nabla P_i}^1 = \nabla \cdot \left[-\frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla P_i}{B} \right] = -\frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla^2 P_i}{B}. \tag{3.49}$$

The divergence of the ∇P_e term is

$$\nabla \cdot \mathbf{J}_{\nabla P_e}^1 = \nabla \cdot \left[\frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla P_e}{B} \right] = \frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla^2 P_e}{B}. \tag{3.50}$$

The divergence of the electric potential and neutral wind term is

$$\begin{aligned}
\nabla \cdot \mathbf{J}_{\mathbf{E}}^1 &= \nabla \cdot \left[\frac{ne}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(-\nabla \phi + \mathbf{u} \times \mathbf{B} \right) \right] \\
&= \frac{e}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(-\nabla \cdot (n \nabla \phi) + \mathbf{u} \times \mathbf{B} \cdot \nabla n \right).
\end{aligned} \tag{3.51}$$

Note that the neutral wind is assumed to be constant in space and time. The divergence of the convective derivative term is

$$\begin{aligned}
\nabla \cdot \mathbf{J}_{conv}^1 &= \nabla \cdot \left[-\frac{ne}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \nabla \phi \right] \\
&= -\frac{e}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left[\nabla \cdot \left(n \frac{\partial \nabla \phi}{\partial t} \right) + \nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right) \right].
\end{aligned} \tag{3.52}$$

Substituting Eqs 3.48 to 3.52 into Eq. 3.46 yields

$$\begin{aligned}
\nabla \cdot \mathbf{J} &= -\frac{\nu_{in}}{\Omega_{ci}} \frac{\nabla^2 P_i}{B} + \frac{\nu_{en}}{\Omega_{ce}} \frac{\nabla^2 P_e}{B} + \frac{e}{B} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(-\nabla \cdot (n \nabla \phi) + \mathbf{u} \times \mathbf{B} \cdot \nabla n \right) \\
&\quad - \frac{e}{B} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left[\nabla \cdot \left(n \frac{\partial \nabla \phi}{\partial t} \right) + \nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right) \right] = 0.
\end{aligned} \tag{3.53}$$

The magnetic field terms cancel. Solving for the time derivative yields

$$\nabla \cdot \left(n \frac{\partial \nabla \phi}{\partial t} \right) = S_\phi, \tag{3.54}$$

where

$$S_\phi = \frac{1}{e} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left(\frac{\nu_{en}}{\Omega_{ce}} \nabla^2 P_e - \frac{\nu_{in}}{\Omega_{ci}} \nabla^2 P_i \right) - \nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right) + \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(\mathbf{u} \times \mathbf{B} \cdot \nabla n - \nabla \cdot (n \nabla \phi) \right). \quad (3.55)$$

3.1.4 Continuity Equation

Since the plasma is quasineutral, only one continuity equation (Eq. 3.1) is needed with the zeroth order velocity used as the advection velocity,

$$\frac{\partial n}{\partial t} + \nabla \cdot (n \mathbf{V}^0) = D \nabla^2 n. \quad (3.56)$$

However, since only one continuity equation is used, the question arises as to which species' zeroth order velocity should be used. The $\mathbf{E} \times \mathbf{B}$ is the same for both species but the diamagnetic drift is not. Due to the constant magnetic field, it can be shown that the diamagnetic drift does not contribute to the continuity equation. The divergence term for an arbitrary species' diamagnetic drift is

$$\begin{aligned} \nabla \cdot (n_\alpha \mathbf{V}_{\nabla P_\alpha \times \mathbf{B}}) &= \nabla \cdot \left(n_\alpha \frac{\nabla P_\alpha \times \mathbf{B}}{q n_\alpha B^2} \right) \\ &= \frac{1}{e B^2} \left[\mathbf{B} (\nabla \times \nabla P_\alpha) - \nabla P_\alpha \cdot (\nabla \times \mathbf{B}) \right] \end{aligned} \quad (3.57)$$

$$= 0. \quad (3.58)$$

Thus, only the species independent $\mathbf{E} \times \mathbf{B}$ drift contributes to the advection in the continuity equation. By showing that the $\mathbf{E} \times \mathbf{B}$ drift is incompressible, the divergence term simplifies to

$$\begin{aligned} \nabla \cdot (n \mathbf{V}_{\mathbf{E} \times \mathbf{B}}) &= n \nabla \cdot \mathbf{V}_{\mathbf{E} \times \mathbf{B}} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla n \\ &= n \nabla \cdot \left(- \frac{\nabla \phi \times \mathbf{B}}{B^2} \right) + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla n \\ &= - \frac{n}{B^2} \left[\mathbf{B} \cdot (\nabla \times \nabla \phi) - \nabla \phi \cdot (\nabla \times \mathbf{B}) \right] + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla n \end{aligned} \quad (3.59)$$

$$= - \frac{\nabla \phi \times \mathbf{B}}{B^2} \cdot \nabla n. \quad (3.60)$$

Therefore, the final continuity equation is

$$\frac{\partial n}{\partial t} - \frac{\nabla \phi \times \mathbf{B}}{B^2} \cdot \nabla n = D \nabla^2 n. \quad (3.61)$$

Note the inclusion of numerical diffusion, which is required to mitigate numerical error that occurs at grid scales. When studying problems involving turbulence and instability development, this error can accumulate at regions with sharp density gradients and eventually lead

to highly unphysical phenomena, such as negative densities or temperatures. The numerical diffusion used in this model preferentially damps higher wavenumber modes throughout the entire domain isotropically. The choice of numerical diffusion constant, D , should be high enough such that the simulation can evolve long enough in time to study the phenomenon in question, but must also be low enough such that the result is still a reasonable representation of nature. The typical process used to find a suitable numerical diffusion constant is to run several simulations for a representative parameter regime with different numerical diffusion constants. The lowest constant that evolves far enough in time should be used.

3.1.5 Energy Equations

The energy equation, Eq. 3.3, can be written in a form which isolates the temporal derivative by dividing by $(3/2)nk_B$. The zeroth order velocities are used for the advection velocities. Therefore, the energy equations are

$$\frac{\partial T_i}{\partial t} + \mathbf{V}_i^0 \cdot \nabla T_i + \frac{2}{3} T_i \nabla \cdot \mathbf{V}_i^0 = \frac{2}{3nk_B} \left(-\nabla \cdot \mathbf{q}_i + Q_i \right) \quad (3.62)$$

$$\frac{\partial T_e}{\partial t} + \mathbf{V}_e^0 \cdot \nabla T_e + \frac{2}{3} T_e \nabla \cdot \mathbf{V}_e^0 = \frac{2}{3nk_B} \left(-\nabla \cdot \mathbf{q}_e + Q_e \right). \quad (3.63)$$

The general formulation of Q_i is

$$Q_i = \sum_j n_i \mu_{ij} \nu_{ij} \left(3k_B (T_j - T_i) + m_j (\mathbf{V}_i - \mathbf{V}_j)^2 \right), \quad (3.64)$$

where the subscript j is an index for the other species and

$$\mu_{ij} = \frac{m_i}{m_i + m_j}. \quad (3.65)$$

The only other species considered are the neutrals and the electrons, which gives a Q_i of

$$Q_i = n \mu_{in} \nu_{in} \left[3k_B (T_n - T_i) + m_n (\mathbf{V}_i^0 - \mathbf{u})^2 \right] + n \mu_{ie} \nu_{ie} \left[3k_B (T_e - T_i) + m_e (\mathbf{V}_i^0 - \mathbf{V}_e^0)^2 \right]. \quad (3.66)$$

This model does not evolve the neutral dynamics. Therefore, the neutrals are considered as a constant background. This effectively creates an infinite source of energy in the frictional heating. For this reason, the $\mathbf{V}_i^0 - \mathbf{u}$ term is not considered. This is similar to the effect of the perturbed model (discussed in greater detail in Section 3.2) in that the background dynamics are not impacted by the neutral wind². Accounting for the denominator in Eq. 3.62, the ion frictional heating source term is

$$\frac{2}{3nk_B} Q_i = 2\mu_{in} \nu_{in} (T_n - T_i) + 2\nu_{ie} \left[(T_e - T_i) + \frac{m_e}{3k_B} (\mathbf{V}_{i0} - \mathbf{V}_{e0})^2 \right]. \quad (3.67)$$

²Note that this method of dealing with the effect of the neutral wind heating on the background dynamics was developed before the development of the perturbed model discussed in Section 3.2. Future work on improving this model should be to fully convert the temperature equations into the perturbed form.

The electron frictional heating term, Q_e , is

$$Q_e = -\frac{3m_e}{2m_i}n\nu_{ei}k_B(T_e - T_i). \quad (3.68)$$

Accounting for the denominator in Eq. 3.63, the electron frictional heating source term is

$$\frac{2}{3nk_B}Q_e = -\frac{m_e}{m_i}\nu_{ei}(T_e - T_i). \quad (3.69)$$

The ion thermal conductivity term, neglecting parallel thermal conduction, is

$$\mathbf{q}_i = -\kappa_i k_B \nabla T_i, \quad (3.70)$$

where κ_i is the thermal conductivity, which is defined as

$$\kappa_i = \frac{nk_B T_i \nu_{ii}}{m_i \Omega_{ci}^2}. \quad (3.71)$$

The ion thermal conduction appears in the source term as

$$-\frac{2}{3nk_B} \nabla \cdot \mathbf{q}_i = \frac{2}{3n} \nabla \cdot (\kappa_i \nabla T_i). \quad (3.72)$$

Similarly, the electron thermal conduction source term is

$$-\frac{2}{3nk_B} \nabla \cdot \mathbf{q}_e = \frac{2}{3n} \nabla \cdot (\kappa_e \nabla T_e). \quad (3.73)$$

Therefore, the full energy equations are

$$\frac{\partial T_i}{\partial t} + \mathbf{V}_{i0} \cdot \nabla T_i + \frac{2}{3} T_i \nabla \cdot \mathbf{V}_{i0} = S_{T_i} \quad (3.74)$$

$$\frac{\partial T_e}{\partial t} + \mathbf{V}_{e0} \cdot \nabla T_e + \frac{2}{3} T_e \nabla \cdot \mathbf{V}_{e0} = S_{T_e}, \quad (3.75)$$

where

$$S_{T_i} = 2\mu_{in}\nu_{in}(T_n - T_i) + 2\nu_{ie} \left[(T_e - T_i) + \frac{m_e}{3k_B} (\mathbf{V}_{i0} - \mathbf{V}_{e0})^2 \right] + \frac{2}{3n} \nabla \cdot (\kappa_i \nabla T_i) \quad (3.76)$$

$$S_{T_e} = -\frac{m_e}{m_i} \nu_{ei} (T_e - T_i) + \frac{2}{3n} \nabla \cdot (\kappa_e \nabla T_e). \quad (3.77)$$

3.1.6 Governing Equations

All of the governing equations are rewritten here for convenience. The velocities are

$$\mathbf{V}_e^0 = -\frac{\nabla\phi \times \mathbf{B}}{B^2} + \frac{\nabla P_e \times \mathbf{B}}{en_e B^2} \quad (3.78)$$

$$\mathbf{V}_e^1 = -\frac{1}{B\Omega_{ce}} \left[(\nu_{en} + \nu_{ei}) \frac{\nabla P_e}{en_e} + \nu_{ei} \frac{\nabla P_i}{en_i} + \nu_{en} \left(-\nabla\phi + \mathbf{u} \times \mathbf{B} \right) - \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \nabla\phi \right] \quad (3.79)$$

$$\mathbf{V}_i^0 = -\frac{\nabla\phi \times \mathbf{B}}{B^2} - \frac{\nabla P_i \times \mathbf{B}}{en_i B^2} \quad (3.80)$$

$$\mathbf{V}_i^1 = \frac{1}{B\Omega_{ci}} \left[-\nu_{ie} \frac{\nabla P_e}{en_e} - (\nu_{in} + \nu_{ie}) \frac{\nabla P_i}{en_i} + \nu_{in} \left(-\nabla\phi + \mathbf{u} \times \mathbf{B} \right) - \left(\frac{\partial}{\partial t} + \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \right) \nabla\phi \right]. \quad (3.81)$$

The current closure equation is

$$\nabla \cdot \left(n \frac{\partial \nabla\phi}{\partial t} \right) = S_\phi, \quad (3.82)$$

where

$$S_\phi = \frac{1}{e} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left(\frac{\nu_{en}}{\Omega_{ce}} \nabla^2 P_e - \frac{\nu_{in}}{\Omega_{ci}} \nabla^2 P_i \right) - \nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla\phi \right) + \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(\mathbf{u} \times \mathbf{B} \cdot \nabla n - \nabla \cdot (n \nabla\phi) \right). \quad (3.83)$$

The continuity, ion energy, and electron energies, respectively, are

$$\frac{\partial n}{\partial t} - \frac{\nabla\phi \times \mathbf{B}}{B^2} \cdot \nabla n = D \nabla^2 n \quad (3.84)$$

$$\frac{\partial T_i}{\partial t} + \mathbf{V}_{i0} \cdot \nabla T_i + \frac{2}{3} T_i \nabla \cdot \mathbf{V}_{i0} = S_{T_i} \quad (3.85)$$

$$\frac{\partial T_e}{\partial t} + \mathbf{V}_{e0} \cdot \nabla T_e + \frac{2}{3} T_e \nabla \cdot \mathbf{V}_{e0} = S_{T_e}, \quad (3.86)$$

where

$$S_{T_i} = 2\mu_{in}\nu_{in}(T_n - T_i) + 2\nu_{ie} \left[(T_e - T_i) + \frac{m_e}{3k_B} (\mathbf{V}_{i0} - \mathbf{V}_{e0})^2 \right] + \frac{2}{3n} \nabla \cdot (\kappa_i \nabla T_i) \quad (3.87)$$

$$S_{T_e} = -\frac{m_e}{m_i} \nu_{ei} (T_e - T_i) + \frac{2}{3n} \nabla \cdot (\kappa_e \nabla T_e). \quad (3.88)$$

3.2 Perturbed Model

The model described in Section 3.1 can adequately solve many ionospheric turbulence problems. These problems, however, require the system to be in an equilibrium when no perturbation is applied. In fundamental physics simulations, this is traditionally done by assigning

initial conditions to all but one of the variables. The remaining variable is solved for such that the system is in an equilibrium. For the problems this model intends to solve, all of the variables need to be assigned. Therefore, the system is not necessarily initialized in an equilibrium.

If the time scales of the background dynamics are much larger than the time scales of the turbulence development, then the model can be modified such that the background is constant in time. In other words, if the background dynamics change much slower than the turbulence development, the background can effectively be considered constant in time. A perturbed model is developed that evolves only perturbed quantities and allows the background to remain constant in time. In nature, there are many factors that are involved in maintaining a quasi-equilibrium of the large scale background morphology that are not considered in this model, e.g., photoionization, recombination, or parallel motion. The perturbed model, by having the background variables constant in time, can account for these missing physics that maintain the quasi-equilibrium.

The perturbed model assumes that all of the primitive variables (ϕ , n , T_i , and T_e) can be split into background and perturbed quantities, e.g., $\phi = \phi_{bg} + \phi_p$. These new definitions can be substituted back into the governing equations.³ The result is a combination of terms that are individually functions of the background and perturbed variables. To ensure that the background does not evolve over time, any term containing only background quantities is removed. Therefore, the remaining terms evolve only the perturbed quantities. This allows for self-consistent turbulence generation without affecting the background dynamics.

Problems for which this model is applied have shown that the energy equations are negligible. Therefore, they have not been split using this perturbed model method. Furthermore, based on the initial conditions that are described in Appendices C.1 to C.4, the residual terms (any non-temporal derivative term on the left hand side of the governing equations, Eqs. 3.84 to 3.86) for the continuity and energy equations already satisfy the constraint of dropping terms of only the background. Therefore, only the temporal derivatives, continuity source term, and current closure equation are split using the perturbed model method. The continuity equation is trivial since the temporal derivative and the source term are linear. The result is

$$\frac{\partial n_p}{\partial t} - \frac{\nabla \phi \times \mathbf{B}}{B^2} \cdot \nabla n = D \nabla^2 n_p. \quad (3.89)$$

The current closure equation requires more work to get into the perturbed form. First, the pressures are separated into background and perturbed terms. The result for an arbitrary

³Note that the collision frequency terms, while still calculated self-consistently based on the density and temperatures, are not split in this way because they contain square root terms which cannot be easily separated into a summation of background and perturbed terms. Future work can consider using Taylor series expansions to approximate the background and perturbed collision frequency terms.

species' pressure is

$$\begin{aligned}
P_\alpha &= n_\alpha k_B T_\alpha \\
&= k_B (n_{\alpha_{bg}} + n_{\alpha_p}) (T_{\alpha_{bg}} + T_{\alpha_p}) \\
&= \underbrace{n_{\alpha_{bg}} k_B T_{\alpha_{bg}}}_{P_{\alpha_{bg}}} + \underbrace{k_B (n_{\alpha_{bg}} T_{\alpha_p} + n_{\alpha_p} T_{\alpha_{bg}} + n_{\alpha_p} T_{\alpha_p})}_{P_{\alpha_p}}.
\end{aligned} \tag{3.90}$$

Similarly, the $\mathbf{E} \times \mathbf{B}$ drift is

$$\begin{aligned}
\mathbf{V}_{\mathbf{E} \times \mathbf{B}} &= -\frac{\nabla \phi \times \mathbf{B}}{B^2} \\
&= -\underbrace{\frac{\nabla \phi_{bg} \times \mathbf{B}}{B^2}}_{\mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}}} - \underbrace{\frac{\nabla \phi_p \times \mathbf{B}}{B^2}}_{\mathbf{V}_{\mathbf{E} \times \mathbf{B}_p}}.
\end{aligned} \tag{3.91}$$

Each term in the current closure equation (Eqs. 3.82 and 3.83) is considered separately. The perturbed nonlinear temporal derivative, assuming $\partial \phi_{bg} / \partial t = 0$, is

$$\begin{aligned}
\nabla \cdot \left(n \nabla \frac{\partial \phi}{\partial t} \right)_p &= \nabla \cdot \left[n \nabla \left(\cancel{\frac{\partial \phi_{bg}}{\partial t}} + \frac{\partial \phi_p}{\partial t} \right) \right] \\
&= \nabla \cdot \left(n \nabla \frac{\partial \phi_p}{\partial t} \right).
\end{aligned} \tag{3.92}$$

The neutral wind term is linear and therefore is simply

$$\left(\mathbf{u} \times \mathbf{B} \cdot \nabla n \right)_p = \mathbf{u} \times \mathbf{B} \cdot \nabla n_p. \tag{3.93}$$

The perturbed form of $\nabla \cdot (n \nabla \phi)$ is

$$\begin{aligned}
\nabla \cdot (n \nabla \phi)_p &= \nabla \cdot \left[(n_{bg} + n_p) \nabla (\phi_{bg} + \phi_p) \right] \\
&= \nabla \cdot \left(\cancel{n_{bg} \nabla \phi_{bg}} + n_{bg} \nabla \phi_p + n_p \nabla \phi_{bg} + n_p \nabla \phi_p \right) \\
&= \nabla \cdot (n_p \nabla \phi + n_{bg} \nabla \phi_p).
\end{aligned} \tag{3.94}$$

The perturbed form of $\nabla \cdot (n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi)$ is

$$\begin{aligned}
\nabla \cdot (n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi)_p &= \nabla \cdot \left[(n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} + n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_p} + n_p \mathbf{V}_{\mathbf{E} \times \mathbf{B}}) \cdot \nabla \nabla \phi \right] \\
&= \nabla \cdot \left(\cancel{n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} \cdot \nabla \nabla \phi_{bg}} + n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} \cdot \nabla \nabla \phi_p + n_p \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right) \\
&= \nabla \cdot \left(n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} \cdot \nabla \nabla \phi_p + n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_p} \cdot \nabla \nabla \phi + n_p \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right).
\end{aligned} \tag{3.95}$$

Combining Eqs. 3.90 to 3.95, the full perturbed current closure equation is

$$\begin{aligned} \nabla \cdot \left(n \nabla \frac{\partial \phi_p}{\partial t} \right) &= \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left[-\frac{\nu_{in}}{e\Omega_{ci}} \nabla^2 P_{i_p} + \frac{\nu_{en}}{e\Omega_{ce}} \nabla^2 P_{e_p} \right. \\ &\quad \left. + \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(\mathbf{u} \times \mathbf{B} \cdot \nabla n_p - \nabla \cdot [n_p \nabla \phi + n_{bg} \nabla \phi_p] \right) \right] \\ &\quad - \nabla \cdot \left(n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} \cdot \nabla \nabla \phi_p + n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_p} \cdot \nabla \nabla \phi + n_p \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right). \end{aligned} \quad (3.96)$$

3.2.1 Governing Equations

The perturbed continuity and current closure equations, rewritten here for convenience, are

$$\frac{\partial n_p}{\partial t} - \frac{\nabla \phi \times \mathbf{B}}{B^2} \cdot \nabla n = D \nabla^2 n_p \quad (3.97)$$

$$\begin{aligned} \nabla \cdot \left(n \nabla \frac{\partial \phi_p}{\partial t} \right) &= \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left[-\frac{\nu_{in}}{e\Omega_{ci}} \nabla^2 P_{i_p} + \frac{\nu_{en}}{e\Omega_{ce}} \nabla^2 P_{e_p} \right. \\ &\quad \left. + \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(\mathbf{u} \times \mathbf{B} \cdot \nabla n_p - \nabla \cdot [n_p \nabla \phi + n_{bg} \nabla \phi_p] \right) \right] \\ &\quad - \nabla \cdot \left(n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} \cdot \nabla \nabla \phi_p + n_{bg} \mathbf{V}_{\mathbf{E} \times \mathbf{B}_p} \cdot \nabla \nabla \phi + n_p \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right). \end{aligned} \quad (3.98)$$

The energy equations, Eqs. 3.85 and 3.86, are still evolved in this system in the same way in their full forms. They are found to be negligible in affecting the plasma instabilities that are studied (the gradient drift and Kelvin-Helmholtz instabilities) for the F region ionosphere.

3.3 Local Linear Theory

3.3.1 Assumptions

The local linear theory of the governing equations is considered in this section to benchmark the mathematical model. The current closure and continuity equations, Eqs. 3.82 and 3.84 respectively, are linearized assuming that the density and electric potential are a background plus a plane wave perturbation of the form

$$n = n_0 + \underbrace{\tilde{n} \exp \left[i(\mathbf{k} \cdot \mathbf{r} - \omega t) \right]}_{n_1} \quad (3.99)$$

$$\phi = \phi_0 + \underbrace{\tilde{\phi} \exp \left[i(\mathbf{k} \cdot \mathbf{r} - \omega t) \right]}_{\phi_1}, \quad (3.100)$$

where \mathbf{k} is the wavenumber, \mathbf{r} is the position vector in Cartesian coordinates, ω is the wave frequency, and the subscripts correspond to the order of the terms. The utility of assuming a Fourier decomposition of this nature is that derivatives simply become algebraic expressions. For example, the time derivative and gradient of n_1 , respectively, are

$$\frac{\partial n_1}{\partial t} = -i\omega\tilde{n} \exp\left[i(\mathbf{k} \cdot \mathbf{r} - \omega t)\right] = -i\omega n_1 \quad (3.101)$$

$$\nabla n_1 = i\mathbf{k}\tilde{n} \exp\left[i(\mathbf{k} \cdot \mathbf{r} - \omega t)\right] = i\mathbf{k}n_1. \quad (3.102)$$

Therefore, Eqs. 3.82 and 3.84 are linearized by substituting Eqs. 3.99 and 3.100 for n and ϕ , respectively. Terms of only first order are considered.

The temperature equations are not linearized since they do not greatly impact the instabilities considered. While the collision frequencies are self-consistently calculated and are functions of the density and temperature, they are considered constant for the linearization. Considering otherwise would cause the derivation of the dispersion relation to become increasingly complicated. For further simplifications, the geometry is limited to a slab geometry such that

$$\mathbf{k} = k_x \hat{\mathbf{x}} + k_y \hat{\mathbf{y}} \quad (3.103)$$

$$\mathbf{u} = u_x \hat{\mathbf{x}} + u_y \hat{\mathbf{y}} \quad (3.104)$$

$$\mathbf{B} = B\hat{\mathbf{z}} \quad (3.105)$$

$$n = n_0(x) + n_1(x, y, t) \quad (3.106)$$

$$\phi = \phi_0(x) + \phi_1(x, y, t) \quad (3.107)$$

$$T_i = \text{const} \quad (3.108)$$

$$T_e = \text{const.} \quad (3.109)$$

The relevant governing equations, Eqs. 3.82 and 3.84, are linearized. The resulting set of linear equations can be written in the form

$$\mathbf{A}\mathbf{x} = \mathbf{0}, \quad (3.110)$$

where \mathbf{A} is a coefficient matrix that is a function of ω and \mathbf{x} is a vector of the first order variables. Therefore, the determinant of \mathbf{A} must be zero. Thus, the determinant of \mathbf{A} is a complicated function of ω and can be solved numerically. The real part of ω is the real frequency and the imaginary part, γ , is the growth rate.

The full derivation for the dispersion relation is found in Section A. The analytical solution for ω is found using Listing A.1, which is evaluated numerically using Listing A.2.

3.3.2 Benchmark Tests

The growth rate calculated using Listing A.2 is benchmarked to the GDI growth rate, Eq. 1.15. Note that this assumes a non-inertial case and therefore $I = 0$. I is defined

in Section A.1 as a switch that turns the inertial terms on or off. In the F region, the collisional effects are assumed to be negligible and the growth rate is considered in the short wavelength limit. Therefore, the simplified growth rate, accounting for the numerical diffusion, is

$$\gamma = \frac{u}{L_N} \left[\hat{\mathbf{k}} \cdot (\hat{\mathbf{b}} \times \hat{\mathbf{g}}) \right] (\hat{\mathbf{k}} \cdot \hat{\mathbf{e}}) - Dk^2. \quad (3.111)$$

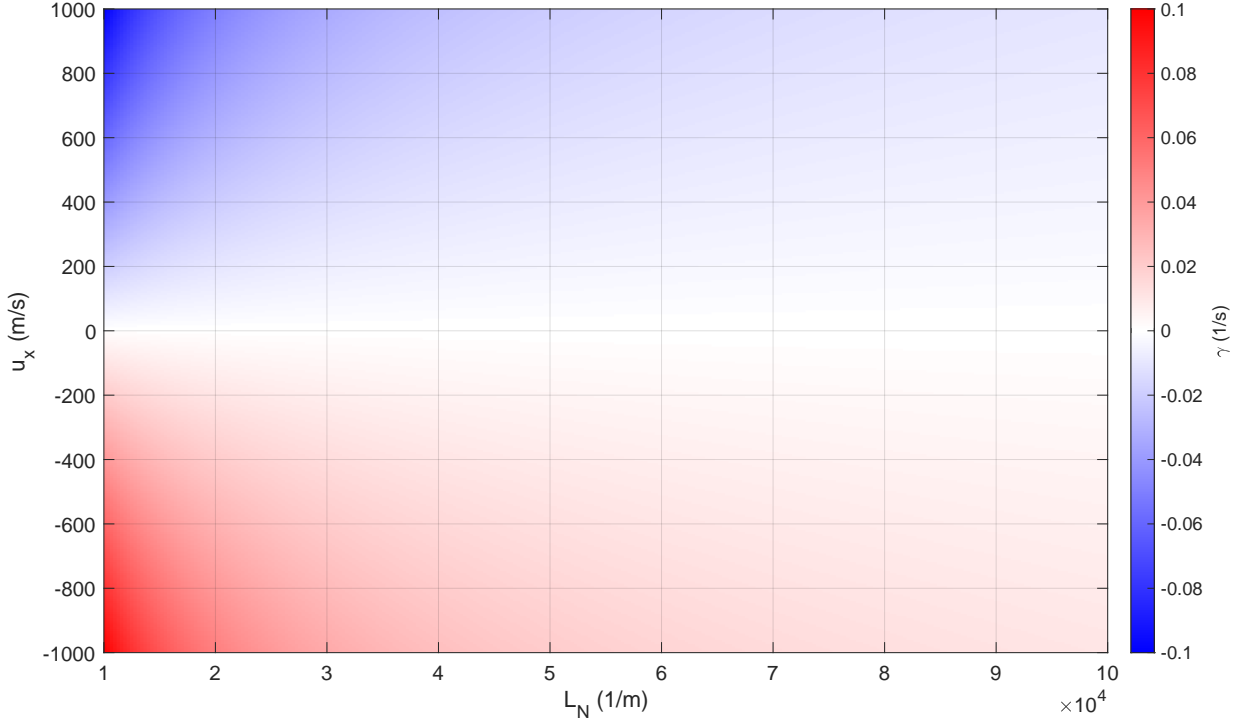


Figure 3.1: A plot of the growth rate as a function of the neutral wind in the $\hat{\mathbf{x}}$ direction, u_x , and the density gradient scale length, L_N , in the $\hat{\mathbf{x}}$ direction. For positive u_x , the GDI is damped. For negative u_x , the GDI is unstable. As the magnitude of the neutral wind increases, so does the magnitude of the growth rate. As the magnitude of the density gradient scale length increases, the magnitude of the growth rate decreases. These results are consistent with Eq. 3.111.

All of the electric field is assumed to come from the neutral wind, which implies that the plasma's $\mathbf{E} \times \mathbf{B}$ drift is the neutral wind speed. The magnetic field unit vector is $\hat{\mathbf{b}} = \hat{\mathbf{z}}$. The electric field unit vector is $\hat{\mathbf{e}} = \hat{\mathbf{y}}$. The density gradient unit vector is $\hat{\mathbf{g}} = \hat{\mathbf{x}}$. The wavenumber unit vector is $\hat{\mathbf{k}} = \hat{\mathbf{y}}$. The numerical diffusion constant is set to 0. Figure 3.1 shows the resulting numerically derived growth rate. The neutral wind is varied from -1000 m/s to 1000 m/s. The density gradient scale length is varied from 10^4 m to 10^5 m. The result matches Eq. 3.111. For positive u_x , the GDI is damped and increasing the magnitude stabilizes the instability further. For negative u_x , the GDI is unstable and grows faster with

higher magnitude. For both cases, increasing the density gradient scale length decreases the magnitude of the growth rate.

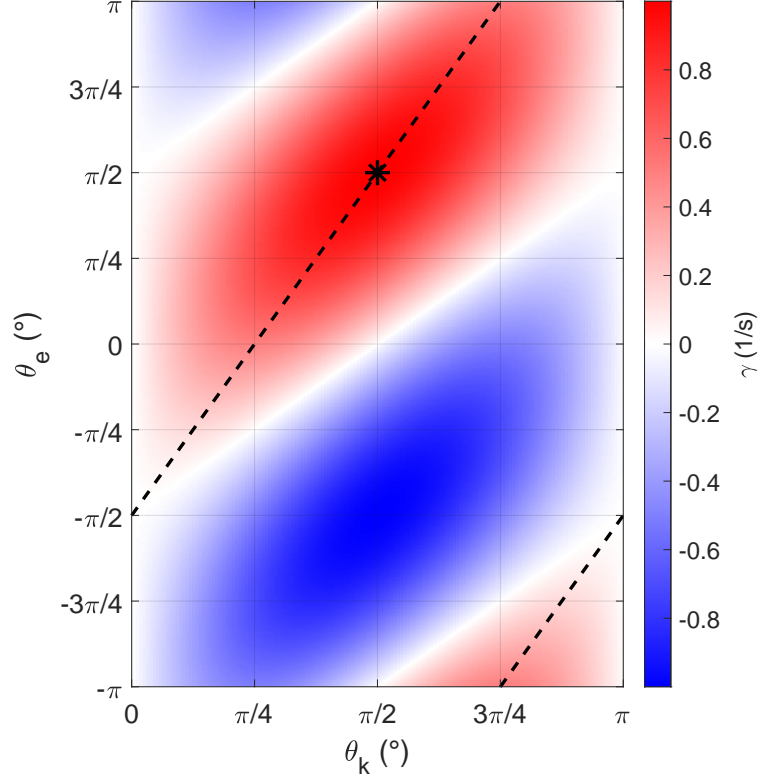


Figure 3.2: A plot of the growth rate as a function of θ_e and θ_k . The dashed black line represents the maxima for each θ_e . The asterisk is the global maximum at $\theta_e = \theta_k = \pi/2$. These results are consistent with Eq. 3.111.

To test the arbitrary geometry of Eq. 3.111, the unit vectors for the electric field and wavenumber are, respectively, defined as $\hat{\mathbf{e}} = \cos\theta_e\hat{\mathbf{x}} + \sin\theta_e\hat{\mathbf{y}}$ and $\hat{\mathbf{k}} = \cos\theta_k\hat{\mathbf{x}} + \sin\theta_k\hat{\mathbf{y}}$. The density gradient unit vector is $\hat{\mathbf{g}} = \hat{\mathbf{x}}$. The magnetic field unit vector is $\hat{\mathbf{b}} = \hat{\mathbf{z}}$. The magnitude of the $\mathbf{E} \times \mathbf{B}$ drift is 10^4 m/s and the magnitude of the density gradient scale length is 10^4 m. The numerical diffusion constant is set to 0. Figure 3.2 shows the resulting numerically derived growth rate, which is consistent with Eq. 3.111. The dashed black line indicates the maxima for each θ_e , which is found using

$$\theta_e = 2\theta_k - \frac{\pi}{2} + 2l\pi \quad l = 0, \pm 1, \pm 2, \dots \quad (3.112)$$

The derivation for Eq. 3.112 is shown in greater detail in Section 5.2 with a more general form described in terms of θ_k found in Eq. 5.16. The total period maximum occurs at $\theta_e = \theta_k = \pi/2$, marked by the asterisk. Note that only the results for θ_k from 0 to π are shown since the effect of the wavenumber direction on the dispersion relation must be symmetric with respect to its sign.

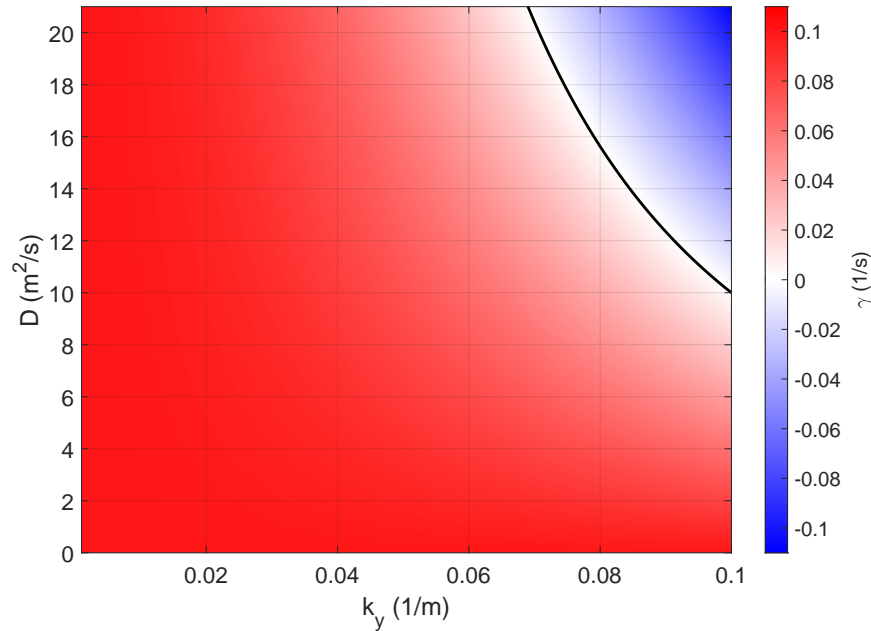


Figure 3.3: A plot of the growth rate as a function of the numerical diffusion constant, D , and the wavenumber. The configuration is the same as that of Figure 3.1 except that the neutral wind is -1000 m/s and the density gradient scale length is 10^4 m. The black line is the calculated contour for $\gamma = 0$ based on Eq. 3.111.

The effect of numerical diffusion acts to damp higher wavenumber modes. To test this, the same parameters are used as in Figure 3.1, with the exception that the neutral wind and density gradient scale length are a constant -1000 m/s and 10^4 m, respectively. The numerical diffusion is varied from 0 m²/s to 21 m²/s and the wavenumber is varied from 0.001 m⁻¹ to 0.1 m⁻¹. Figure 3.3 shows the results, with the black line representing the contour of $\gamma = 0$ calculated using Eq. 3.111.

3.4 Numerical Methods

This section discusses the different numerical methods used to solve the governing equations: Eqs. 3.97, 3.98, 3.85, and 3.86. First, the general discretization method is discussed. Then, an iterative method for Eq. 3.98 is explained. Lastly, the time integration scheme is described.

3.4.1 Pseudo-Spectral or Collocation Method

There are many methods that have been developed to spatially discretize partial differential equations (PDEs), e.g. finite difference methods, finite volume methods, finite element

methods, discontinuous Galerkin methods, and spectral methods. This code uses the spectral method approach [Canuto et al., 2012].

Spectral methods write the global solution, u , of an arbitrary PDE as

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k \phi_k, \quad (3.113)$$

which is an infinite sum of orthogonal basis functions, ϕ_k , multiplied by expansion coefficients, \hat{u}_k . The choice of the basis functions determines the inherent assumptions in the solution, methods of solving the numerical system, computational efficiency, and accuracy. Examples of basis functions include Fourier series, Chebychev polynomials, or Legendre polynomials. This code uses the Fourier series as its basis functions and therefore naturally has periodic boundary conditions. Any periodic function, $u(x)$, can be written as a summation of complex exponentials as

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx}. \quad (3.114)$$

The expansion coefficients, $\hat{u}_k(k)$, are found using

$$\hat{u}_k(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} u(x) e^{-ikx} dx = \mathcal{F}[u(x)], \quad (3.115)$$

which is called the Fourier transform and denoted by the operator \mathcal{F} . This converts the function from the time domain to the frequency domain. The inverse Fourier transform,

$$u(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{u}_k(k) e^{ikx} dx = \mathcal{F}^{-1}[\hat{u}_k(k)], \quad (3.116)$$

is denoted by the operator \mathcal{F}^{-1} . For simplicity, Eqs. 3.115 and 3.116 are written in 1D. The 2D extensions are shown in Eqs. 3.121 and 3.122 later in this section.

Realistically, Eq. 3.114 cannot be evaluated to $\pm\infty$ and needs to be stopped at some arbitrary integer value. Moreover, computationally, the solution u is defined on a discrete spatial grid as u_i , with points indexed by i . If the 1D domain is divided into N grid points with constant spacing, Δx , then the largest frequency that can be sampled by the domain, also called the Nyquist frequency, is

$$k_{\max} = \frac{2\pi}{2\Delta x} = \frac{\pi}{\Delta x}, \quad (3.117)$$

This is because at least two points of data are required to discern a periodic wave. The factor of 2π converts from linear to angular spatial frequency. Similarly, the smallest non-steady state frequency that can be obtained in this domain is

$$k_{\min} = \frac{2\pi}{N\Delta x}. \quad (3.118)$$

This is because wavelengths larger than the size of the domain, $L = N\Delta x$, cannot be captured. Therefore, the discrete version of Eq. 3.114, which is also the discrete inverse Fourier transform, is

$$U_n = \sum_{j=-N/2}^{N/2-1} \hat{u}_k e^{ikx_n}, \quad (3.119)$$

where U is the numerical approximation to the exact solution u , the subscript n corresponds to a value at a discrete grid point, and the wavenumber is

$$k_j = \frac{2\pi j}{N\Delta x}. \quad (3.120)$$

Note that from a practical perspective, the FFT function in MATLAB internally sets the wavenumber as shown in Listing 3.1, which uses MATLAB's notation.

Listing 3.1: Definition of the wavenumber vector in MATLAB.

```
1 k = (2 * pi / L) * [0:n/2-1 , -n/2:-1]';
```

The mathematical model described in Sections 3.1 and 3.2 assumes the physical domain is 2D. Therefore, the discrete Fourier and inverse Fourier transforms in 2D are

$$\hat{u}_{j_x, j_y} = \frac{1}{N_x} \frac{1}{N_y} \sum_{n_x=0}^{N_x-1} \sum_{n_y=0}^{N_y-1} U_{j_x, j_y} \exp(-ik_x x_{n_x}) \exp(-ik_y y_{n_y}) \quad (3.121)$$

$$U_{n_x, n_y} = \sum_{j_x=-N_x/2}^{N_x/2-1} \sum_{j_y=-N_y/2}^{N_y/2-1} \hat{u}_{j_x, j_y} \exp(ik_x x_{n_x}) \exp(ik_y y_{n_y}). \quad (3.122)$$

Note that depending on the convention used for the discrete Fourier transform, the normalization factor, $N_x N_y$, can be divided in either the forward or inverse transform (but not both). Certain numerical packages already apply the normalization and others do not. Therefore, it is of importance to understand the numerical package that is used in case the normalization needs to be externally applied by the user. For example, the normalization is internally included in MATLAB, but needs to be implemented by the user if using FFTW3 in C.

The pseudo-spectral or collocation method [Orszag, 1971] considers the evolution of the approximate solution U at discrete locations on a Cartesian grid, which in this case is 2D. These values are used in Eq. 3.121 to obtain the expansion coefficients \hat{u}_{n_x, n_y} . The utility of this method arises when taking derivatives. The gradient operator in Fourier space becomes

$$\mathcal{F}[\nabla] = i\mathbf{k}. \quad (3.123)$$

Similarly, the ∇^2 operator becomes

$$\mathcal{F}[\nabla^2] = -k^2. \quad (3.124)$$

Thus, differentiation in real space is just an algebraic expression in Fourier space. Because of this property, dealing with highly nonlinear terms is much simpler using the collocation method compared to other discretization schemes.

There are situations in which the exact definition of k from Eq. 3.120 results in an unstable numerical scheme and causes an accumulation of numerical error. This can be resolved by using a finite difference approximation of a wavenumber space that is itself forced to be periodic. The resulting approximations for k and k^2 , respectively, are

$$\mathbf{K} = \frac{\sin(k_x \Delta x)}{\Delta x} \hat{\mathbf{k}}_x + \frac{\sin(k_y \Delta y)}{\Delta y} \hat{\mathbf{k}}_y \quad (3.125)$$

$$\hat{K}^2 = \left[\frac{\sin\left(\frac{\Delta x}{2}\right)}{\frac{k_x \Delta x}{2}} \right]^2 + \left[\frac{\sin\left(\frac{k_y \Delta y}{2}\right)}{\frac{\Delta y}{2}} \right]^2. \quad (3.126)$$

As $\mathbf{k}\Delta x$ approaches 0, \mathbf{K} and \hat{K}^2 approach \mathbf{k} and k^2 , respectively [Birdsall and Langdon, 2005].

Now that differentiation can be performed in Fourier space, a closer look is taken as to how multiplication is handled. Multiplication in physical space is equivalent to a convolution in Fourier space. Numerically, taking a multi-dimensional complex convolution is non-trivial. Therefore, the simplified approach to the convolution is to take the inverse Fourier transform of the two Fourier transformed variables, which converts them back to physical space. In physical space, they are multiplied normally. The product is then converted back to Fourier space. Thus, for two arbitrary 1D functions, f and g , whose representations in Fourier space are F and G respectively, the convolution is

$$S = F * G = \mathcal{F}\left(\mathcal{F}^{-1}[F]\mathcal{F}^{-1}[G]\right) = \mathcal{F}[fg], \quad (3.127)$$

where the $*$ operator denotes the convolution and S is the convolved result in the Fourier domain. From a numerical perspective, the result is not an exact solution and aliasing errors can arise. Consider the discrete approximation of f and g by the inverse Fourier transforms of F and G ,

$$f_n = \sum_{j=-N/2}^{N/2-1} \hat{f}_k \exp(ikx_n) \quad (3.128)$$

$$g_n = \sum_{j=-N/2}^{N/2-1} \hat{g}_k \exp(ikx_n). \quad (3.129)$$

The discrete multiplication in real space is

$$s_n = f_n g_n, \quad (3.130)$$

where s is the exact product of f and g in real space and s_n is the approximation of the product. Substituting Eqs. 3.128 and 3.129 into Eq. 3.130 and using the fact that Fourier series are an orthogonal basis yields

$$\tilde{s}_j = \sum_{m+l=j} \hat{f}_m \hat{g}_l + \sum_{m+l=j \pm N} \hat{f}_m \hat{g}_l = \hat{s}_j + \underbrace{\sum_{m+l=j \pm N} \hat{f}_m \hat{g}_l}_{\varepsilon_{AE}}, \quad (3.131)$$

where $\hat{s}_j = \mathcal{F}[s]$ is the Fourier transform of the exact product, \tilde{s}_j is the approximation of s in Fourier space after using the convolution in Eq. 3.130, the subscripts m and l represent the indices for the Fourier series for f and g respectively, and ε_{AE} is the aliasing error. The error is from the additional summation terms in the convolution that do not exist in the Fourier representation of s .

Two possible methods of de-aliasing, i.e., mitigating the aliasing error, are the zero padding and the grid shifting methods. Only the zero padding method is considered in this code. The zero padding method [Patterson Jr and Orszag, 1971] considers a Fourier transformed variable padded with zeros of the form

$$\check{f}_j = \begin{cases} \hat{f}_j & |j| \leq N/2 \\ 0 & \text{otherwise} \end{cases}, \quad (3.132)$$

where \check{f} is the Fourier transform of f on a grid with M points instead of N points. The same can be done for a variable g . Then, similar to Eq. 3.131, the padded M point Fourier transform of s is

$$\check{s}_j = \sum_{m+l=j} \check{f}_m \check{g}_l + \sum_{m+l=j \pm M} \check{f}_m \check{g}_l. \quad (3.133)$$

For an N point Fourier transform, only \check{s} for $|j| \leq N/2$ are needed. All \check{f}_m and \check{g}_l for $|m|, |l| > N/2$ are 0. Therefore, the number of points, M , needed for the aliasing error term to disappear is based on the summation criterion for the error term. The worst case occurs when m and l are minimized, j is maximized, and M is subtracted, which results in

$$-\frac{N}{2} - \frac{N}{2} = \frac{N}{2} - 1 - M. \quad (3.134)$$

Therefore, the minimum number of points needed to pad the Fourier transformed variables to eliminate the aliasing error term is

$$M \geq \frac{3}{2}N. \quad (3.135)$$

The concept of the convolution can be simply extended to two dimensions by using the 2D forward and inverse Fourier transforms, Eqs. 3.121 and 3.122. Similarly, the zero padding de-aliasing technique is extended to two dimensions by padding the second dimension with $M_y \geq 3N_y/2$.

The effect of de-aliasing removes the aliasing error at the cost of increasing computation time. The magnitude of the aliasing error is dependent upon the number of grid points. The aliasing error drops asymptotically as N increases. Therefore, once sufficient resolution is achieved, the effect of the aliasing error is minimal. The issue arises when the resolution required for the simulated physical phenomenon is too large to be solved computationally [Birdsall and Langdon, 2005]. For example, this can occur during turbulence simulations as the scales needed to resolve the physics become smaller. Since the need for de-aliasing is problem dependent, the user has the option in the convolution function to choose whether to use no de-aliasing or zero padding de-aliasing.

Through the use of the Fourier transform to convert derivatives into algebraic expressions and the use of the convolution to handle multiplications in the governing equations, the pseudo-spectral method turns the set of PDEs into a set of ordinary differential equations (ODEs) in time. The continuity and energy equations become linear ODEs in time and the current closure equation is a nonlinear temporal ODE.

3.4.2 Iterative Method

The current closure equation, whether the full equation (Eq. 3.82) or the perturbed version (Eq. 3.98), is a nonlinear PDE of the form

$$\nabla \cdot \left(n \nabla \frac{\partial \phi}{\partial t} \right) = S_\phi. \quad (3.136)$$

For evolving Eq. 3.82, the full ϕ is used in the temporal derivative, and for the perturbed equation (Eq. 3.98), ϕ_p is used in the temporal derivative. The corresponding source term, S_ϕ , is chosen accordingly. The numerical method is agnostic to the choice of equation. The key difference between the current closure equation and the other governing equations is that the temporal derivative term in Eq. 3.136 is nonlinear. The term $\partial\phi/\partial t$ cannot be isolated. Therefore, an iterative method is used to find a numerical approximation of the temporal derivative. The goal is to get Eq. 3.136 to be of the form

$$x = f(x), \quad (3.137)$$

where $f(x)$ is some function of some variable x . The vector calculus identity for the divergence of a scalar times a gradient, for arbitrary scalars a and b , is

$$\nabla \cdot (a \nabla b) = a \nabla^2 b + \nabla a \cdot \nabla b. \quad (3.138)$$

Eq. 3.136 is expanded using Eq. 3.138, resulting in

$$n \nabla^2 \frac{\partial \phi}{\partial t} + \nabla n \cdot \nabla \frac{\partial \phi}{\partial t} = S_\phi. \quad (3.139)$$

Solving for $\nabla^2 \partial\phi/\partial t$ yields

$$\nabla^2 \frac{\partial\phi}{\partial t} = \frac{1}{n} \left(S_\phi - \nabla n \cdot \nabla \frac{\partial\phi}{\partial t} \right). \quad (3.140)$$

Taking the Fourier transform of both sides and using Eq. 3.124 for the Laplacian yields

$$-k^2 \frac{\partial\hat{\phi}}{\partial t} = \mathcal{F} \left[\frac{1}{n} \left(S_\phi - \nabla n \cdot \nabla \frac{\partial\phi}{\partial t} \right) \right]. \quad (3.141)$$

Solving the left hand side for $\partial\hat{\phi}/\partial t$ yields

$$\frac{\partial\hat{\phi}}{\partial t} = -\frac{1}{k^2} \mathcal{F} \left[\frac{1}{n} \left(S_\phi - \nabla n \cdot \nabla \frac{\partial\phi}{\partial t} \right) \right] = f \left(\frac{\partial\hat{\phi}}{\partial t} \right). \quad (3.142)$$

Therefore, Eq. 3.142 is of the form in Eq. 3.137. This equation can be solved for iteratively using a previously estimated value of $\partial\hat{\phi}/\partial t$ in the function to calculate the next estimate, i.e.,

$$\left. \frac{\partial\hat{\phi}}{\partial t} \right|_i = f \left(\left. \frac{\partial\hat{\phi}}{\partial t} \right|_{i-1} \right). \quad (3.143)$$

Note that $1/k^2$ spans the entire Fourier domain, which includes the steady state values. The steady state modes have $k_x = k_y = 0$, which results in $k^2 = 0$ and thus becomes undefined for $1/k^2$. Based on the governing equations, only the derivative of the electric potential plays any role in affecting the dynamics. Therefore, the effect of any steady state term, i.e., an added constant, disappears. To resolve the issue of dividing by 0, the wavenumber for the 0 mode, just for this single calculation, is defined as 1 such that $1/k^2 = 1$ since the 0 mode, solely for the electric potential, is never actually used in the model.

The initial guess is required to be nonzero. In this case, it is arbitrarily chosen to be 1. If the conditions on f are good, then Eq. 3.143 converges. The method is considered converged if the difference of the maximum values between iterations, normalized by the present iteration, is within some tolerance,

$$\left| \frac{\left| \frac{\partial\hat{\phi}}{\partial t} \right|_i^{\max} - \left| \frac{\partial\hat{\phi}}{\partial t} \right|_{i-1}^{\max}}{\left| \frac{\partial\hat{\phi}}{\partial t} \right|_i^{\max}} \right| \leq TOL, \quad (3.144)$$

where TOL is a user defined tolerance. Because these variables exist in two dimensions and are thus 2D arrays, choosing the maximum value in the criterion provides a scalar quantity to evaluate. If Eq. 3.144 is met, then the method is considered to have converged and the loop is broken. A maximum number of iterations are also set to break the loop if the conditions for f are not good and result in the method diverging. Since this is a highly nonlinear and complicated system, it is not entirely obvious what causes the iterative method to diverge. Generally, hot spots appear in the plasma density, which either cause discontinuities that the

iterative method cannot handle or negative densities that are unphysical and cause the system of equations to break down. Three methods of dealing with this issue are to either increase the numerical diffusion, D , increase the resolution, or apply the zero padding de-aliasing algorithm to the convolutions. The easiest solution is typically to increase D since it does not cause a significant increase in computational time compared to the other two methods. However, high numerical diffusion can lead to unphysical results. The zero padding de-aliasing method often plays only a minimal role despite the increase in computational time. Increasing the resolution by increasing the number of grid points⁴ is the ideal solution in terms of numerical and physical accuracy; however, it comes at the cost of greatly increased computational effort. The best combination of all three is problem dependent.

3.4.3 Runge-Kutta Method

Obtaining the temporal derivative of the electric potential converts the set of governing equations into a system of 4 linear ODEs. There are many time integration schemes that can be used to evolve the variables in time. The method used in this computational model is a multi-stage Runge-Kutta (RK) method. The RK method is useful for solving ODEs of the form

$$\frac{\partial \phi}{\partial t} = F(\phi, t), \quad (3.145)$$

where ϕ is an arbitrary dependent variable, t is an arbitrary independent variable (in this case, time), and F is an arbitrary function of ϕ and t . If this problem is set up as an initial value problem such that the value of ϕ_n at t_n is known, then the general form for an explicit s -stage RK method is

$$\xi_1 = \phi_n \quad (3.146)$$

$$\xi_2 = \phi_n + \Delta t a_{2,1} F(\xi_1, t_n) \quad (3.147)$$

$$\xi_3 = \phi_n + \Delta t \left[a_{3,1} F(\xi_1, t_n) + a_{3,2} F(\xi_2, t_n + c_3 \Delta t) \right] \quad (3.148)$$

⋮

$$\xi_s = \phi_n + \Delta t \sum_{j=1}^{s-1} a_{s,j} F(\xi_j, t_n + c_j \Delta t) \quad (3.149)$$

$$\phi_{n+1} = \phi_n + \Delta t \sum_{j=1}^s b_j F(\xi_j, t_n + c_j \Delta t), \quad (3.150)$$

where ξ is an intermediate weighted slope, and the coefficients a , b , and c are derived based on the number of stages and order of accuracy desired [Durrant, 2010, p. 49-50]. The traditional

⁴It is possible to decrease the domain length in order to increase the resolution. This is not always practical depending on the problem. Furthermore, the effect of this change may be minimal compared to adjusting the resolution.

Euler method taught in differential equations classes is a 1-stage RK method with $b_1 = 1$ and $c_1 = 0$. A common RK method is the 4-stage fourth order method,

$$\xi_1 = \phi_n \tag{3.151}$$

$$\xi_2 = \phi_n + \frac{\Delta t}{2} F(\xi_1, t_n) \tag{3.152}$$

$$\xi_3 = \phi_n + \frac{\Delta t}{2} F(\xi_2, t_n + \frac{\Delta t}{2}) \tag{3.153}$$

$$\xi_4 = \phi_n + \Delta t F(\xi_3, t_n + \frac{\Delta t}{2}) \tag{3.154}$$

$$\phi_{n+1} = \phi_n + \frac{\Delta t}{6} \left[F(\xi_1, t_n) + 2F(\xi_2, t_n + \frac{\Delta t}{2}) + 2F(\xi_3, t_n + \frac{\Delta t}{2}) + F(\xi_4, t_n + \Delta t) \right]. \tag{3.155}$$

One issue with this method is that it requires additional storage of 4 sets of data (ξ_1 to ξ_4). For large simulations, this can become a limitation. RK methods can be developed such that only the most recent approximation is needed. The method used in this code for the time integration method assumes that the function F is only a function of ϕ and not time. The resulting low storage 4-stage fourth order RK method [Hirsch, 1990, p. 334] is

$$\xi_1 = \phi_n \tag{3.156}$$

$$\xi_2 = \phi_n + \alpha_1 \Delta t F(\xi_1) \tag{3.157}$$

$$\xi_3 = \phi_n + \alpha_2 \Delta t F(\xi_2) \tag{3.158}$$

$$\xi_4 = \phi_n + \alpha_3 \Delta t F(\xi_3) \tag{3.159}$$

$$\phi_{n+1} = \phi_n + \alpha_4 \Delta t F(\xi_4), \tag{3.160}$$

where

$$\alpha_1 = \frac{1}{4}, \quad \alpha_2 = \frac{1}{3}, \quad \alpha_3 = \frac{1}{2}, \quad \alpha_4 = 1. \tag{3.161}$$

3.5 Code Verification

The codes written for use in computational physics are complicated and it is not always obvious by inspection if any mistakes have been made in the coding process. Code verification is the process of ensuring that the coded numerical algorithms behave as expected and provide reasonable approximations to the physics problems being solved. This section provides several tests used to verify the code. The errors not considered in the code verification are aliasing error (which was discussed in greater detail in Section 3.4.1) and roundoff error.

3.5.1 Order of Accuracy

Consider an arbitrary PDE model,

$$L(u) = 0, \tag{3.162}$$

where L is the PDE operator and u is the variable to be solved for, whose exact solution is \tilde{u} . To solve this numerically, L needs to be spatially and temporally discretized. This results in the discretized equation

$$L_h(u) = L(u) + TE_h(u), \quad (3.163)$$

where L_h is the discretized PDE operator and TE_h is the truncation error. The act of discretizing L inherently means that the derivatives are approximated. The resulting error between the discretized and the exact operators is the truncation error. If the truncation error approaches 0 as the discretization parameter, e.g., Δx or Δt , approaches 0, the numerical scheme is considered consistent [Oberkampf and Roy, 2010]. The discretized PDE is solved exactly u_h , the discretized solution, as opposed to u (the exact solution to the operator L).

The difference between the exact and discretized solution is

$$\varepsilon_{DE} = u_h - \tilde{u}, \quad (3.164)$$

where ε_{DE} is the discretization error. One goal of computational methods is to reduce the discretization error as much as possible. The rate at which u_h approaches \tilde{u} as the mesh is refined is called the order of accuracy. Different numerical schemes have different theoretical or formal orders of accuracy, which is how solutions should behave. The observed order of accuracy is how the solutions actually behave. The observed order of accuracy can be calculated by solving L_h . The solution is then compared to known exact solutions on multiple grids that are systematically refined such that the grid with the fine mesh spacing, h_1 , and coarse mesh spacing, h_2 , have the constant refinement factor

$$r = \frac{h_2}{h_1}. \quad (3.165)$$

Therefore, if h_1 is defined as h , then $h_2 = rh$. The discretization error is calculated using the known exact solution, \tilde{u} and the approximate solution, u_h . Issue arises when comparing the discretization errors between the grids. Firstly, the discretization error is a function of space and time; it is not a singular scalar quantity. Secondly, due to the different meshes, the discretization errors occupy more grid points in the fine mesh. To resolve issues with mapping, the discretization error is converted to a global scalar value by taking the L^p norm. The L^p norm is defined as

$$L^p(x) = \|x\|_p = \left(\frac{1}{N} \sum_{i=1}^N |x_i|^p \right)^{1/p}, \quad (3.166)$$

where x is some vector indexed from 1 to N . Note that the infinity norm is

$$L^\infty(x) = \|x\|_\infty = \max(|x|). \quad (3.167)$$

The now scalar norms of the discretization errors can be compared between the different meshes. The resulting observed order of accuracy, \hat{p} , is

$$\hat{p} = \frac{\ln \left(\frac{\|\varepsilon_{rh}\|}{\|\varepsilon_h\|} \right)}{\ln(r)}. \quad (3.168)$$

The observed order of accuracy is a strong test for verifying the code due to its sensitivity to many smaller errors in code development [Oberkampf and Roy, 2010].

The difficulty in calculating the observed order of accuracy lies in having an exact solution to the PDE. If many exact solutions were known, then the equations would not have to be solved using numerical methods. If an exact solution to the set of PDEs is not known, then the method of manufactured solutions can be used [Oberkampf and Roy, 2010]. If the operator L has no known exact solutions (or the solutions are difficult to implement properly), then a solution, \hat{u} , can be chosen. This solution is purely a mathematical construct and does not necessarily represent any physical phenomenon. Since the solution \hat{u} is chosen arbitrarily, it is not expected to be an actual solution to L . However, operating L on \hat{u} ,

$$L(\hat{u}) = S, \quad (3.169)$$

yields an analytical source term, S . Therefore, \hat{u} is the exact solution for the equation

$$L(u) = S. \quad (3.170)$$

The observed order of accuracy can then be calculated based on this new PDE model, for which the exact solution is the chosen solution \hat{u} .

3.5.2 Testing Spatial Derivatives

The first test for the code verification is to ensure that the derivatives follow expected behavior when discretized. This is tested using the functions F_1 through F_7 , which are

$$F_1 = a_x \sin\left(\frac{2\pi n_{x1}x}{L_x}\right) + a_y \cos\left(\frac{2\pi n_{y1}y}{L_y}\right) + a_{xy} \cos\left(\frac{2\pi n_{xy1}x}{L_x}\right) \sin\left(\frac{2\pi n_{xy2}y}{L_y}\right) \quad (3.171)$$

$$F_2 = b_x \left[\tanh\left(\frac{x - x_{a1}}{c_x}\right) - \tanh\left(\frac{x - x_{a2}}{c_x}\right) \right] + b_y \left[\tanh\left(\frac{y - y_{a1}}{c_y}\right) - \tanh\left(\frac{y - y_{a2}}{c_y}\right) \right] \quad (3.172)$$

$$F_3 = \sum_{i=1}^4 d_i \left[e_x \ln\left(\cosh\left[\frac{x - x_{b_i}}{g_x}\right]\right) + e_y \ln\left(\cosh\left[\frac{y - y_{b_i}}{g_y}\right]\right) \right] \quad (3.173)$$

$$F_4 = l_x \exp\left[-\frac{(x - x_c)^2}{h_x}\right] + l_y \exp\left[-\frac{(y - y_c)^2}{h_y}\right] \quad (3.174)$$

$$F_5 = m_x \exp\left[-\frac{(x - x_d)^2}{o_x} - \frac{(y - y_d)^2}{o_y}\right] \quad (3.175)$$

$$F_6 = p_x \exp\left[-\frac{(x - x_e)^2}{q_x}\right] \sin\left(\frac{2\pi n_{y2}y}{L_y}\right) + p_y \exp\left[-\frac{(y - y_e)^2}{q_y}\right] \cos\left(\frac{2\pi n_{x2}x}{L_x}\right) \quad (3.176)$$

$$F_7 = \sum_{i=1}^6 F_i. \quad (3.177)$$

Note that all of these are functions of both x and y in order to test the derivatives in both directions as well as the Laplacian. These functions are considered because they are typically used to set up different initial conditions to solve ionospheric turbulence problems. Eq. 3.171 is an example of a sinusoidal perturbation used to initialize the growth of some plasma instability. Eq. 3.172 is typically used as a background density or velocity profile to set up a gradient. Eq. 3.173 is the integral of Eq. 3.172, which is used to set up the electric potential profile (further explained in Appendix C.3). Eq. 3.174 is a set of Gaussian functions which are useful for understanding 1D plasma clouds. Eq. 3.175 is a Gaussian blob that is useful for studying 2D plasma clouds. Eq. 3.176 is a Gaussian multiplied by a sinusoid, which is useful for applying a sinusoidal perturbation in a specific region of the domain. Lastly, Eq. 3.177 is a summation of Eqs. 3.171 through 3.176. The full set of exact solutions of the derivatives and Laplacians, as well as the constants used in this calculation, are found in Appendix B.1.

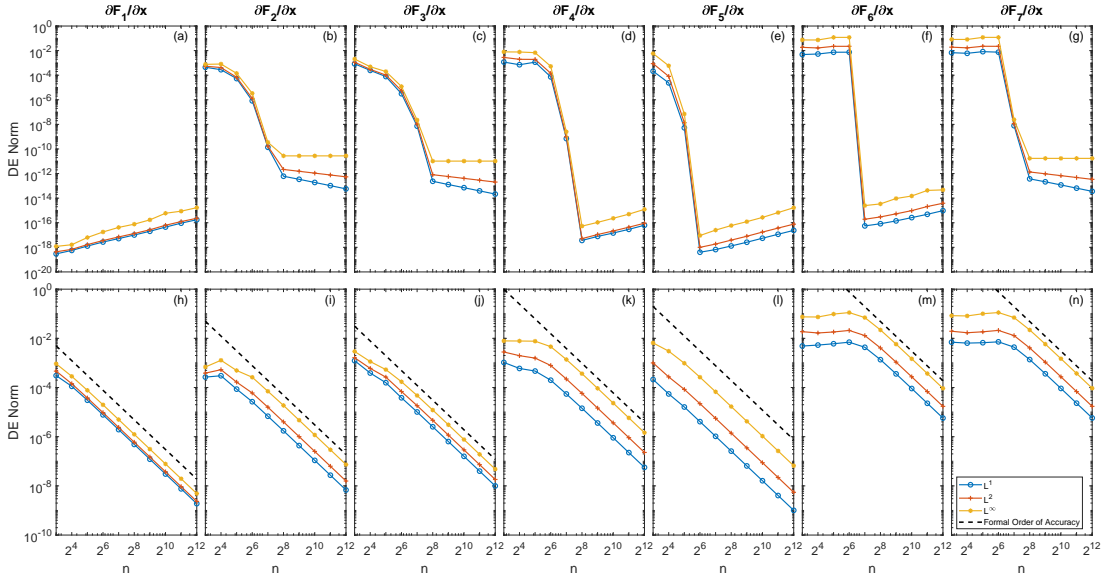


Figure 3.4: Plots of the norm of the discretization error of the x direction derivatives versus the number of grid points, n . The top panels use the exact definition of k_x from Eq. 3.120, and the bottom panels use the approximation, K_x , from Eq. 3.125. The top panels show that the discretization error decreases significantly, typically to machine precision, once the Nyquist criterion is satisfied. Note that for Panel (a), the Nyquist criterion is already satisfied for the coarsest mesh. The bottom panels show that the discretization error falls at the expected order of accuracy of 2.

Figures 3.4, 3.5, and 3.6 show the discretization error versus the number of grid points, n , for the derivatives with respect to x , the derivatives with respect to y , and the Laplacians, respectively. The meshes are systematically refined by a factor of 2 with the coarsest mesh containing 8×8 grid points and the finest mesh containing 4096×4096 grid points. For each figure, the top panels use the exact definition of k and k^2 from Eq. 3.120. All of the

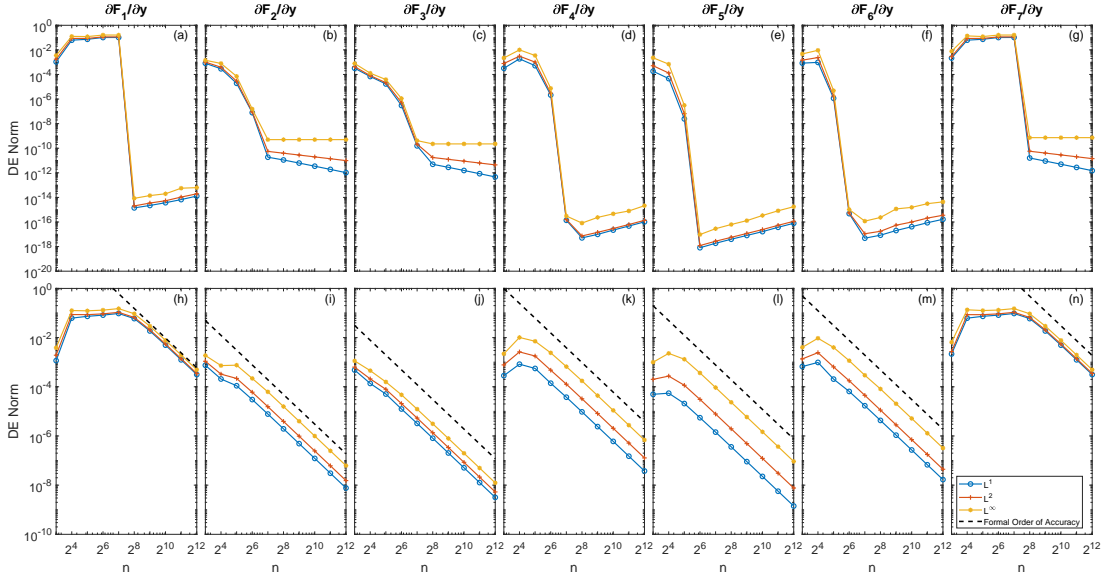


Figure 3.5: Plots of the norm of the discretization error of the y direction derivatives versus the number of grid points, n . The top panels use the exact definition of k_y from Eq. 3.120 and the bottom panels use the approximation, K_y , from Eq. 3.125. The top panels show that the discretization error decreases significantly, typically to machine precision, once the Nyquist criterion is satisfied. The bottom panels show that the discretization error falls at the expected order of accuracy of 2.

top panels show that upon reaching the Nyquist criterion, the discretization error drops significantly, typically to machine precision. Note that for Figure 3.4(a), the coarsest mesh already satisfies the Nyquist criterion. The bottom panels calculate the derivatives using the approximations K and \hat{K}^2 from Eqs. 3.125 and 3.126, respectively. The results indicate that the discretization error decays with the expected order of accuracy of 2.

The results from Figures 3.4 to 3.6 show that using the exact definition of k is always superior to using the approximation. This is definitely true for these simplified functions. However, as the primitive variables evolve through the dynamics of the system, their functional approximations are no longer as well defined and can cause numerical instability. The utility of the approximation is that it allows for the differentiation of complicated functions even if the Nyquist criterion for that function may not necessarily be met (within some degree). This is useful because the computational effort required to meet some Nyquist criteria might be too large to obtain a solution within a reasonable amount of time. Additionally, note how once the Nyquist criterion is met for the exact definition of k , the error, while still more accurate than the approximation, tends to increase, resulting in a numerical scheme that is not consistent. Using the approximation results in a consistent scheme.

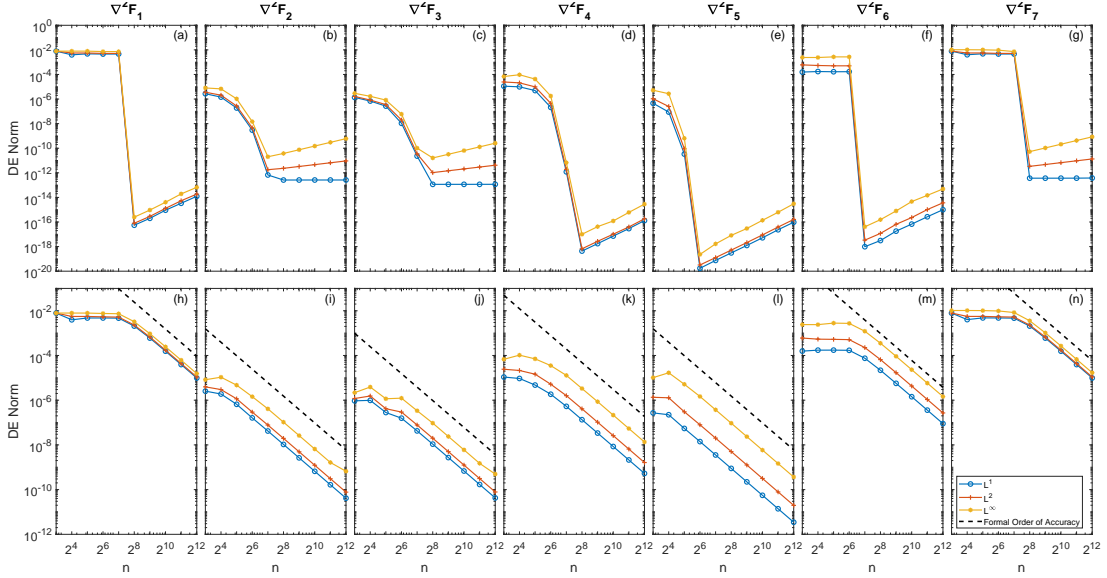


Figure 3.6: Plots of the norm of the discretization error of the Laplacians versus the number of grid points, n . The top panels use the exact definition of k^2 from Eq. 3.120 and the bottom panels use the approximation, \hat{K}^2 , from Eq. 3.126. The top panels show that the discretization error decreases significantly, typically to machine precision, once the Nyquist criterion is satisfied. The bottom panels show that the discretization error falls at the expected order of accuracy of 2.

3.5.3 Iterative Solver

The method of manufactured solutions is used to calculate the observed order of accuracy for the iterative solver for Eq. 3.136. This method is fully discussed in Section 3.4.2. The manufactured solutions for n_{ng} , n_p , ϕ_{bg} , ϕ_p , and $\partial\phi_p/\partial t$ are of the form

$$f(x, y) = \left[a_x \sin\left(\frac{2\pi n_x}{L_x} x\right) a_y \cos\left(\frac{2\phi n_y}{L_y} y\right) + a_{xy} \cos\left(\frac{2\phi n_{xy1}}{L_x} x\right) \sin\left(\frac{2\pi n_{xy2}}{L_y} y\right) + a_0 \right] v_0, \quad (3.178)$$

where a_x , a_y , a_{xy} , n_x , n_y , n_{xy1} , n_{xy2} , a_0 , and v_0 are constant coefficients. The full source terms and coefficients used are found in Appendix B.2.

The grids are refined by a factor of two from 8×8 to 2048×2048 points. Figure 3.7 shows the L^1 , L^2 , and L^∞ norms of the discretization error of $\partial\phi_p/\partial t$. The results follow the same behavior shown in Section 3.5.2. Figure 3.7(a) shows that upon satisfying the Nyquist criterion, the error when using the exact definition of the Fourier mesh significantly drops and remains low. Figure 3.7(b) shows that for the approximation of the Fourier mesh, the error is second order accurate.

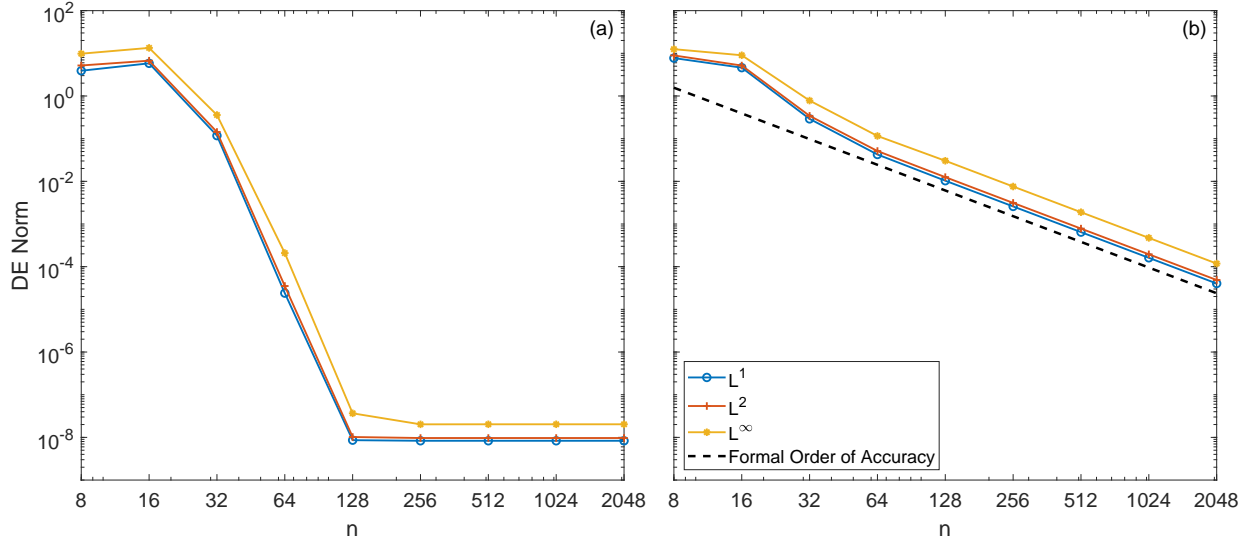


Figure 3.7: Plots of the norm of the discretization error of $\partial\phi_p/\partial t$. Panel (a) uses the exact definition of the Fourier mesh (\mathbf{k} and k^2), whereas Panel (b) uses the approximation of the Fourier mesh, \mathbf{K} and \hat{K}^2 . Both meshes show the expected behavior. Panel (a) shows the error decreasing to its minimum amount upon satisfying the Nyquist criterion. Panel (b) shows the error following the expected order of accuracy of 2 (black dashed line).

3.6 Benchmark Problems

With the code verified, the model now needs to be validated to ensure that it accurately models the intended physics. To truly validate the model would require comparisons to experimental or observational data. For many problems, this is not always feasible. Another method of validating the model is to compare to the results of previously validated models or previously understood benchmark problems.

This section considers three different benchmark problems: the development of the GDI in a slab geometry, the development of the KHI in a slab geometry, and the development of both the GDI and the KHI in a plasma cloud. All three of these problems also explore the effects of varying collisionality on the instability growth.

3.6.1 GDI in Slab Geometry

The gradient drift instability (GDI) is the primary instability that this model attempts to study. A full description of this instability is found in Section 1.3. It is useful to benchmark to the problem of the GDI in a slab geometry that has been extensively studied in the literature [Mitchell Jr et al., 1985b, Keskinen and Huba, 1990, Gondarenko and Guzdar, 1999]. Figure 3.8 shows the model geometry with a low density region on the left and a high

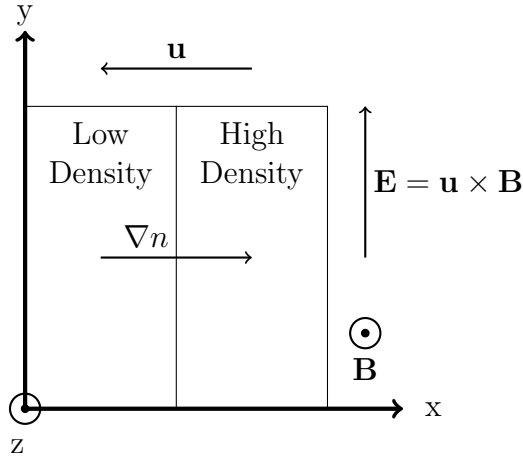


Figure 3.8: A diagram of the slab model of the GDI with labeled axes and important vectors. On the left is a low density region and on the right is a high density region with the corresponding gradient direction labeled. The neutral wind is in the $-\hat{x}$ direction and results in an electric field in the \hat{y} direction based on the \hat{z} direction magnetic field. This is unstable to the F region GDI, as shown in Figure 1.2(b). The density is modeled using a set of hyperbolic tangents, to maintain periodicity, according to Eq. 3.179.

density region on the right, resulting in a density gradient vector in the \hat{x} direction. The background neutral wind in the $-\hat{x}$ direction in conjunction with the background magnetic field in the \hat{z} direction results in an applied electric field in the \hat{y} direction. This geometry is unstable to the F region GDI as per Figure 1.2(b). This sort of density profile is well modeled by a hyperbolic tangent function. To maintain periodicity within the domain, a set of two hyperbolic tangent functions are used. They are of the form

$$n_{bg}(x) = n_0 \left[a^L \tanh \left(\frac{x - x_g^L}{L_g^L} \right) + a^R \tanh \left(\frac{x - x_g^R}{L_g^R} \right) + d \right], \quad (3.179)$$

where n_0 , $a_{L,R}$, $x_g^{L,R}$, and d are constant input coefficients. The full description for the initial condition is provided in Appendix C.1.

The GDI in a slab geometry uses Listing C.1 for the input file. Table 3.1 shows the input parameters used for the collisional GDI case. The parameters use nominal ionospheric F region values. Figure 3.9 shows the resulting plasma density evolution at different times. Figure 3.9(a) shows the initial conditions. Figure 3.9(b) shows the finger-like structures of the GDI that grow at the location of minimum density gradient scale length (which, for a homogeneous neutral wind, is the location of maximum GDI growth based on Eq. 3.180). This is around the end of the linear growth phase of the GDI. Starting in Figure 3.9(c), some of the fingers bifurcate and begin cascading to smaller scales. The late stage evolution of the GDI in Figures 3.9(d-e) show thick structures growing at the tips of many of the leading edges of the instability fingers. These “mushroom-like” structures are sometimes called

Table 3.1: A table of the input parameters for slab GDI simulations using Listing C.1 as the input file. The neutral number density, n_n , is 10^{14} m^{-3} for the collisional case and 10^{12} m^{-3} for the inertial case.

Parameter	Value
$[L_x, L_y]$	$[5 \times 10^2, 2.5 \times 10^2]$ km
$[n_x, n_y]$	[256, 512]
D	$10^3 \text{ m}^2/\text{s}$
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
\mathbf{B}	$[0, 0, 5 \times 10^{-5} \text{ T}]$
\mathbf{u}	$[-500, 0, 0] \text{ m/s}$
m	1
o	2.5
$[x_{g_N}^L, x_{g_N}^R]$	$[1.25 \times 10^2, 3.75 \times 10^2]$ km
$[L_{g_N}^L, L_{g_N}^R]$	[30, 30] km
n_0	10^{11} m^{-3}
T_{i0}	1000 K
T_{e0}	1000 K
n_n	10^{12} or 10^{14} m^{-3}
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-3} n_0$
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

“hammerheads” and are a result of the onset of secondary Kelvin-Helmholtz instabilities (KHI). The velocity shear caused by an instability finger moving into a region of stationary plasma causes the KHI to develop. Qualitatively, the results are consistent with previous findings [Mitchell Jr et al., 1985b, Gondarenko and Guzdar, 1999].

The GDI is expected to behave differently in an inertial, i.e., a less collisional, regime [Osakow et al., 1978, Mitchell Jr et al., 1985b] as indicated by Eq. 1.14. From a qualitative perspective, Mitchell Jr et al. [1985b] show that the inertial GDI has less of a “finger-like” structure and instead develops into the “mushroom-like” structure more readily. Figure 3.10 shows the resulting density plots for the inertial simulation using $n_n = 10^{12} \text{ m}^{-3}$, which corresponds to a significantly smaller collisionality. The first noticeable difference is that the inertial GDI takes significantly longer to grow than the collisional GDI. Note how the end

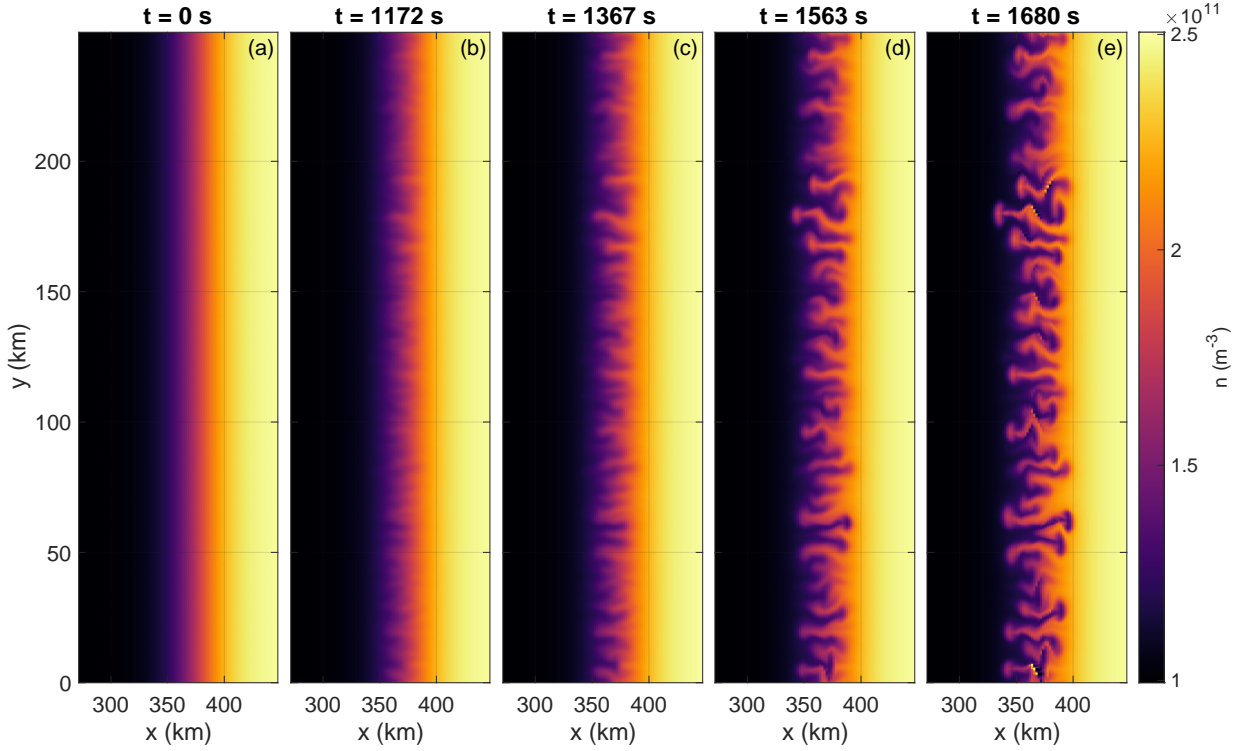


Figure 3.9: Plots of the plasma density showing the GDI evolution at different times using the parameters from Table 3.1. The GDI initially grows as anticipated with thin fingers growing from the minimum density gradient scale length location as shown in Panels (b) and (c). At later times, some of these fingers begin to bifurcate and cascade into smaller scales, as seen in Panel (d). Additionally, the leading edges of the instabilities begin to roll up, leading to the “mushroom-like” structures seen in Panels (d) and (e). These structures are due to the onset of secondary Kelvin-Helmholtz instabilities. Qualitatively, the results match previous work [Mitchell Jr et al., 1985b, Gondarenko and Guzdar, 1999].

of the linear phase in Figure 3.10(b) occurs about 3000 s or about 3.5 times later than the highly nonlinear GDI from Figure 3.9(e). Later in time, the inertial GDI in Figure 3.10(c), enters the nonlinear phase and some of the instabilities begin to bifurcate. At late times, as in Figures 3.10(d-e), the “mushroom-like” structures become prominent and the primary instabilities begin to curve in different directions. The secondary KHI appear earlier due to the inertial terms having a larger effect. This agrees qualitatively with previous work on the inertial GDI [Mitchell Jr et al., 1985b].

To get a quantitative understanding of the GDI growth, Eq. 1.15 [Makarevich, 2014] is simplified into the F region limit. For the geometry in Figure 3.8 and accounting for the inclusion of the numerical diffusion, the simplified growth rate formulation is

$$\gamma = -\frac{u_x}{L} - Dk_y^2, \quad (3.180)$$

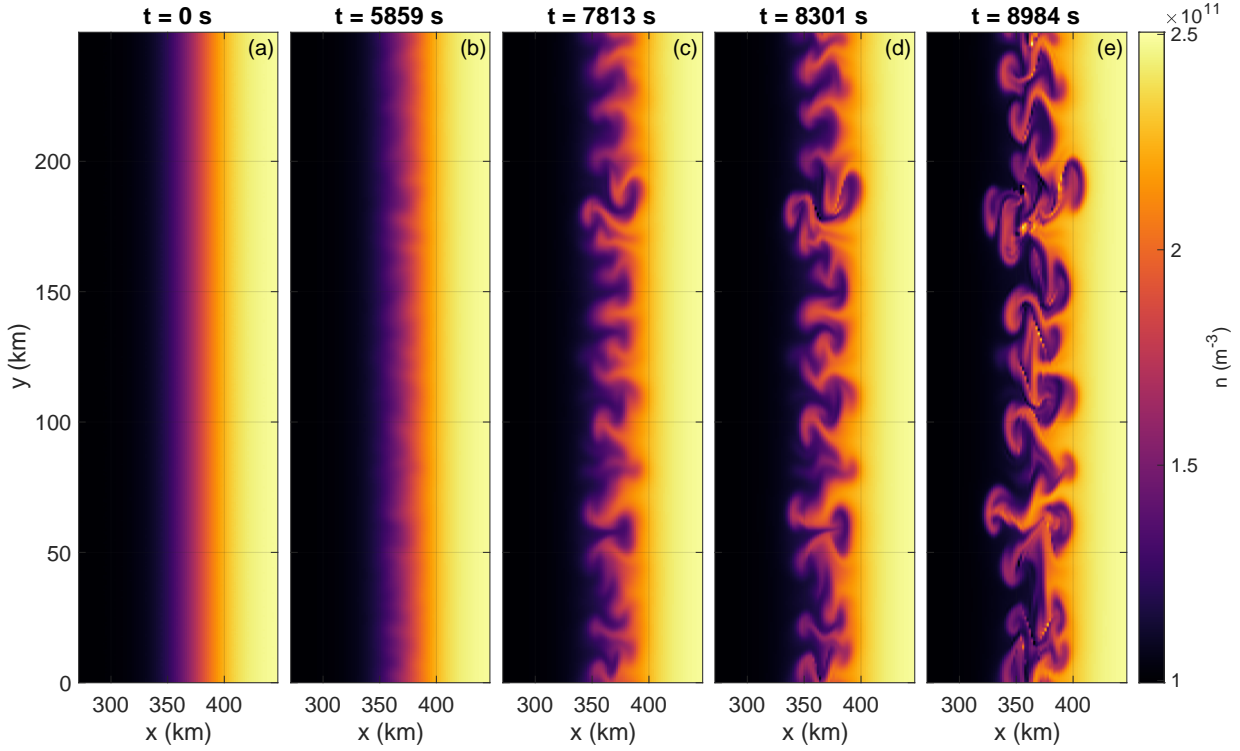


Figure 3.10: Plots of the plasma density showing the inertial GDI evolution at different times using the parameters from Table 3.1. Note how the inertial GDI grows much more slowly than the collisional GDI shown in Figure 3.9. Panel (b) shows how the beginnings of visual GDI growth, near the end of the linear phase, occurs about 3000 s later than the final frame shown in Figure 3.9(e). Note how the structure is less finger-like. Panel (c) shows the growth into the nonlinear phase with visible bifurcations. As time progresses, as in Panels (d-e), due to the decreased collisionality the secondary KHI become more prominent and the individual primary GDI begin to curve into many different directions. Qualitatively, this is consistent with previous simulation work [Mitchell Jr et al., 1985b].

where u_x is the signed component of the neutral wind in the \hat{x} direction, L is the density gradient scale length defined as $L = n(\partial n/\partial x)^{-1}$, D is the numerical diffusion constant, and k_y is the wavenumber in the \hat{y} direction. Note that only the collisional GDI growth is considered because an analytical form for the inertial GDI does not exist in the limits considered; the assumptions used to obtain Eq. 1.14 and Eq. 32 from Makarevich [2019a] are broken as $k_y \rightarrow \infty$. Several single mode simulations are run, using input parameters from Table 3.2, to accurately capture the linear growth effects and mitigate the effects of nonlinear mode coupling. The growth rate is calculated using a least squares regression to find the best exponential fit to the domain-integrated electric field energy as a function of time. Appendix D.1 explains this calculation in further detail. Figure 3.11 plots the growth rate, γ , versus the wavenumber, k , for each of these simulations (blue circles). The red

curve represents the analytical growth rate from Eq. 3.180, which is the short wavelength limit. The black line represents the analytical growth rate from Eq. 1.17, which is the long wavelength limit. As k approaches 0, i.e., as the wavelength approaches ∞ , the simulation growth rate approaches the long wavelength limit from Eq. 1.17. As k approaches ∞ , i.e., as the wavelength approaches 0, the simulation growth rate approaches the short wavelength limit from Eq. 3.180. Therefore, the model accurately captures the expected behavior of the GDI in the linear regime.

Table 3.2: A table of the input parameters for several collisional single mode slab GDI simulations using Listing C.1 as the input file. The y direction domain length is defined based on the wavenumber that is perturbed. The perturbed wavenumbers range from $6.28 \times 10^{-3} \text{ km}^{-1}$ to 1.61 km^{-1} .

Parameter	Value
$[L_x, L_y]$	$[7.5 \times 10^5, 2\pi/k_{pert}] \text{ m}$
$[n_x, n_y]$	[512, 16]
D	$10^3 \text{ m}^2/\text{s}$
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
\mathbf{B}	$[0, 0, 5 \times 10^{-5} \text{ T}]$
\mathbf{u}	$[-500, 0, 0] \text{ m/s}$
m	1
o	2.5
$[x_{gN}^L, x_{gN}^R]$	[220, 525] km
$[L_{gN}^L, L_{gN}^R]$	[50, 75] km
n_0	10^{11} m^{-3}
T_{i0}	1000 K
T_{e0}	1000 K
n_n	10^{14} m^{-3}
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	0: Single Mode
Amplitude	$\pm 10^{-6} n_0$
Mode	1
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

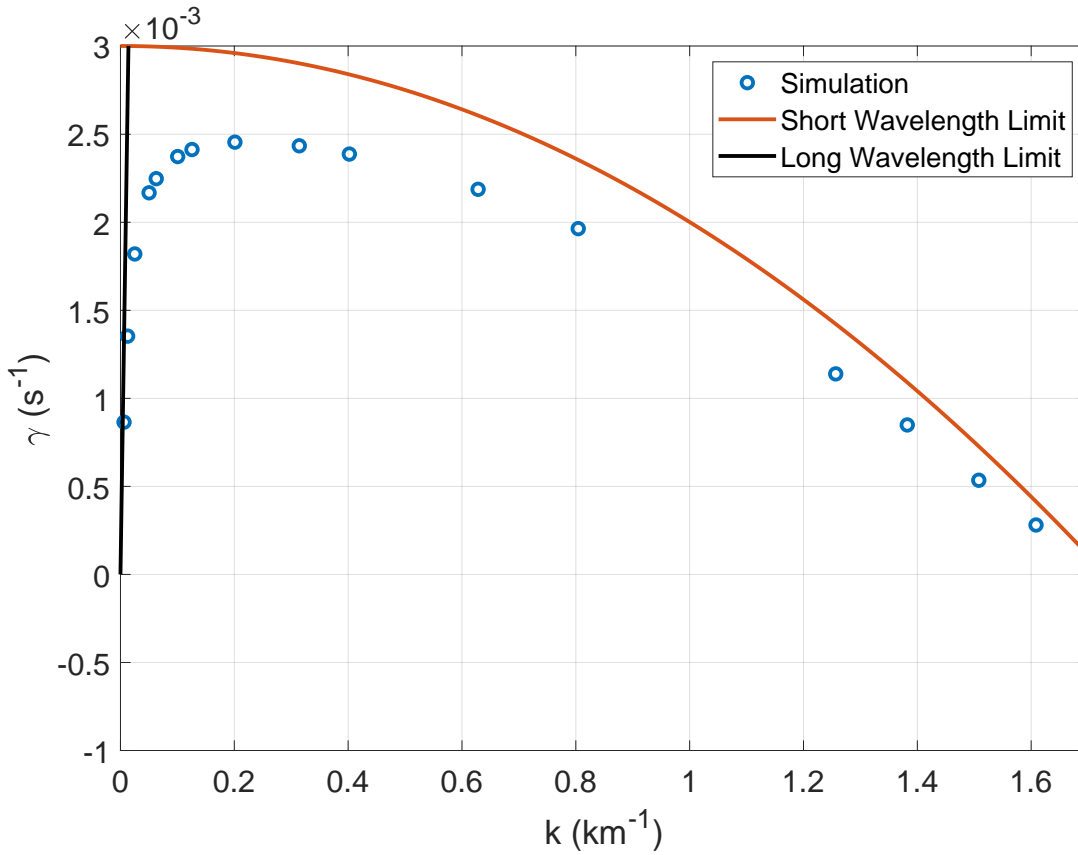


Figure 3.11: The growth rate, γ , vs the wavenumber, k , for a set of single mode GDI simulations whose parameters are found in Table 3.2. The blue circles indicate the simulation growth rates calculated using the method described in Appendix D.1. The red curve is the analytical short wavelength (high wavenumber) limit of the GDI based on Eq. 3.180. The black line is the analytical long wavelength (low wavenumber) limit of the GDI based on Eq. 1.17. The results indicate a successful approximation of the theoretical GDI physics with the simulation growth tending to the appropriate curve as k tends to 0 or ∞ .

3.6.2 KHI in Slab Geometry

A novelty of this model, compared to Keskinen et al. [2004], is the inclusion of inertial effects. Figure 3.10 shows the importance of inertia on the GDI through the development of secondary Kelvin-Helmholtz instabilities (KHI). Figure 3.12 shows how the KHI can itself be setup in a slab geometry to be studied as the primary instability [Keskinen et al., 1988, Keskinen, 1996]. The low density region has a high velocity and the high density region has no velocity. This velocity shear at the density interface is unstable to the KHI, as in Figure 1.3. The density is modeled using Eq. 3.179, as in Section 3.6.1. The velocity is

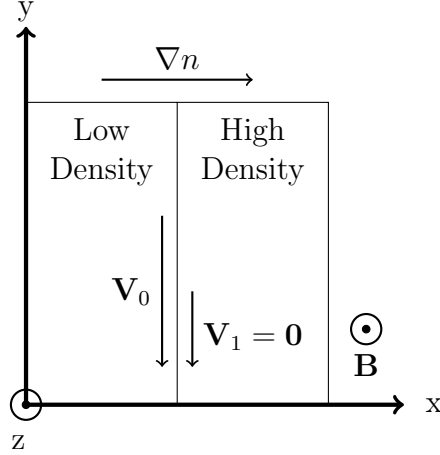


Figure 3.12: A diagram of the slab model of the KHI with labeled axes and important vectors. On the left is a low density region and on the right is a high density region with the corresponding gradient direction labeled. The magnetic field is in the \hat{z} direction. The low density region has a velocity \mathbf{V}_0 in the $-\hat{y}$ direction. The high density region has a velocity of 0. Thus, there exists a velocity shear at the density interface. This is unstable to the KHI as shown in Figure 1.3. The density is modeled using a set of hyperbolic tangents, to maintain periodicity, according to Eq. 3.179. The velocity is modeled using Eq. 3.181 as a set of hyperbolic tangents to maintain periodicity in the velocity and the electric potential.

modeled using a set of hyperbolic tangent functions of the form

$$V_{ybg}(x) = \frac{V_0}{2} \left[\tanh\left(\frac{x - x_{gv}^L}{L_{gv}^L}\right) - \tanh\left(\frac{x - x_{gv}^R}{L_{gv}^R}\right) - \tanh\left(\frac{x - L_x + x_{gv}^R}{L_{gv}^R}\right) + \tanh\left(\frac{x - L_x + x_{gv}^L}{L_{gv}^L}\right) \right], \quad (3.181)$$

where L_x is the domain length in x and V_0 , $x_{gv}^{L,R}$, and $L_{gv}^{L,R}$ are constant input coefficients. The full explanation for the background profile is provided in Appendix C.4.

Figure 3.13 shows the results of a simulation using Table 3.3 as the input parameters with a neutral number density of $n_n = 10^8 \text{ m}^{-3}$ resulting in the development of the KHI with negligible collisions. Figure 3.13(b) shows the end of the linear phase and the start of the classical vortex structure that continues to spiral as seen in Figure 3.13(c). Figures 3.13(d-e) show the nonlinear mode coupling as the individual vortices interact with each other. The instability growth, especially in early times, is consistent with previous work [Keskinen et al., 1988].

Figure 3.14 shows the results of a KHI simulation with $n_n = 10^{13} \text{ m}^{-3}$, which is a more collisional case. The most noticeable difference is that the KHI does not grow as large in

Table 3.3: A table of the input parameters for a slab KHI simulation using Listing C.4 as the input file. The simulations use $n_n = 10^8 \text{ m}^{-3}$ for the low collisional case and $n_n = 10^{13} \text{ m}^{-3}$ for the high collisional case.

Parameter	Value
$[L_x, L_y]$	$[8.5 \times 10^2, 2.5 \times 10^2] \text{ km}$
$[n_x, n_y]$	[512, 512]
D	$10^3 \text{ m}^2/\text{s}$
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
\mathbf{B}	$[0, 0, 5 \times 10^{-5} \text{ T}]$
\mathbf{u}	$[-500, 0, 0] \text{ m/s}$
$x_{gV}^{L,R}$	$[1.25 \times 10^2, 3.75 \times 10^2] \text{ km}$
$[L_{gN}^L, L_{gN}^R]$	[10, 10] km
V_0	-1000 m/s
m	1
o	2.5
$[x_{gN}^L, x_{gN}^R]$	$[1.25 \times 10^2, 3.75 \times 10^2] \text{ km}$
$[L_{gN}^L, L_{gN}^R]$	[30, 30] km
n_0	10^{11} m^{-3}
T_{i0}	1000 K
T_{e0}	1000 K
n_n	$10^8 \text{ or } 10^{13} \text{ m}^{-3}$
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-3} n_0$
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

amplitude as the negligibly collisional case from Figure 3.13. Figure 3.14(b) shows the vortex structure expected from the KHI. However, by Figure 3.14(c), this structure quickly devolves as the collisions begin diffusing the vortex. Figures 3.14(d-e) show how the vortices continue to be broken up into smaller features, a phenomenon called wavebreaking, which is consistent behavior with Keskinen et al. [1988].

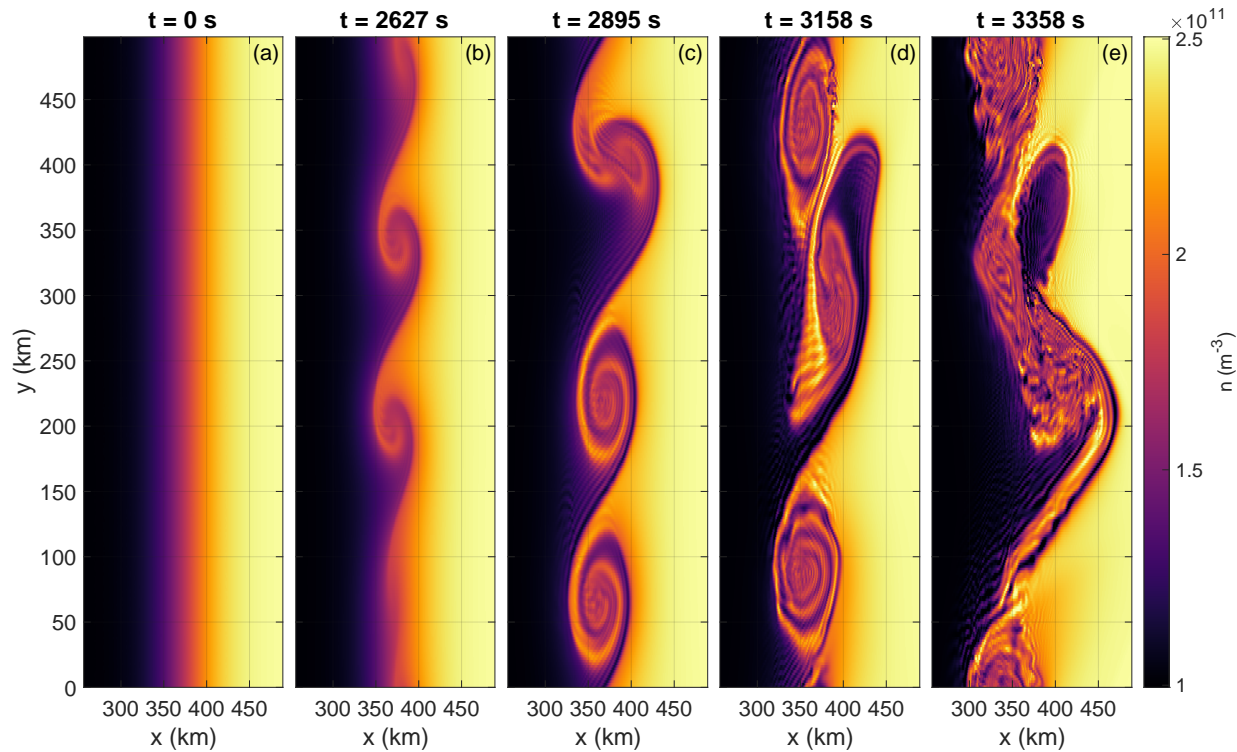


Figure 3.13: Plots of the plasma density to show the KHI evolution at different times using the parameters from Table 3.3 with weak collisions. Panel (a) shows the initial conditions. Panel (b) shows the initial KHI development. Panel (c) sees the KHI vortices begin to roll up. Panel (d) shows the continued roll up as the instability begins to grow in amplitude. Panel (e) shows the nonlinear growth as the individual vortices begin interacting with each other. These results are consistent with previous work in the weakly collisional KHI regime [Keskinen et al., 1988].

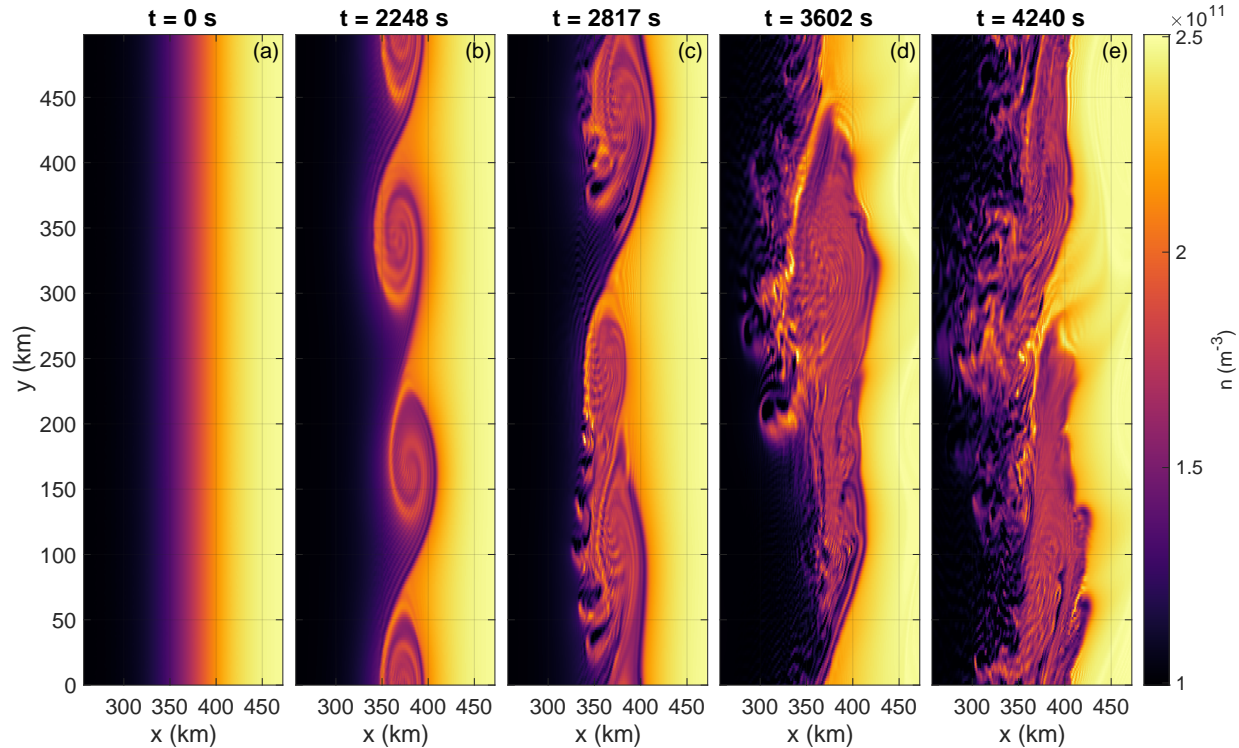


Figure 3.14: Plots of the plasma density to show the KHI evolution at different times using the parameters from Table 3.3 for stronger collisions. Panel (a) shows the initial conditions. Panel (b) shows the traditional vortical structure of the KHI that is also observed in Figure 3.13(c). Note how, for this more strongly collisional case, the amplitude of the KHI is always smaller than in Figure 3.13. Panel (c) shows how the increased collisionality is diffusing the vortex. Panels (d-e) show how, at late times, the collisional KHI breaks up into smaller modes and the traditional KHI structure is lost. This is consistent with simulations done by Keskinen et al. [1988].

3.6.3 Plasma Clouds

The evolution of plasma clouds in the ionosphere is the final benchmark problem considered due to how it beautifully combines the effects of both the GDI and the KHI.

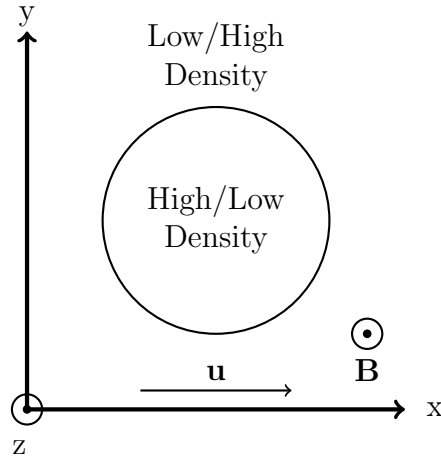


Figure 3.15: A diagram of a plasma cloud in a background plasma with labeled axes and important vectors. The densities are such that if the cloud has a high density, then the background plasma has a low density and vice versa. The magnetic field is in the \hat{z} direction. The neutral wind is in the \hat{x} direction. For plasma enhancements (high density clouds), this configuration is unstable to the GDI on the right interface. For plasma depletions (low density clouds), this configuration is unstable to the GDI on the left interface. The density is modeled using a Gaussian type function shown in Eq. 3.182.

Initially, experiments involving the release of neutral metallic and molecular clouds into the atmosphere were conducted to understand neutral atmospheric phenomena such as diffusion and winds by observing how the evolution of the clouds [Haerendel, 2019]. It was discovered that these neutral metallic clouds, especially barium, can become ionized through photoionization [Armstrong, 1963]. Based on this development, these barium clouds could also be used to study the ionosphere, as well as conduct active experiments that are not easily created in a lab or as readily observed in nature. For example, Haerendel et al. [1967] conducted an active experiment using barium clouds in the ionosphere to artificially simulate the effects of a comet in the background interplanetary medium. They developed a relationship between the ionospheric electric field and the plasma cloud's density and velocity, resulting in a better understanding of both comet tails and the ionospheric electric field. Further experiments showed the development of striations as well as distortions of the barium clouds [Haerendel and Lüst, 1968]. Theoretical work suggested that these structures are caused by the GDI [Linson and Workman, 1970, Perkins et al., 1973]. Further simulations have shown the bifurcations and structures that develop in plasma clouds in the F region ionosphere are due to the background electric and magnetic fields [Zabusky et al., 1973, Bernhardt, 1988, McDonald et al., 1980]. For cloud releases at higher speeds (and at higher altitudes),

the inertial effects become more important in causing the transverse elongation of the cloud [Mitchell Jr et al., 1985a] and vortices at the trailing edge of the cloud [Scales and Bernhardt, 1991]. Studies show that the behavior of the plasma clouds changes depending on the altitude due to the collisionality [Lloyd and Haerendel, 1973]. Furthermore, three-dimensional studies of plasma clouds have been conducted to understand the development of the GDI [Zalesak et al., 1990].

Table 3.4: A table of the input parameters for a plasma cloud simulation using Listing C.5 as the input file. For the collisional case, $\nu_{in} = 1$ Hz. For the inertial case, $\nu_{in} = 0.1$ Hz. For the plasma enhancement case, $a = 4 \text{ m}^{-3}$. For the plasma depletion case, $a = -0.55 \text{ m}^{-3}$. All of the cases use $x_0 = 64 \text{ m}$ except for the inertial plasma depletion case which uses $x_0 = 102.4 \text{ m}$. All of the cases use $D = 0.1 \text{ m}^2/\text{s}$ except for the inertial plasma depletion case which uses $D = 0.05 \text{ m}^2/\text{s}$.

Parameter	Value
$[L_x, L_y]$	[128, 128] m
$[n_x, n_y]$	[256, 256]
D	0.05 or 0.1 m^2/s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
\mathbf{B}	[0, 0, 1 T]
\mathbf{u}	[10, 0, 0] m/s
$[x_0, y_0]$	[64 or 102.4, 64] m
$[r_{0x}, r_{0y}]$	[20, 20] m
d	1 m^{-3}
a	-0.55 or 4 m^{-3}
p	3
b	0.015
c	12
T_{i0}	1000 K
T_{e0}	1000 K
constCollSwitch	1: Constant Collision Frequencies
ν_{in}	0.1 or 1 Hz
ν_{en}	0 Hz
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

Figure 3.15 provides a diagram of the plasma cloud geometry. The density is modeled as a Gaussian type function of the form

$$n(x, y) = a \exp \left[- \left(\left[\frac{x - x_0}{r_{0x}} \right]^2 + \left[\frac{y - y_0}{r_{0y}} \right]^2 \right)^p \left(\left[1 - b \cos(c\theta) \right] \right)^p \right] + d, \quad (3.182)$$

where a , b , c , d , $r_{0x,y}$, x_0 , and y_0 are constant input coefficients. The angle θ is defined about the point (x_0, y_0) such that $\theta = \text{atan2}(y - y_0, x - x_0) - \frac{\pi}{c}$. These initial conditions are explained in further detail in Appendix C.5. The four simulation cases are run using Listing C.5 as the input file with parameters from Table 3.4.

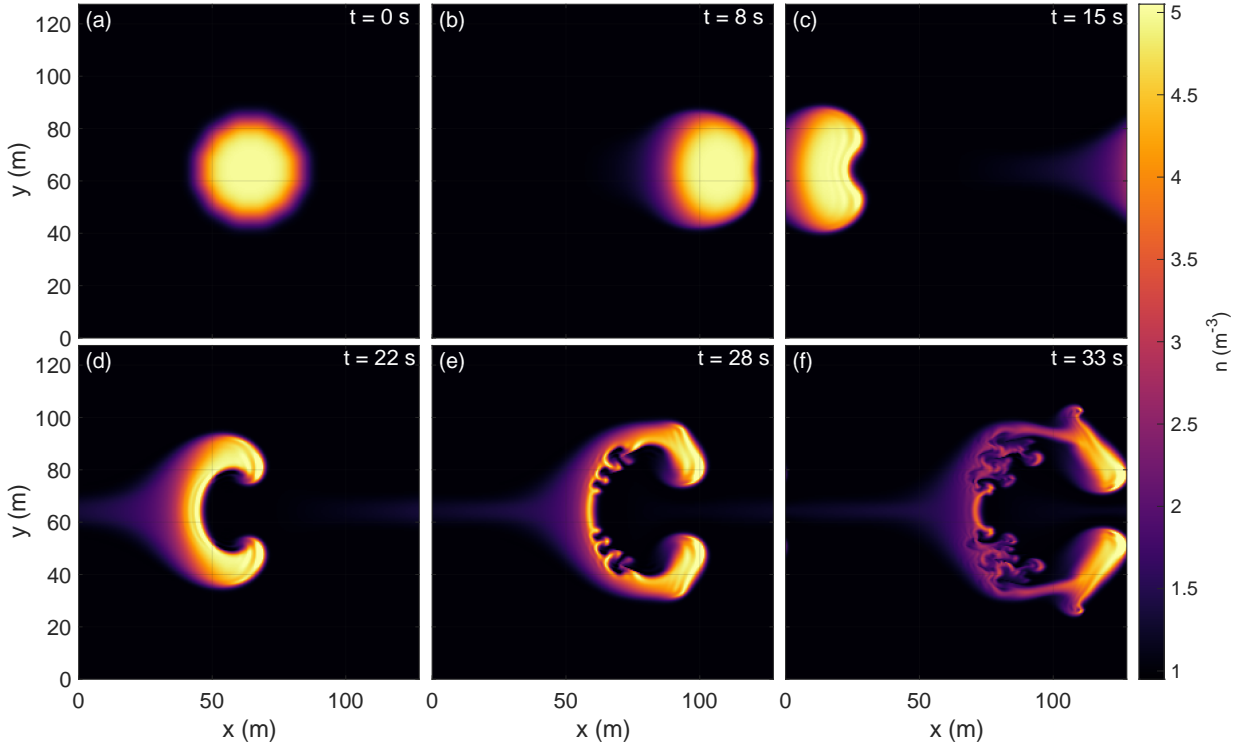


Figure 3.16: Density plots showing the evolution of a plasma enhancement cloud in a collisional regime. Panel (a) shows the initial cloud. The cloud then begins to move to the right as seen in Panel (b) and begins a GDI induced bifurcation as further shown in Panel (c). In Panel (d), the bifurcation tips begin to curl into themselves into a secondary KHI. By Panel (e), the trough of the bifurcation becomes unstable to secondary GDI. Each of these GDI also indicate the onset of tertiary KHI by the “mushroom heads”. Panel (f) shows continual instability development all around the plasma cloud. Note that since the neutral wind is relatively fast, the cloud elongates in x ; this is shown by the tail progressively getting longer in time. In the collisional regime, the plasma cloud structure is dominated by the development of the GDI.

Note that for Figures 3.16-3.19, the boundary conditions are double periodic. This means that anything that leaves the domain on one side reappears on the other side.

Figure 3.16 shows the results of a collisional plasma enhancement cloud simulation. Because the cloud density is higher than the background density, its interaction with the \hat{x} direction neutral wind causes the entire cloud to drift to the right, as shown in Figure 3.16(b). As the cloud moves further right, a trough begins to form on the leading edge, which is seen in

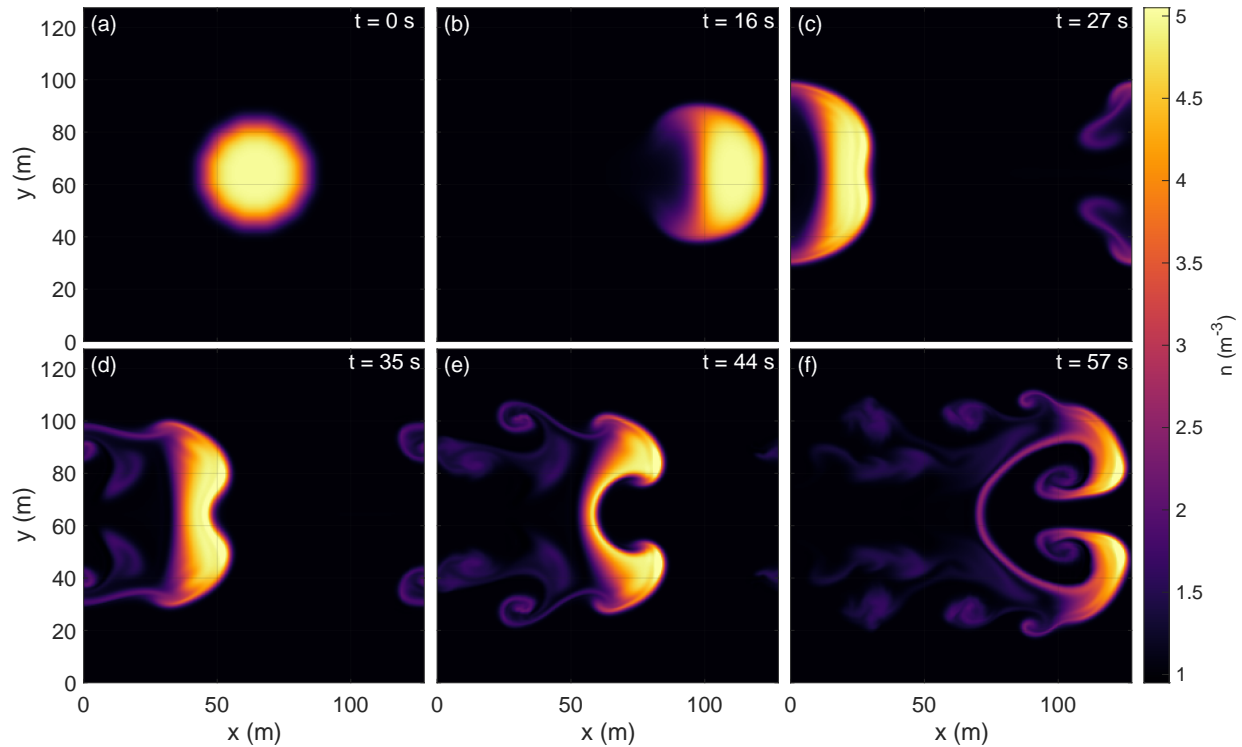


Figure 3.17: Density plots showing the evolution of a plasma enhancement cloud in an inertial regime. Panel (a) shows the initial cloud. Panel (b) shows the cloud moving to the right with the trailing edge beginning to roll up into a vortex-like structure. Panel (c) shows the beginning of a GDI caused bifurcation on the face of the leading edge. Panel (d) shows the continued growth of the GDI as well as the trailing edge vortices continuing to roll up. Panel (e) shows the continued vortex shedding as well as the onset of a secondary KHI from the GDI bifurcation. Panel (f) shows the main structure begin to roll up into a large secondary KHI. Note that due to the low collisionality, there is no tail since the trailing edge vortices dominate the structure. Thus, in the inertial regime, the plasma cloud structure is dominated by the development of the KHI.

Figure 3.16(b). This edge is unstable to the GDI and begins to bifurcate. Figure 3.16(d) shows the continued growth of the GDI. The leading edges begin to curl into themselves, showing the onset of a secondary KHI. The trough interface is now unstable to the GDI, as shown by the additional growing instabilities in Figure 3.16(e). Each of these instabilities also cascades into smaller secondary KHI, as evidenced by the “mushroom” structures. Figure 3.16(f) shows the continued development of these inner instabilities as well as secondary KHI appearing on the edges of the leading bifurcations. Since the neutral wind is relatively fast, a tail is seen progressively getting longer over time. Based on these results, it is clear that the GDI dominates the structuring of a plasma cloud in the collisional regime.

Figure 3.17 shows the results of a plasma enhancement cloud simulation in an inertial regime.

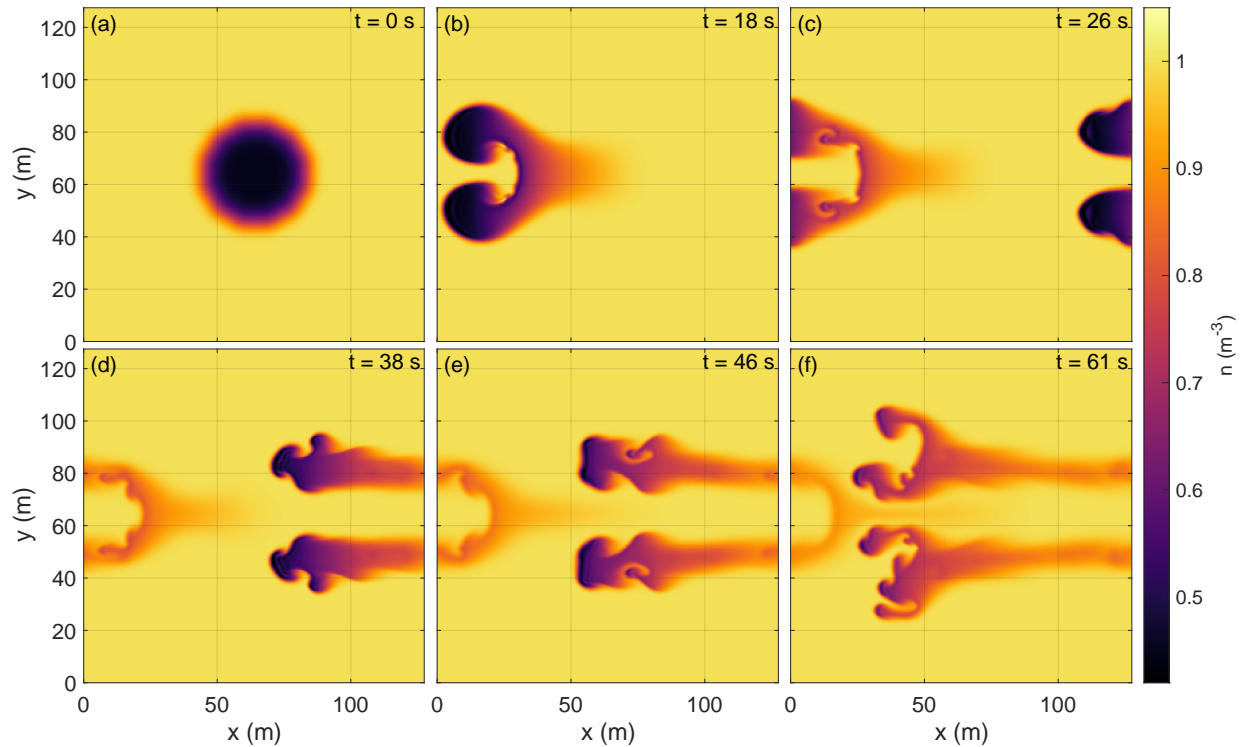


Figure 3.18: Density plots showing the evolution of a plasma depletion cloud in a collisional regime. Panel (a) shows the initial cloud. Note that because this is a plasma depletion, the effect of the neutral wind causes the cloud to drift to the left, instead of to the right, as shown in Panel (b). The cloud has an initial GDI induced bifurcation. In the trough region, secondary instabilities develop. In Panel (c), these secondary instabilities grow further. Panel (d) shows the two new leading edges (because of the bifurcation) cascading into smaller structures. Panels (e-f) further show the continual bifurcations caused by the GDI. Note that the effects in the last few panels are not entirely realistic because the plasma cloud is interacting with its tail. Note how the structures in Panel (f) become pushed away from the center. Much like in Figure 3.16, the GDI dominates in the collisional regime.

Figure 3.17(b) shows the cloud moving to the right; immediately, the trailing edge begins turning in on itself into the start of a vortex. Figure 3.17(c) shows a small trough forming on the leading edge caused by the slow development of the GDI. Figure 3.17(d) shows the continued development of the trailing edge vortices. The GDI begins to roll in on itself due to a secondary KHI, as seen in Figure 3.17(e). The trailing edge vortices continue to roll up. Figure 3.17(f) shows the increasing growth of the secondary KHI of the main GDI bifurcation. Since the collisionality is low, no tail develops, as in Figure 3.16. Instead, the trailing edge vortices dominate the tail structure. Thus, in the inertial regime, the plasma cloud structure is dominated by the development of the KHI.

Figure 3.18 shows the results of a collisional plasma depletion cloud simulation. In this

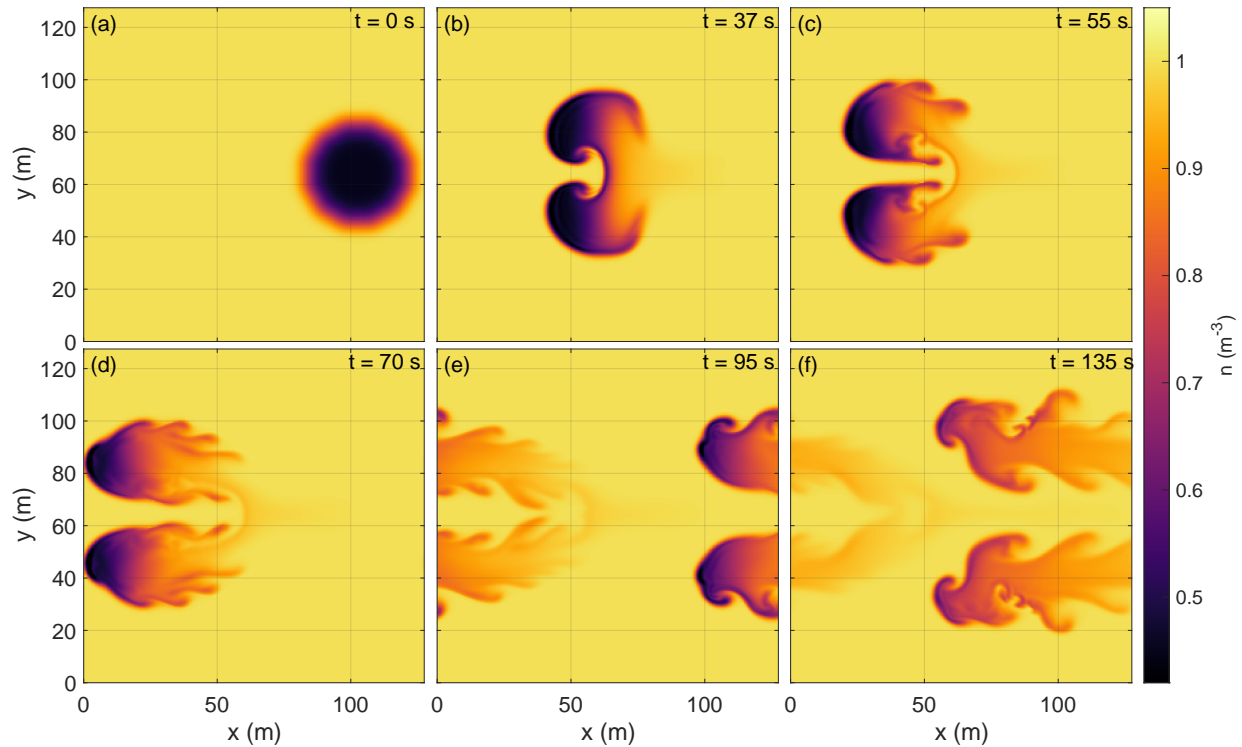


Figure 3.19: Density plots showing the evolution of a plasma depletion cloud in an inertial regime. The cloud is initialized further to the right, as seen in Panel (a), so that the structures that develop are easier to see. Panel (b) shows the initial GDI induced bifurcation along with a secondary KHI. Panel (c) shows the development of structures at the trailing edge of the cloud. Panel (d) shows these structures evolving in time. Panel (e) shows those structures developing into trailing KHI as well as KHI developing near the leading edge of the bifurcated cloud. Panel (f) shows the further development of KHI as the cloud moves to the right. Note that the effects in the last few panels are not entirely realistic because the plasma cloud is interacting with its tail. Note how the structures in Panel (f) become pushed away from the center. As in Figure 3.17, the inertial regime is dominated by the KHI.

case, because the cloud has less density than the background plasma, the cloud moves to the left despite the neutral wind in the \hat{x} direction, as shown in Figure 3.18. The leading edge quickly bifurcates due to being unstable to the GDI. The trough region then begins to develop secondary GDI. The secondary instabilities continue to grow through Figure 3.18(c). Figure 3.18(d) shows the two new leading edges develop finer structures. Figure 3.18(e) shows the continued development of these structures until the leading edges each bifurcate into smaller structures, as seen in Figure 3.18(f). The results from Figures 3.18(d-f) are not entirely realistic since the cloud begins to interact with its own tail. Note how the bifurcations are pushed away from the center in Figure 3.18(f). However, these results still

have value in minimally understanding the late stage behavior of the continual bifurcations. As in Figure 3.16, the GDI dominates the structural development of the plasma cloud in the collisional regime.

Figure 3.19 shows the results of an inertial plasma depletion cloud simulation. The cloud bifurcates due to the GDI and develops a secondary KHI, as seen in Figure 3.19(b). Figure 3.19(c) shows the development of damped trailing edge vortices. These are seen to develop further in Figure 3.19(d). The outer faces of each bifurcated cloud begin to develop KHI, shown in Figure 3.19(e). Figure 3.19(f) shows these KHI develop further along with the growth of additional smaller scale KHI. As with Figure 3.18, the last panel is not entirely realistic due to the cloud interacting with its tail. However, it still shows that, in the inertial regime, the KHI is dominant and develops on the shear interfaces of the bifurcated plasma cloud.

Figures 3.16-3.19 show the expected qualitative behavior of the plasma clouds in the different regimes. The instabilities that grow, bifurcate, and cascade into smaller instabilities showcase the model's ability to accurately study ionospheric turbulence.

Chapter 4

Modeling the Gradient Drift Instability in Subauroral Polarization Streams

4.1 Subauroral Polarization Streams (SAPS)

Subauroral polarization streams (SAPS) are latitudinally narrow regions of a strong poleward electric field that, in conjunction with Earth's geomagnetic field, drive large westward plasma flows via the $\mathbf{E} \times \mathbf{B}$ drift [Foster and Burke, 2002]. These typically occur in the dusk-midnight sector. The term SAPS is a generalization of ionospheric phenomena involving narrow channels with high velocities. Other names that have been used in the literature for these phenomena are subauroral ion drifts (SAID) [Spiro et al., 1979] and polarization jets (PJ) [Galperin, 2002]. SAPS are the result of a magnetosphere-ionosphere coupling process caused by Region 2 field-aligned currents (FACs) [Kelley, 2009, Section 8.5] partially flowing into the ionosphere due to pressure gradients in the ring current [Harel et al., 1981, He et al., 2018]. This allows current continuity to be maintained for Region 1 and Region 2 FACs through the developed meridional electric field [Mishin et al., 2017]. The increased current, and therefore increased electric field, causes an increase in the plasma velocity. Since the neutral particles are unaffected by the electric field, the increase in plasma velocity causes an increase in the frictional drag between the plasma and the neutrals. This increases the ion recombination rate and, consequently, decreases the plasma density [Schunk et al., 1976]. The lower densities result in a lower conductance, causing a feedback loop that lowers the density further; this creates an ionospheric trough with a latitudinal profile approximately co-located with SAPS [Anderson et al., 1993].

A significant amount of statistical work has been done to gain a better understanding of the general features of SAPS. Foster and Vo [2002] used data from the Millstone Hill incoherent

scatter radar to understand the SAPS structure longitudinally. SAPS are faster (about 900 m/s) and wider (about 3° to 5°) latitudinally in the dusk-midnight sector than in the midnight-dawn sector (about 400 m/s and 3°). Using DMSP satellite data, Wang et al. [2008] found that there is a correlation between strongly perturbed geomagnetic conditions and the location of peak SAPS velocity occurring more equatorward. This was corroborated by Erickson et al. [2011], who used the same dataset as Foster and Vo [2002]. Additionally, they found that the SAPS velocity tends to be smaller at higher MLT. Kunduri et al. [2017] used SuperDARN radar data to show that SAPS tend to occur more frequently in more disturbed geomagnetic conditions. They corroborated the previous findings, showing higher velocities and a more equatorward peak in storm-time conditions. SAPS have also been studied to understand their long term occurrences and variations seasonally and at different points in the solar cycle from a climatological perspective [He et al., 2014, Aa et al., 2020]. More recent studies have begun looking at the effect SAPS has on the thermospheric winds [Wang et al., 2011, Zhang et al., 2015, Wang et al., 2018].

In addition to the large scale features of SAPS, many observations have been made using both satellite and radar data to show the presence of density irregularities within SAPS. Erickson et al. [2002] observed irregularities in the E region electric field within SAPS. Mishin et al. [2003] additionally observed electric field irregularities along with the presence of density irregularities in SAPS using DMSP satellites; these have also been observed in SuperDARN data [Oksavik et al., 2006] and GPS scintillation data [Basu et al., 2001, Ledvina et al., 2002]. Many other works match these observations, suggesting that the density and electric field irregularities are a common feature of SAPS [Foster et al., 2004, Mishin and Burke, 2005, Mishin and Blaunstein, 2008]. The electromagnetic portion of the irregularities have been explained by Alfvén wave propagation from the E region [Streltsov and Mishin, 2003, Streltsov and Foster, 2004, Mishin and Burke, 2005]. However, the cause of the electrostatic portion of the irregularities has not yet been identified. Several plasma instabilities have been hypothesized to cause these irregularities [Mishin et al., 2003, Mishin and Blaunstein, 2008, Keskinen et al., 2004]. These are the temperature gradient instability [Hudson and Kelley, 1976, Greenwald et al., 2006, Eltrass and Scales, 2014, Eltrass et al., 2014], the ion frictional heating instability [Keskinen et al., 2004], and the gradient drift instability (see Section 1.3). This study focuses on modeling the gradient drift instability (GDI). Due to the background SAPS morphology of the westward velocity, the density trough, and the background neutral wind, the GDI is a prime candidate for explaining the observed density irregularities.

Initial work modeled SAPS potentials empirically [Goldstein et al., 2005] and more recent empirical work provides better predictive capabilities [Kunduri et al., 2018]. Because SAPS are the result of the complicated coupling of the magnetosphere-ionosphere-thermosphere (MIT) system [He et al., 2017], much of the work on SAPS modeling has coupled empirical models to first-principle models. As the studies started implementing more first-principle models and less empirical models, their accuracy compared to observations tended to increase [Zheng et al., 2008, Wang et al., 2012b, Yu et al., 2015]. Raeder et al. [2016] demonstrated the

ability to model self-consistent SAPS behaviors and Lin et al. [2019] expand upon this to be the first to model SAPS fully self-consistently, incorporating the entire MIT system using the Coupled Magnetosphere Ionosphere Thermosphere (CMIT) model. Significant work has been done specifically in understanding the effect of SAPS on the neutral winds and temperatures in the thermosphere [Wang et al., 2012b, 2018, Ferdousi et al., 2019, Zhang et al., 2021]. All of the models described thus far are global in nature, with the Thermosphere-Ionosphere-Electrodynamics General Circulation Model (TIE-GCM) achieving a finest resolution of 0.625° in latitude and longitude [Dang et al., 2020]. However, these resolutions are not fine enough to model the irregularities observed. Initial work has been done on explaining the electromagnetic component of the observed irregularities at finer scales [Streltsov and Mishin, 2003, Streltsov and Foster, 2004]. The model described in Chapter 3 is used to simulate the, as of yet, unmodeled electrostatic irregularities that develop within SAPS, with specific focus on the GDI.

4.2 Model Initialization

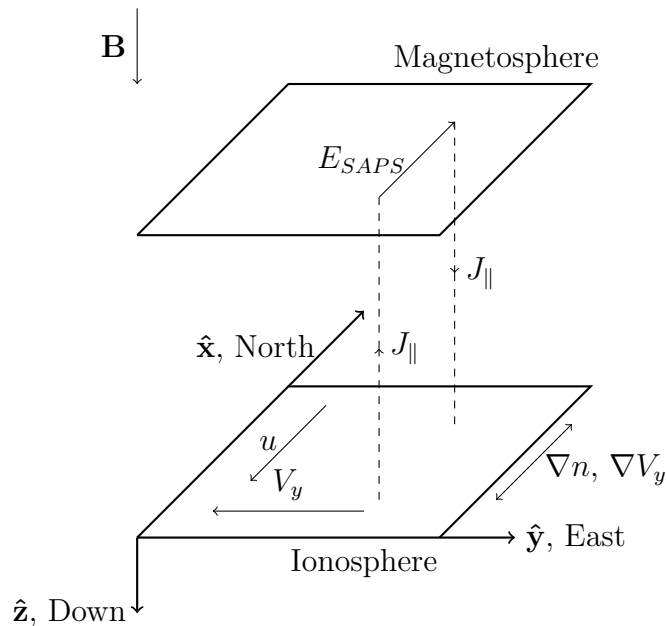


Figure 4.1: A diagram of the coordinate system used, where \hat{x} is north, \hat{y} is east, and \hat{z} is down. Since this model only considers motion perpendicular to the magnetic field, the simulation domain is in the xy plane with the magnetic field in \hat{z} . The westward SAPS flow, V_y , is in $-\hat{y}$. Both the density, n , and the SAPS velocity have latitudinal profiles and thus have gradients in $\pm\hat{x}$. The poleward electric field that drives SAPS is in \hat{x} . Note that the parallel currents do not exist within the 2D model and are only shown to provide a physical understanding of the magnetospheric electric field mapping down into the ionosphere.

The phenomenological framework of the problem is set by Figure 4.1. The simulation domain is in the xy plane to model the motion perpendicular to the magnetic field, \mathbf{B} , which is in the $\hat{\mathbf{z}}$ direction. In the northern hemisphere, the geomagnetic field points down; therefore, $\hat{\mathbf{z}}$ is the down direction. To maintain a right-handed coordinate system, $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are then defined as north and east, respectively. Therefore, the westward SAPS velocity is in $-\hat{\mathbf{y}}$. The velocity is caused by the poleward (therefore in $\hat{\mathbf{x}}$) electric field that is mapped down from the magnetosphere into the ionosphere. Note that the parallel currents do not actually exist within the framework of the model and are shown in the diagram to provide an understanding of the underlying physics. Because both the SAPS velocity profile and the mid-latitude trough change with latitude, the velocity and density gradients, ∇V_y and ∇n , respectively, are in the $\pm\hat{\mathbf{x}}$ direction.

The last remaining directional parameter is the neutral wind, which is chosen to be equatorward, i.e., in $-\hat{\mathbf{x}}$. During storm-time conditions, the meridional neutral wind is frequently observed to be in the equatorward direction [Buonsanto, 1999]. Poleward winds have been observed by Zhang et al. [2015] with agreement from TIE-GCM simulations by Zhang et al. [2021]. Additionally, Mishin and Blaunstein [2008] have measured background electric fields, suggesting the existence of meridional neutral winds in the poleward and equatorward directions for different SAPS events. For the scope of this work, only the equatorward direction is considered for the meridional wind. The anticipated effect of a poleward neutral wind is briefly commented on in Section 4.4. Additionally, westward zonal winds have been observed in SAPS [Wang et al., 2011, 2012a, Zhang et al., 2015, Ferdousi et al., 2019] and are considered in only one set of simulations to show that they have minimal impact on the ability of the GDI to develop.

The background velocity configuration is based on measurements from the U.S. mid-latitude SuperDARN radars [Kunduri et al., 2017]. The SAPS are assumed to be perfectly westward with a cosine correction factor [Erickson et al., 2011, Kunduri et al., 2018]. The median profiles with standard deviations are found at different MLTs within the dusk-midnight sector. The density profile configuration is based on Global Positioning System (GPS) Total Electron Content (TEC) data obtained from the Madrigal Database (<http://millstonehill.haystack.mit.edu/>). The minimum scallop estimation approach [Rideout and Coster, 2006] is used to process the data. The data are binned into $1^\circ \times 1^\circ$ cells and updated every 5 minutes. Through the use of distributed receivers, the spatial coverage is improved using the median filtering technique [Thomas et al., 2013]. From this, the median and standard deviation TEC values are obtained.

Figure 4.2 shows an example of latitudinal GPS TEC (a) and SAPS velocity (b) profiles at 20 MLT for an event on May 2, 2013. These profiles are at an altitude of about 300 km. The SAPS velocity profiles are described by the parameters d and w . The distance between the center of the poleward density gradient and the absolute maximum velocity (when considering the uncertainty) is defined as d . Due to the inherent uncertainty in the measurements, the velocity profile can exist anywhere within the error bar space. The thick black line in Figure 4.2(b) shows a region in which the the velocity profile can have minimal

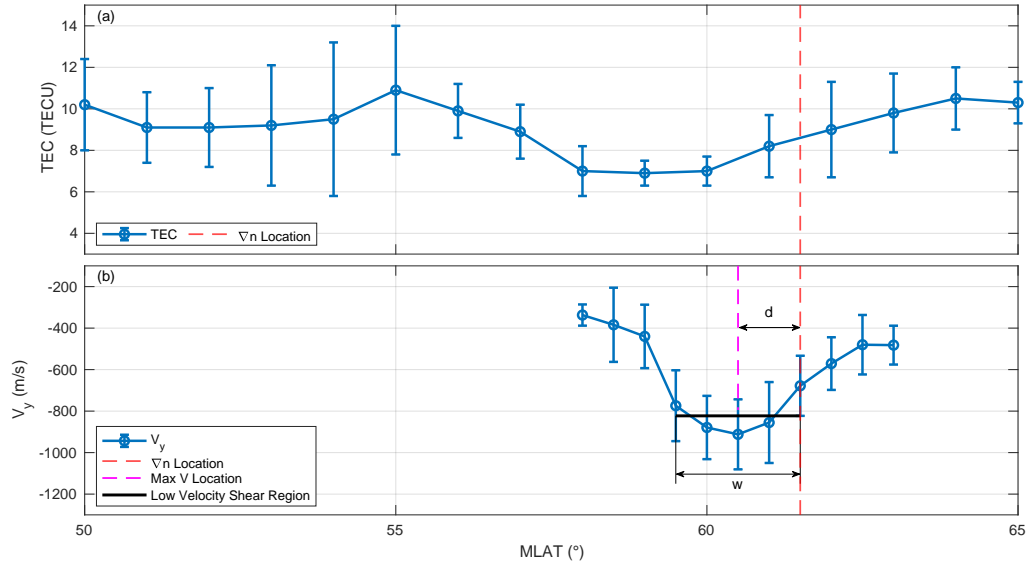


Figure 4.2: Latitudinal profiles at 20 MLT of TEC (a) and SAPS velocity (b) of a May 2, 2013 SAPS event. The red dashed lines indicate the location of the center of the poleward density gradient and the magenta dashed lines indicate the maximum velocity location. The distance between these two points is denoted as d . The thick black line represents a region in which it is possible to maintain a low velocity shear. When considering the total velocity space within the error bars, the SAPS velocity can maintain an approximately constant velocity of 822 m/s. The width of this region is defined as w .

shear. From 59.5° to 61.5° MLAT, the velocity can be an approximately constant 822 m/s and remain entirely within the error bars.

The SAPS profiles are parameterized by four parameters: the minimum velocity gradient scale length (L_V^g), the width of the largest low velocity shear region (w), the distance between the location of the center of the poleward density gradient and the location of maximum velocity (d), and the maximum possible velocity with a low shear in w (V). Note that d is defined such that it is positive if the velocity maximum is equatorward of the poleward density gradient. The TEC data do not provide exact densities; however, the data can be considered proportional to the actual profile of the density. Therefore, the important parameters used to define the density profiles are the minimum density gradient scale lengths for both the poleward and equatorward sides ($L_{N_{P,E}}^g$).

The SAPS profiles used in Section 4.3 are chosen such that GDI growth is expected. In order to help facilitate this, all of the parameters described above are calculated to optimize GDI growth by including the uncertainties. For example, since the GDI growth is inversely proportional to the density gradient scale length (see Eq. 1.15 for the generalized growth rate or Eq. 4.12 for the simplified growth rate in this parameter regime), density profiles with small density gradient scale lengths are expected to show faster GDI growth. To optimize

this, consider the median profile N with uncertainty δN . The derivative of the density with respect to MLAT is calculated using a second order central difference approximation. Traditionally, this approximation is calculated using

$$\frac{\partial N}{\partial x} \approx \frac{N_{j+1} - N_{j-1}}{2\Delta x} \quad (4.1)$$

for a density, N , indexed by j . In this case, to convert from MLAT to km, Δx is

$$\Delta x = (R_E + z) \sin \Delta\theta, \quad (4.2)$$

where $R_E = 6371$ km is the radius of the Earth, z is the altitude (which in this case is 300 km), and $\Delta\theta$ is the distance between points in MLAT. But, to maximize the poleward density gradient, the uncertainty is included to obtain the largest possible difference as

$$\left. \frac{\partial N_j}{\partial x} \right|_{\max_P} = \frac{(N_{j+1} + \delta N_{j+1}) - (N_{j-1} - \delta N_{j-1})}{2\Delta x}. \quad (4.3)$$

Similarly, the maximum equatorward gradient is

$$\left. \frac{\partial N_j}{\partial x} \right|_{\max_E} = \frac{(N_{j+1} - \delta N_{j+1}) - (N_{j-1} + \delta N_{j-1})}{2\Delta x}. \quad (4.4)$$

From Eqs. 4.3 and 4.4, the minimum possible density gradient scale lengths for the poleward and equatorward density gradient can be calculated using

$$\min L_{N_{P,E}}^g = \min \left| \frac{N_j - \delta N_j}{\left. \partial N_j / \partial x \right|_{\max_{P,E}}} \right|. \quad (4.5)$$

Similarly, the absolute value of the maximum velocity derivative, using the notation of V to represent the median and δV to represent the standard deviation, is

$$\left. \frac{\partial V_j}{\partial x} \right|_{\max} = \frac{\max \left[\left| (V_{j+1} - \delta V_{j+1}) - (V_{j-1} + \delta V_{j-1}) \right|, \left| (V_{j+1} + \delta V_{j+1}) - (V_{j-1} - \delta V_{j-1}) \right| \right]}{2\Delta x}. \quad (4.6)$$

Note that because the direction of the maximum velocity derivative is not as important, the absolute value is maximized. Therefore, the minimum velocity gradient scale length is

$$\min L_{V^g} = \min \left| \frac{V_j - \delta V_j}{\left. \partial V_j / \partial x \right|_{\max}} \right|. \quad (4.7)$$

The distance d is calculated using

$$d = x_{\nabla N_P} - x_{V_{\max}}, \quad (4.8)$$

where $V_{\max} = \max |V \pm \delta V|$ and $x_{\nabla N_P}$ is the location in between the point of minimum density and the point of maximum density poleward of the point of minimum density. The maximum width of low velocity shear is found by iterating through all of the velocity points and finding the region of most points that share overlapping error bars. A constraint put on this region is that it must include the location $x_{V_{\max}}$ to make sure that the main portion of the SAPS is captured (and not some effect at the boundary or perhaps not even in the SAPS). Lastly, the maximum velocity in this region that overlaps the error bars of all the points in the region is found.

All of these calculations define the wanted parameters for one profile. Figure 4.2 shows the latitude profiles for a singular MLT cut of a singular SAPS event. As discussed in Section 4.1, SAPS profiles can vary largely with position, geomagnetic conditions, and other factors [Foster and Vo, 2002, Wang et al., 2008, Erickson et al., 2011, Kunduri et al., 2017]. One single profile does not suffice in understanding the general behavior of SAPS. Therefore, 10 different SAPS events are considered to understand the parameter space. Within those events are 86 total velocity profiles at different MLT and 100 total TEC profiles at different MLT; 42 of these profiles overlap in time and space. Figure 4.3 shows a set of histograms for all of the data considered. Figure 4.3(a) shows that the minimum velocity gradient scale length, L_V^g , varies from 10 to 200 km. Figure 4.3(b) shows that the largest width of low velocity shear, w , varies from 100 to 1000 km. Figure 4.3(c) shows that the minimum poleward density gradient scale length varies from about 20 to 700 km. Figure 4.3(d) shows that the parameter d varies from about -350 to 580 km. Figure 4.3(e) shows that the maximum possible velocity that maintains a low velocity shear within the region of w varies from 200 to 1400 m/s. Figure 4.3(f) shows that the minimum equatorward density gradient scale length varies from about 30 to 540 km.

Based on the data, the density can be well approximated by using a set of hyperbolic tangent functions as

$$n_{bg} = n_0 \left(a_1 \tanh \left[\frac{x - x_{NE}}{L_{NE}} \right] + a_2 \tanh \left[\frac{x - x_{NP}}{L_{NP}} \right] + c \right), \quad (4.9)$$

where n_0 is a reference density, L_N is a length scaling factor, x_N is the location of the density gradient, subscripts E and P denote the equatorward and poleward sides, respectively, and a_1 , a_2 , and c are constants chosen to have a reasonably good fit to the data. The same background density profile is used throughout Section 4.3; this profile is shown in Figure 4.4(a). Additionally, the perturbed density is initialized to be a random noise perturbation with an amplitude of $\pm 10^{-6} n_0$.

Based on the shape of the SAPS velocity profile from Figure 4.2, a hyperbolic secant squared function acts as a reasonable approximation. The function is of the form

$$V_{ybg}^s = V_0^s \operatorname{sech}^2 \left[\frac{x - x_{NP} + d}{L_V^s} \right], \quad (4.10)$$

where V_0^s is a reference velocity, L_V^s is a length scaling factor, and d is the distance between the function extremum and the poleward density gradient location as defined in Figure 4.2.

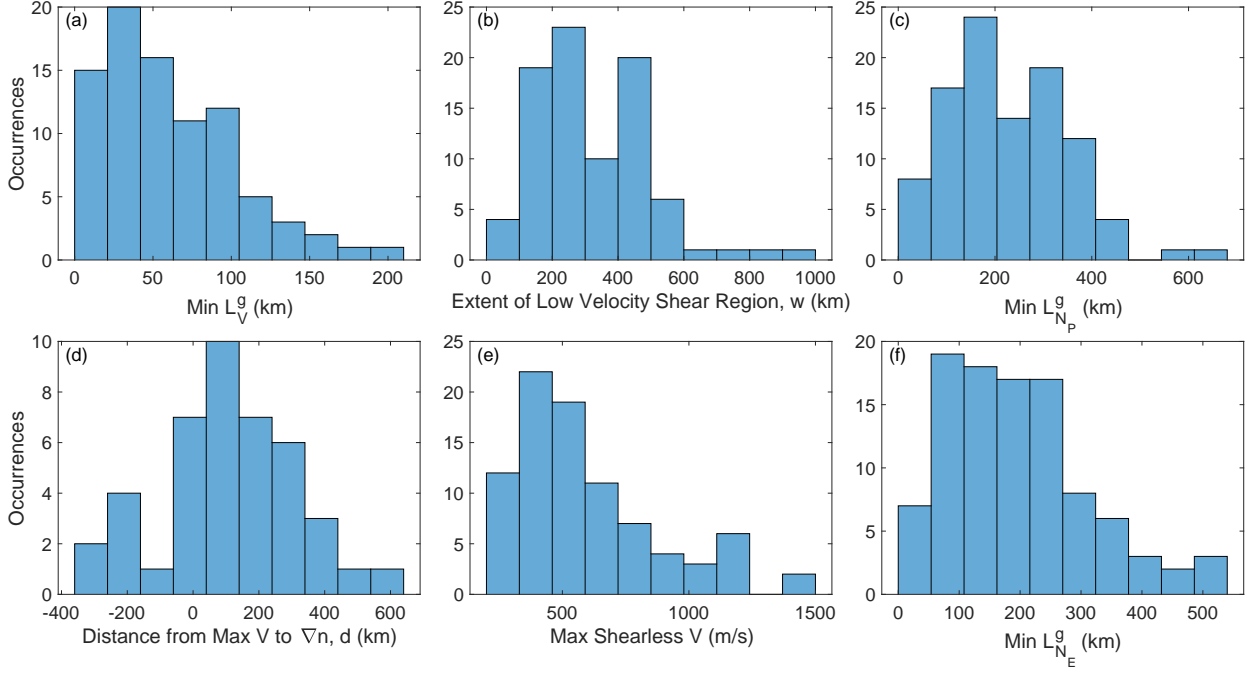


Figure 4.3: Histograms of different latitudinal TEC and velocity profile parameters for 10 different SAPS events across multiple MLT cuts. Panels (a-f) represent, respectively, the minimum velocity gradient scale length (L_V^g), the largest width of low velocity shear around the peak velocity location (w), the minimum density gradient scale length on the poleward edge ($L_{N_P}^g$), the distance between the location of maximum velocity and the location of the center of poleward density gradient (d), the maximum velocity with low velocity shear in the region of w , and the minimum density gradient scale length on the equatorward edge ($L_{N_E}^g$). Note that d is defined such that it is positive if the velocity maximum is equatorward of the poleward density gradient. These histograms guide the choice of parameters in Table 4.1.

Figure 4.4(b) shows an example of this type of background velocity profile. Section 4.3 considers the effects of changing d on GDI growth in SAPS.

One issue with using Eq. 4.10 for the background velocity is that the function inherently couples the width of the velocity channel and the velocity gradient scale length. Instead, using a set of hyperbolic tangent functions, similar to Eq. 4.9, allows w and L_V to be decoupled to independently understand their effects on GDI growth. The resulting function is

$$V_{y_{bg}}^t = \frac{V_0^t}{2} \left(\tanh \left[\frac{x - x_{N_P} + w}{L_V^t} \right] - \tanh \left[\frac{x - x_{N_P}}{L_V^t} \right] \right), \quad (4.11)$$

where V_0^t is a reference velocity and w is the distance between the equatorward and poleward velocity gradients (as defined in Figure 4.2). Eq. 4.11 allows for the gradient scale length and the width of the profile to be changed independently. An example of this type of profile is provided in Figure 4.4(c). Section 4.3 considers the effects of changing L_V^t and w on GDI

growth in SAPS.

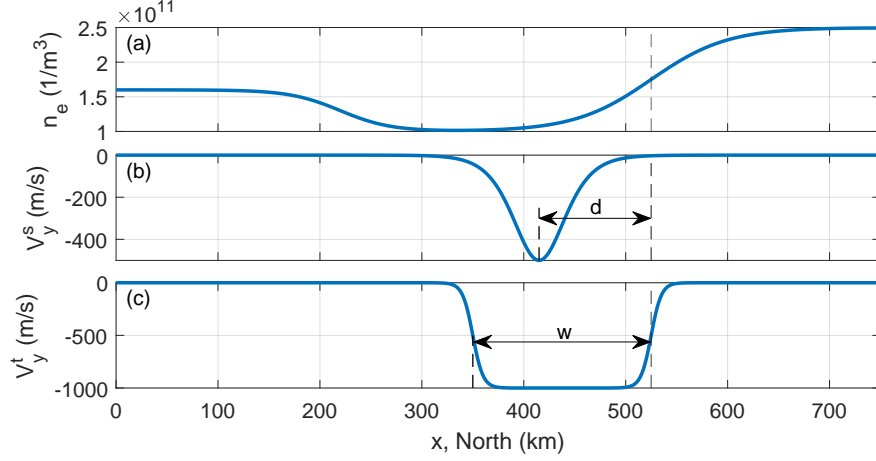


Figure 4.4: Plots of background density and velocity profiles. The functions vary in x from 0 to 750 km. Panel (a) shows the plasma density based on Eq. 4.9, which is used for the entirety of this chapter. Panel (b) shows the y direction velocity based on Eq. 4.10, where $V_0^s = -500$ m/s, $L_V^s = 34$ km, and $d = 110$ km. This profile is used to study the effect of changing d on instability development. Panel (c) shows the y direction velocity based on Eq. 4.11, where $V_0^t = -1000$ m/s, $L_V^t = 10$ km, and $w = 175$ km. This profile is used to study the effects of changing L_V^t and w on instability development. Note that d and w are chosen based on Figure 4.3.

Table 4.1 lists the variables used in the simulations based on Figure 4.3 and nominal ionospheric values at 300 km in altitude. The nominal values are found using the IRI [Bilitza, 2018], MSIS [Picone et al., 2002], and IGRF models [Thébault et al., 2015] for the date of May 2, 2013 at a local time of 20:00, a latitude of 57.5° , and an altitude of 300 km. When sweeping across the planet in longitude, the resulting values are approximately what is provided in Table 4.1. The meridional neutral wind, u_x , is chosen to be at the higher end of observed values [Buonsanto, 1999] to induce faster GDI growth. The zonal neutral wind, u_y , is chosen based on observed values [Zhang et al., 2015] corroborated with simulations [Zhang et al., 2021].

Table 4.1: Nominal simulation parameters for modeling SAPS. The values from n_0 to B are obtained from a combination of the IRI [Bilitza, 2018], NRLMSISE-00 [Picone et al., 2002], and IGRF models [Thébault et al., 2015] at an altitude of 300 km. The neutral wind is within the range of observed values [Buonsanto, 1999]. The values from x_{NE} to w are determined by Figure 4.3, the GPS TEC data, and the SuperDARN radar data. The numerical diffusion constant, D , mitigates numerical error (see Eq. 3.97) and k_x and k_y provide adequate spatial resolution.

Variable	Value
n_0	10^{11} m^{-3}
$T_i = T_e = T$	1000 K
Ion Species	O^+
Neutral Species	O
n_n	10^{14} m^{-3}
B	$5 \times 10^{-5} \text{ T}$
u_x	-500 m/s
u_y	0 or -300 m/s
x_{NE}	220 km
x_{NP}	525 km
L_{NE}	50 km
L_{NP}	75 km
a_1	-0.30
a_2	0.75
c	2.05
V_0^s	-500 m/s
V_0^t	-1000 m/s
L_V^s	34 km
L_V^t	10 or 34 km
d	75 to 175 km
w	125 to 305 km
D	10^2 to $8 \times 10^4 \text{ m}^2/\text{s}$
$[k_{x_{max}}, k_{y_{max}}]$	$[2.14, 0.804] \text{ km}^{-1}$

4.3 Results

Table 4.2: A table of the input parameters for a control GDI simulation using Listing C.2 as the input file. This simulates the SAPS-associated density trough without the SAPS velocity.

Parameter	Value
$[L_x, L_y]$	$[1.5 \times 10^3, 1 \times 10^3]$ km
$[n_x, n_y]$	[1024, 256]
D	10^3 m ² /s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
B	$[0, 0, 5 \times 10^{-5}$ T]
u	[-500, 0, 0] m/s
V_0	0 m/s
levels	[1.6, 1, 2.5]
$[x_{g_N}^L, x_{g_N}^R]$	[220, 525] km
$[L_{g_N}^L, L_{g_N}^R]$	[50, 75] km
n_0	10^{11} m ⁻³
T_{i0}	1000 K
T_{e0}	1000 K
n_n	10^{14} m ⁻³
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-6} n_0$
pert Domain	1: Left Half Only
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

A control simulation is run without any background velocity to get a baseline understanding of the problem. The simulation uses Listing C.2 as the input file using parameters from Table 4.2. The resulting GDI evolution is presented in Figure 4.5(a). In this regime, the GDI behaves in a highly collisional manner. The instability “fingers” grow with minimal to no vorticity. The beginnings of either secondary bifurcations or Kelvin-Helmholtz instabilities are seen at the tips of the “fingers”. Figure 4.5(b) shows the comparison of this multimode simulation with a linear growth rate simplified from Eq. 1.15 for this parameter regime. The

growth rate formula is

$$\gamma_{GDI} = \frac{|\mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} - \mathbf{u}|}{L_N^g} \left[\hat{\mathbf{k}} \cdot (\hat{\mathbf{b}} \times \hat{\mathbf{g}}) \right] \left[\hat{\mathbf{k}} \cdot \hat{\mathbf{e}} \right] - Dk^2, \quad (4.12)$$

where L_N^g is the density gradient scale length, $\hat{\mathbf{k}}$ is the wavenumber unit vector, $\hat{\mathbf{b}}$ is the magnetic field unit vector, $\hat{\mathbf{g}}$ is the density gradient unit vector, $\hat{\mathbf{e}}$ is the electric field unit vector, and k is the wavenumber magnitude. There is reasonable agreement between the simulation and the linear theory with at most a 26.5% difference. The difference is primarily in the longer wavelength modes and is due to nonlinear mode coupling, as can be noted by the difference from Figure 3.11.

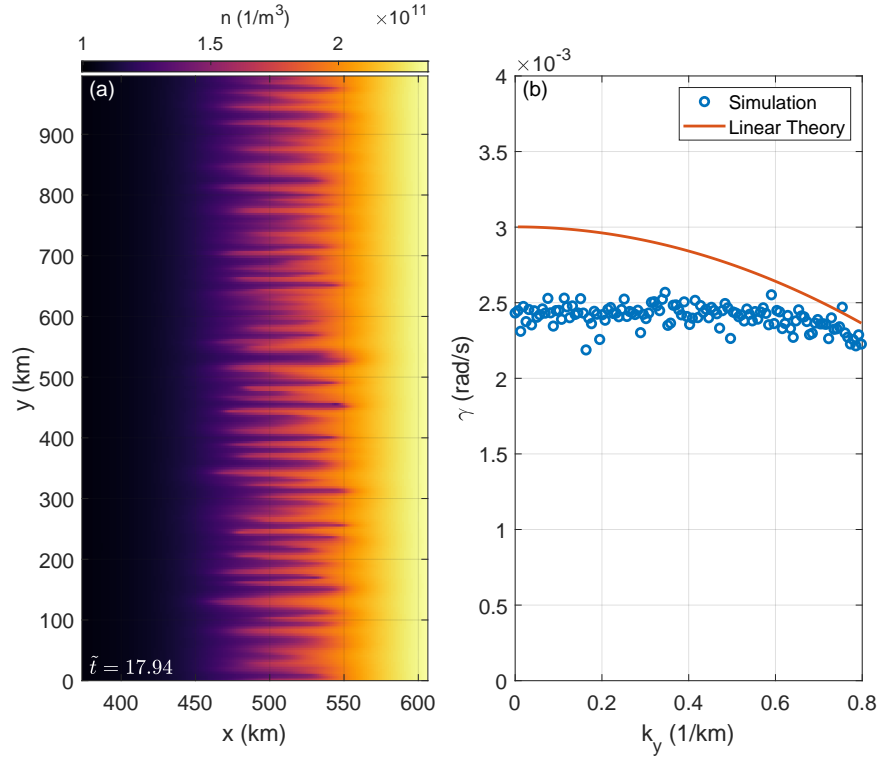


Figure 4.5: Results of a shearless GDI simulation. Panel (a) shows the density evolution. Thin fingers are seen growing into thicker tips which indicate the onset of secondary Kelvin-Helmholtz instabilities or GDI bifurcations. Panel (b) shows the comparison of the instability growth rate between simulation (blue circles) and theory (red line), with reasonable agreement. Note that the differences are due to nonlinear mode coupling.

Note that the plots in this section are shown using times normalized to the maximum growth period of the purely meridional neutral wind driven GDI, i.e., $\mathbf{V}_{\mathbf{E} \times \mathbf{B}_{bg}} = \mathbf{0}$, as per Eq. 4.12. The maximum predicted growth by this equation occurs at $k = 0$ with $\gamma_{\max} = -u_x/L_N^g$. Therefore, the normalized time is $\tilde{t} = |u_x t/L_N^g|$ with one growth period corresponding to

about 333 s. The time scales of visual growth are strongly dependent on the initial perturbation and minimally so on the numerical diffusion used. Since the actual initial perturbations that occur in nature are unknown, the focus should be placed less on the absolute time of the simulations and more on the relative differences in time. In other words, the focus should be on if changing a parameter generally makes growth slower or faster compared to another set of conditions. Additionally, for many of the density plots that are shown, the background velocity profile is represented as a white curve. The location of maximum, purely neutral wind driven growth is denoted by a dashed cyan line. As per Eq. 4.12, this is equivalent to the location of minimum density gradient scale length. Additionally, since the choice of meridional neutral wind direction is equatorward, or south, the interface that is unstable to the GDI is the poleward density gradient. Therefore, in all of the plots shown, only the region around the poleward density gradient is considered. If a poleward wind would have been chosen, the equatorward density gradient would then be unstable to the GDI, which can be seen using Eq. 4.12.

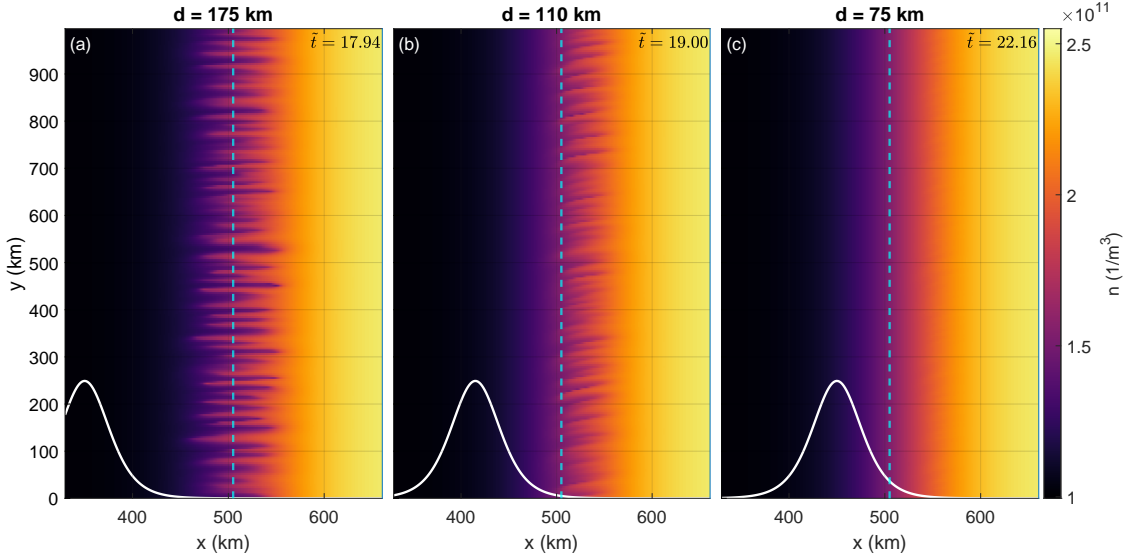


Figure 4.6: Plots of plasma density using Eq. 4.10 for the V_y initial conditions with $V_0^s = -500$ m/s, $L_V^s = 34$ km, and d varying from 75 to 175 km. The solid white curve represents the location of the velocity profile. The dashed cyan line is the location of maximum GDI growth based on Eq. 4.12 with no background velocity. When the velocity is far from the density gradient, as in Panel (a), the GDI grows unimpeded at the location of maximum shear-less GDI growth. When the velocity profile is initialized closer to the density gradient, as in Panels (b) and (c), the GDI is damped and relatively small growth occurs poleward of the maximum shear-less GDI growth location. The plots are at different times to further emphasize the stabilization effect of the velocity shear.

The effect of looking at various SAPS profiles is considered. A set of simulations are run using Listing C.2, which uses Eq. 4.10 for the background velocity. The input parameters

Table 4.3: A table of the input parameters for SAPS profiles using Listing C.2 as the input file and Eq. 4.10 for the background velocity. The parameter d is varied from 75 to 175 km. This is manifested in the input parameters through the different values of $x_{g\phi}^L$.

Parameter	Value
$[L_x, L_y]$	$[1.5 \times 10^3, 1 \times 10^3]$ km
$[n_x, n_y]$	[1024, 256]
D	10^3 m ² /s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
B	$[0, 0, 5 \times 10^{-5}$ T]
u	[-500, 0, 0] m/s
$x_{g\phi}^L$	350, 415, or 450 km
$x_{g\phi}^R$	$1500 - x_{g\phi}^L$ km
$L_{g\phi}$	34 km
V_0	-500 m/s
levels	[1.6, 1, 2.5]
$[x_{gN}^L, x_{gN}^R]$	[220, 525] km
$[L_{gN}^L, L_{gN}^R]$	[50, 75] km
n_0	10^{11} m ⁻³
T_{i0}	1000 K
T_{e0}	1000 K
n_n	10^{14} m ⁻³
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-6} n_0$
pert Domain	1: Left Half Only
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

used are found in Table 4.3, where the parameter d is varied from 75 to 175 km. The resulting evolution of the density is shown in Figure 4.6. In Figure 4.6(a), the SAPS is sufficiently equatorward of the poleward density resulting in an instability evolution nearly, if not entirely, identical to what is shown in Figure 4.5(a), which has no background velocity. Therefore, if the velocity profile is sufficiently far from the most unstable location, nothing is expected to change from the case where there is no velocity profile, which is what intuition would suggest. When the background velocity is set closer to the unstable point, as shown

in Figure 4.6(b), the GDI still grows but at a slower rate. It takes approximately two growth periods longer to reach a similar growth amplitude. This is in agreement with nonlocal linear analyses on GDI growth showing that the velocity shear has a stabilizing effect on GDI growth [Perkins and Doles III, 1975, Huba et al., 1983, Huba and Lee, 1983]. As a result, the location at which the GDI grows in Figure 4.6(b) is further poleward of the anticipated location to a new location of fastest growth. When the velocity profile is initialized even closer to the unstable location, as shown in Figure 4.6(c), the growth becomes even more stabilized with no visual growth seen after many growth periods.

The simulation from Figure 4.6(b) is rerun using a y domain of 500 km with 512 grid points in y . Thus, the smallest resolvable wavelength becomes 195.3 m; this is closer to the spatial scales associated with GPS scintillations and SuperDARN radar scatter. Figure 4.7 shows the power spectra for the perturbed density divided by the total density (a) and the perturbed electric potential (b). The spectra are calculated by taking the Fourier transform of the variable in the y direction and then integrating over the turbulent region in the x direction (see Appendix D.3 for more information on this calculation). Note that only the y direction spectra are considered since the instability growth is dominated by k_y . Both spectra exhibit a turbulence cascade with power laws of about $-5/3$ or -2 , especially within the region of 1 to 10 km^{-1} . These are in agreement with DMSP satellite data [Mishin and Blaunstein, 2008] and nonlinear GDI theory [Keskinen, 1984].

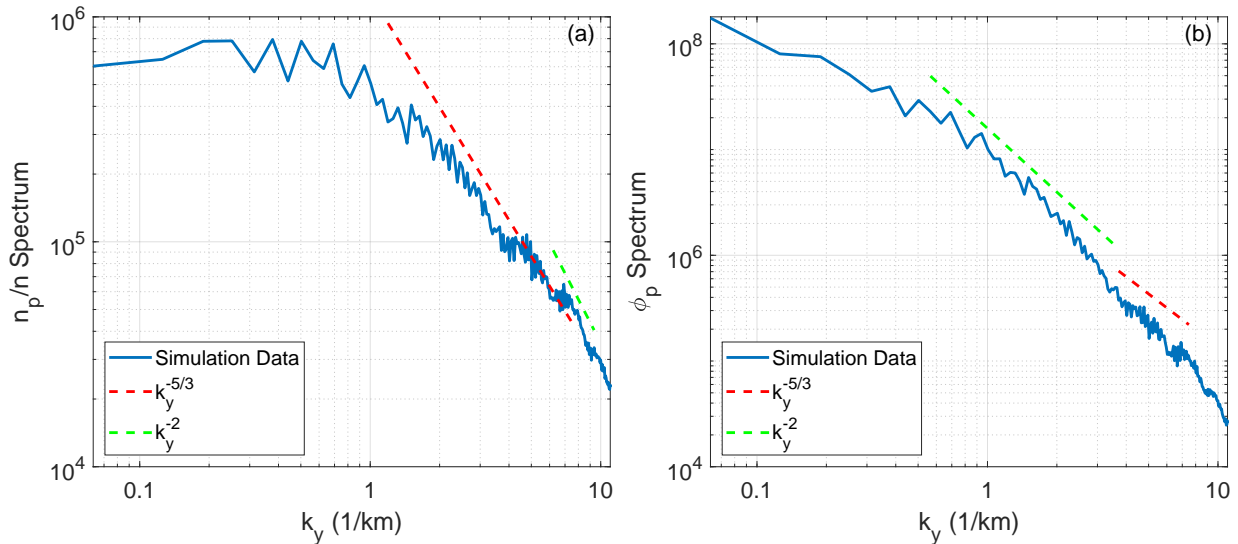


Figure 4.7: A plot of power spectra of the perturbed density divided by the total density and perturbed electric potential at $\tilde{t} = 6.45$ from a higher resolution simulation of Figure 4.6(b). The dashed red and green lines correspond to different power law fits for each spectrum. In the region from 1 to 10 km^{-1} , the powers of $-5/3$ and -2 closely match observed spectra [Mishin and Blaunstein, 2008].

The next logical thought is if the simulations from Figure 4.6 are allowed to run longer,

perhaps the GDI growth will extend into the SAPS region. It could be an explanation for the density irregularities observed inside the SAPS region. The simulations are run again with the input parameters from Table 4.3 using Listing C.2; however, to be able to run longer, the numerical diffusion is increased to $8 \times 10^4 \text{ m}^2/\text{s}$. Figure 4.8 shows that the instability grows and extends in the poleward direction, as expected. However, in the equatorward direction, the velocity sheared region prevents the GDI from extending further. The result is the same for both values of d used. Therefore, the model predicts that if the poleward density gradient is unstable to the GDI, then the GDI will not be able to extend into the SAPS region. But, the GDI that does occur follows observed turbulence cascade power laws.

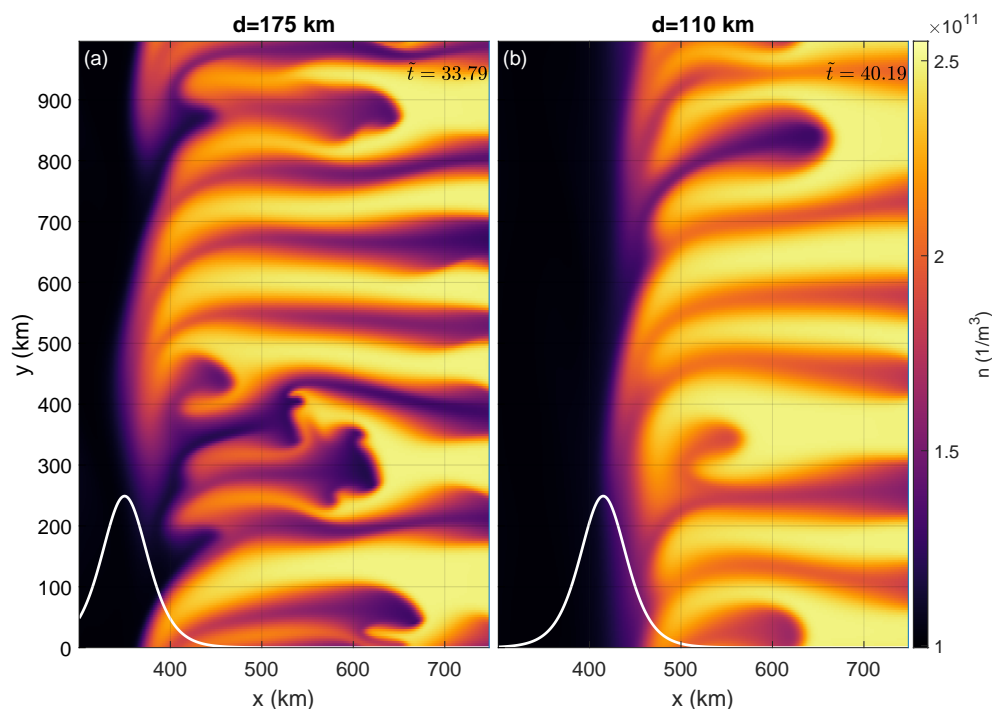


Figure 4.8: Plots of plasma density later in time for Figures 4.6(a) and (b). The velocity shear region acts as a barrier and prevents the GDI from extending further in the equatorward direction.

The observations suggest that the density irregularities also appear within SAPS [Mishin et al., 2003, Mishin and Blaunstein, 2008]. The effect of velocity shear, as seen in Figure 4.8, serves to prevent GDI extension into SAPS if they begin growing outside the SAPS region. It may be possible for the GDI to grow within the SAPS region if there exists a sufficiently large region of low velocity shear. Several simulations are run using Eq. 4.11 for the velocity. The input parameters from Table 4.4 are used with Listing C.3 as the input file.

The effect of changing the velocity gradient scale length is examined by changing the length scaling factor L_V^t . The bottom panels of Figure 4.9 show the 2D density evolution. The top panels show the normalized y -averaged density. Figure 4.9(a) shows that the GDI can

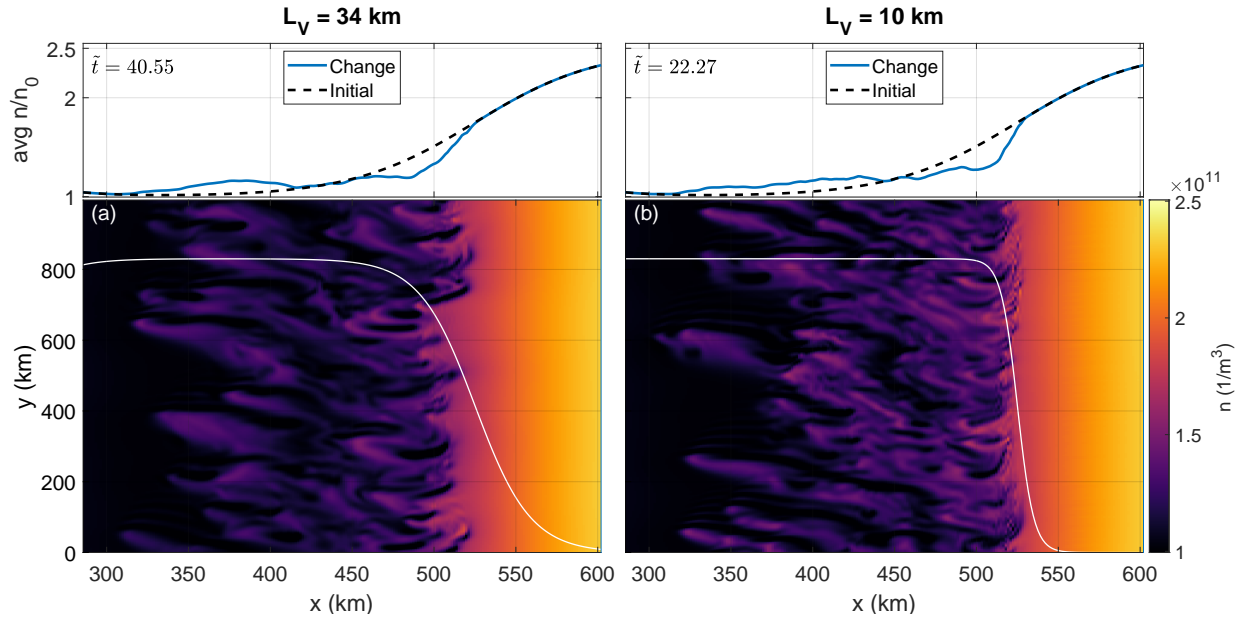


Figure 4.9: The bottom panels show plots of plasma density for a case using Eq. 4.11 for the background, V_y ; the top panels show the y -averaged density normalized by n_0 . The larger velocity gradient scale length causes the GDI to grow much more slowly despite extending to the same general location. Note that due to the placement of the velocity shear, the GDI is growing on the inside of the SAPS region.

grow within the SAPS region for this type of velocity profile due to the placement of the velocity shear. Figure 4.9(b) shows a similar result. The key difference between the two is that the larger velocity gradient scale length causes the GDI to grow much more slowly despite growing to approximately the same amplitude. The larger region of velocity shear has a clear stabilizing effect on the GDI. Therefore, if the velocity profile has low velocity shear in the regions where large GDI growth would otherwise be expected, then the GDI has the ability to grow inside SAPS.

The width of the SAPS channel in Figure 4.9 is on the larger end of what is typically observed in nature. Figure 4.10 shows the results of a set of simulations in which the parameter w is varied. Figure 4.10(a) shows the early time linear growth phase of the GDI, which is similar for all of the cases from Figures 4.10(b-d). Note that the GDI growth location has shifted equatorward due to the choice of velocity profile. Because of this, the GDI is now growing inside the SAPS region. Over time, the instability begins growing both equatorward and poleward. However, it quickly hits the poleward velocity gradient and therefore is prevented from extending further, much like what is seen in Figure 4.8. The GDI continues to extend in the equatorward direction until it reaches the equatorward velocity gradient and is prevented from extending further, as shown in Figures 4.10(b-d). Thus, the density irregularities are confined within the SAPS region due to the existence of the background velocity shear on

Table 4.4: A table of the input parameters for SAPS profiles using Listing C.3 as the input file and Eq. 4.11 for the background velocity. The parameter L_V^t is either 10 or 34 km, which is shown by $L_{g_\phi}^{L,R}$ in the input file. Note that the 10 km value is used unless otherwise specified. The parameter w is varied from 125 to 305 km. This is manifested in the input parameters through the different values of $x_{g_\phi}^L$. The zonal neutral wind, u_y , is always 0 unless otherwise specified.

Parameter	Value
$[L_x, L_y]$	$[1.5 \times 10^3, 1 \times 10^3]$ km
$[n_x, n_y]$	[1024, 256]
D	10^3 m ² /s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
B	$[0, 0, 5 \times 10^{-5}$ T]
u	[-500, 0 or -300, 0] m/s
$x_{g_\phi}^L$	220, 350, or 400 km
$x_{g_\phi}^R$	525 km
$[L_{g_\phi}^L, L_{g_\phi}^R]$	[10, 10] or [34, 34] km
V_0	-1000 m/s
levels	[1.6, 1, 2.5]
$[x_{g_N}^L, x_{g_N}^R]$	[220, 525] km
$[L_{g_N}^L, L_{g_N}^R]$	[50, 75] km
n_0	10^{11} m ⁻³
T_{i0}	1000 K
T_{e0}	1000 K
n_n	10^{14} m ⁻³
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-6} n_0$
pert Domain	1: Left Half Only
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

both sides. Note how in all of the averaged densities, there exists a region of approximately constant density that develops within the SAPS region. Additionally, in Figure 4.10(c), smaller instabilities poleward of the poleward velocity gradient can be seen growing. These are the same kinds of instabilities that appeared in Figure 4.6. They begin growing outside

of the SAPS region and are able to readily extend away from the SAPS region, but cannot enter it.

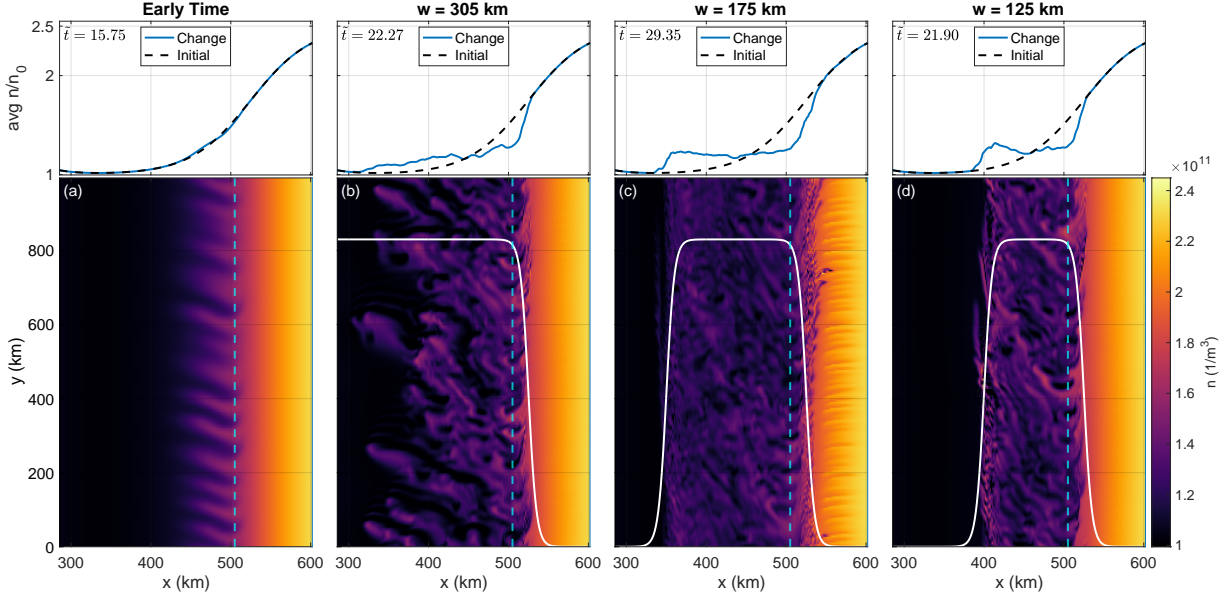


Figure 4.10: The bottom panels show plots of plasma density using Eq. 4.11 for the V_y initial conditions with $V_0^t = -1000$ m/s, $L_V^t = 10$ km, and w varying from 125 to 305 km; the top panels show the y averaged density normalized by n_0 . The white curve represents the velocity profile and the dashed cyan line is the location (same as shown in Figure 4.6) of maximum purely neutral wind driven GDI growth. Panel (a) shows the early time evolution of the GDI, which is similar for all three of the cases shown in Panels (b-d). The initial GDI growth, shown in Panel (a), occurs equatorward of the cyan line and continues to grow in both directions. It is then prevented from extending further. This occurs first in the poleward direction and then in the equatorward direction by the velocity shear, as shown in Panels (b-d). The result is a highly structured SAPS region. The turbulent mixing results in the average density appearing to have an effect that creates another region of approximately constant density. The different times are chosen to capture the late stage effects of the different turbulence evolution.

Figure 4.11 shows the power spectra of the late stage turbulence development of Figures 4.10(b-d). The normalized density shows a steeper decay following a power law of about -6 for all three of the cases. Similarly, the perturbed electric potential also decays more quickly, but at a different power law of about -8. Therefore, the different velocity profile greatly affects the turbulence cascade. Two possible explanations for this are that the velocity shear is damping the higher order modes or that the background motion of the plasma in the y direction significantly impacts the turbulence cascade.

Additional simulations are run using the same conditions as those of Figures 4.10(b-d), but with the addition of a zonal neutral wind component of -300 m/s. The results are shown in

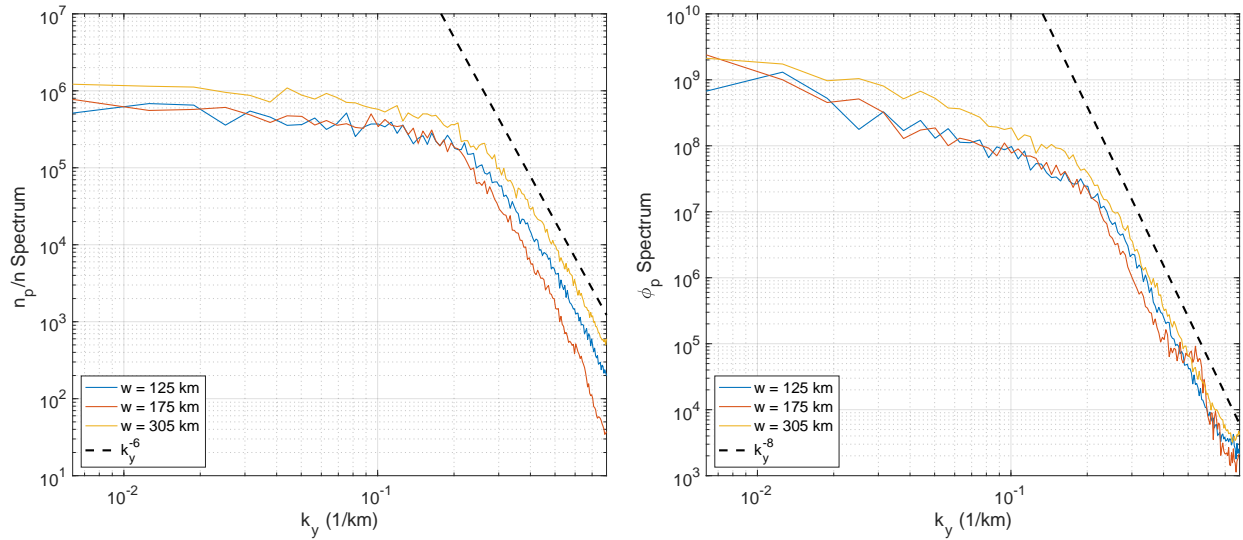


Figure 4.11: A plot of power spectra of the perturbed density divided by the total density and the perturbed electric potential at late time for all three cases from Figure 4.10 in the turbulent region away from the velocity shear. The blue, red, and yellow lines represent the spectra for channel widths of 125 km, 175 km, and 305 km respectively. The dashed black line represents the slope of the turbulence cascade. The y direction spectra decay at a faster rate than those from Figure 4.7, suggesting that the velocity shear is impacting the GDI turbulence cascade.

Figure 4.12. The general behavior is the same as that from Figures 4.10(b-d). The GDI grows inside the SAPS region and remains confined there due to the velocity shear on either side. The main difference is that higher wavenumber modes grow more quickly and are dominating the turbulence development. It has been shown that velocity shear preferentially stabilizes higher wavenumber modes [Perkins and Doles III, 1975, Huba et al., 1983, Huba and Lee, 1983]. By including a zonal neutral wind in the same direction as the background SAPS velocity, the difference in the velocity that defines the velocity shear is smaller. Therefore, the shear does not impact as many of the shorter wavelength modes and they are allowed to grow further. The turbulence cascade for these irregularities is shown in Figure 4.13. They follow the same power laws as Figure 4.11. A small difference is that the power law decay occurs at a slightly higher wavenumber. Thus, the zonal neutral wind plays a minimal role in the turbulence cascade and the ability for the GDI to grow within the SAPS region. The zonal neutral wind's main contribution is to decrease the effective velocity shear to allow higher wavenumber modes to grow.

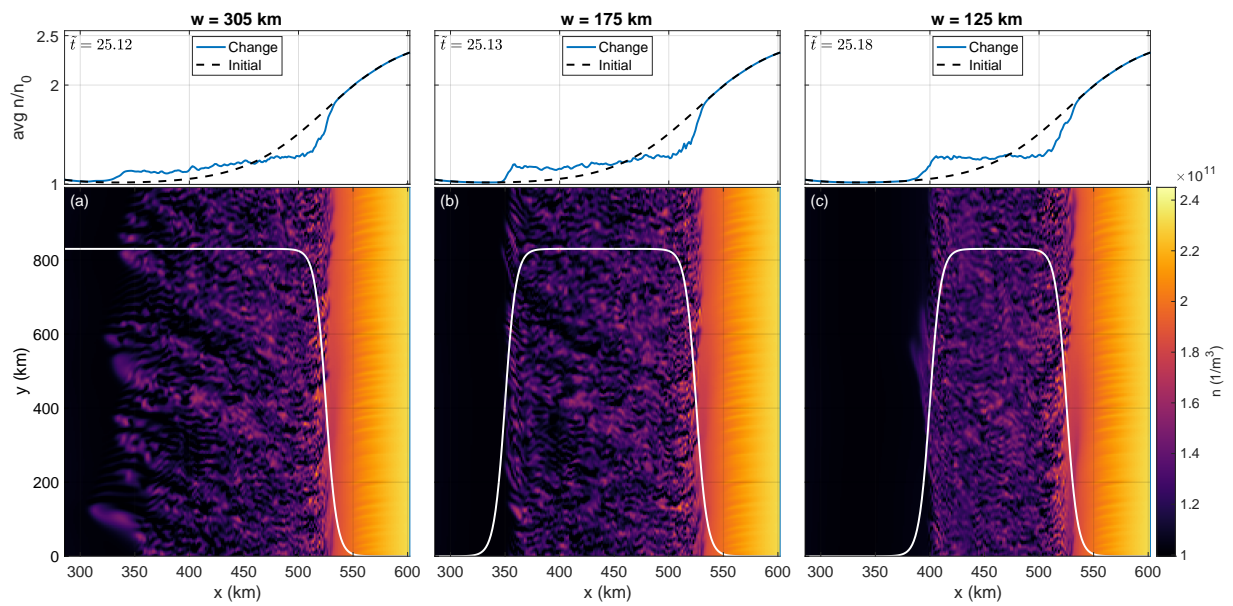


Figure 4.12: The bottom panels show plots of the plasma density for the same simulations as in Figures 4.10(b-d), but with an additional zonal component of neutral wind of -300 m/s; the top panels show the y -averaged normalized density profiles. The resulting behavior is effectively the same as what is observed in Figures 4.10(b-d), with the GDI growing and being confined inside the SAPS region. The key difference is that, visually, the turbulent structure is comprised of higher wavenumber modes.

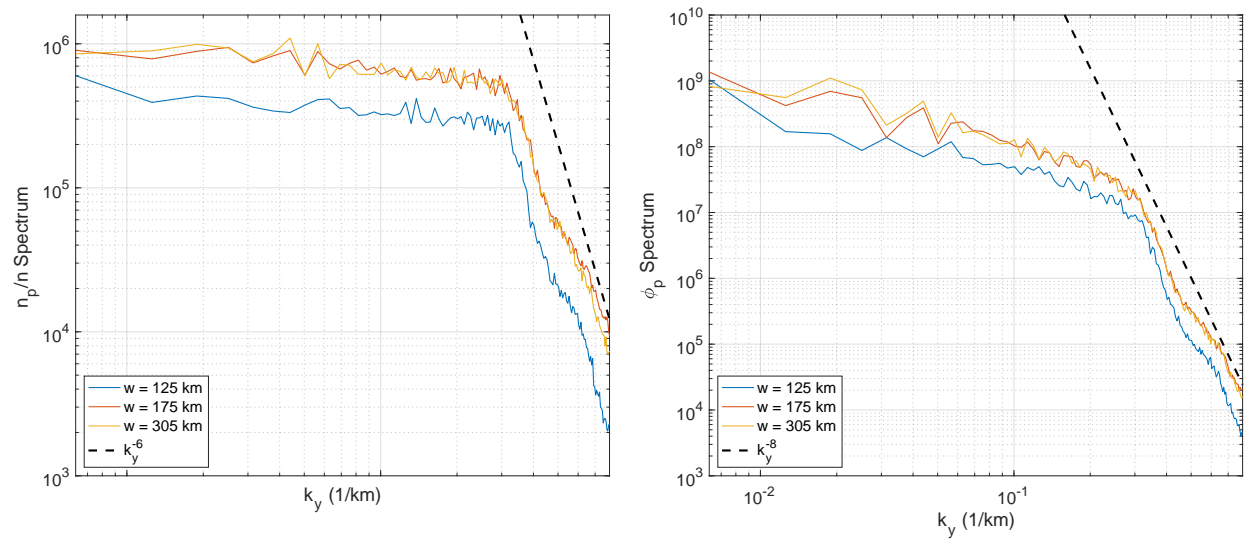


Figure 4.13: A plot of power spectra of the perturbed density divided by the total density and the perturbed electric potential at late time for all three cases from Figure 4.12 in the turbulent region away from the velocity shear. The blue, red, and yellow lines represent the spectra for channel widths of 125 km, 175 km, and 305 km, respectively. The dashed black line represents the slope of the turbulence cascade. Both the density and electric potential follow the same power laws as their equivalent cases in Figure 4.11, showing that the zonal neutral wind plays a minimal role in the turbulence cascade.

4.4 Summary

The shape of the SAPS velocity profile has a strong effect on how the GDI may or may not develop. Because the velocity shear acts as a damping mechanism for higher wavenumber modes of the GDI [Perkins and Doles III, 1975, Huba et al., 1983, Huba and Lee, 1983], it is found that velocity profiles with large regions of velocity shear are not conducive to GDI growth. This is shown by Figure 4.9(a), where the large region of velocity shear prevents the GDI from growing. For velocity profiles that do not have regions of low velocity shear, such as a hyperbolic secant squared profile, e.g., Eq. 4.10, the GDI can also be stabilized, as shown in Figure 4.6(c). The entire profile is basically a sheared velocity region and therefore the GDI is stabilized. For cases where the velocity profile is far enough away from the unstable growth location, the GDI still grows, as in Figure 4.6(a). However, if the velocity profile exists somewhere in the middle, as in Figure 4.6(b), the GDI grows more slowly and no longer at the most neutral wind driven unstable location. The stabilization effect of the velocity shear causes a new location to be the most unstable point. Even when no longer in the linear growth phase, the velocity shear stabilizes GDI growth and prevents extension through the velocity sheared region, as seen in Figure 4.8. Therefore, if any GDI caused density irregularities begin growing outside of the SAPS region, they will not be able to extend into the SAPS region.

For instability growth of this nature, the turbulence cascade, shown in Figure 4.7, agrees well with observations from DMSP satellites. Figure 3 from Mishin and Blaunstein [2008] shows a normalized density power spectrum following a power law of about $-5/3$ or -2 in the wavenumber regime between 1 and 10 km^{-1} , which is consistent with theory [Keskinen, 1984]. Therefore, the density irregularities which have been observed could be viably caused by the GDI, but would initially grow and then remain outside of the SAPS region.

For velocity profiles with large spatial regions of low velocity shear, such as hyperbolic tangent functions, e.g., Eq. 4.11, the GDI is observed to grow inside the SAPS region, as shown in Figure 4.10. The GDI cannot extend through the velocity sheared regions, similar to what is shown in Figure 4.8. Therefore, the turbulence is confined to within the SAPS region. The turbulence cascade is shown in Figure 4.11 and is found to follow a different set of power laws than those of Figure 4.7. Therefore, the power spectra from observations can also provide insight into the general shape of the SAPS profile. The effect of a zonal neutral wind does not change the general nature of if and where the GDI grows, as seen in Figure 4.12. However, it will affect the effective velocity shear. In the case of Figure 4.12, the velocity shear is decreased and therefore more higher wavenumber modes are seen growing. This is further shown by Figure 4.13, where the turbulence cascade starts at a slightly higher wavenumber. Theory predicts that if the zonal neutral wind is in the eastward direction, the effective shear would increase, causing even lower wavenumber modes to dominate the structure. The effect of the neutral wind, however, has no bearing on the turbulence cascade power laws, as evidenced by Figure 4.13.

For all of the simulations run, the meridional neutral wind is in the equatorward direction. Therefore, as per Eq. 4.12, the poleward density gradient is unstable to the GDI. If it was instead in the poleward direction, as was measured by Zhang et al. [2015] for the March 17, 2013 SAPS event, the equatorward density gradient would be unstable to the GDI.

The GDI is considered a viable candidate for turbulence generation in SAPS. The results indicate that any density irregularities that develop cannot cross the velocity sheared region. Therefore, irregularities either remain outside the SAPS region or confined within the SAPS region. Because of this, if density irregularities are observed on both side of the velocity sheared region, then there are likely multiple sources causing them. Furthermore, observations of the power laws of a turbulence cascade can provide insight into the broader SAPS velocity profile. The location of maximum GDI growth is truly a function of the density profile, velocity profile, and neutral wind direction. Therefore, the location, latitudinally, at which density irregularities are observed will shed light as to their cause. For example, since the GDI is not expected to grow in regions of high velocity shear, if density irregularities are observed in such regions, then the cause is likely not the GDI. Future work constitutes creating a predictive model to determine the latitudinal location of maximum GDI growth.

Chapter 5

Dominance of the GDI or KHI in Sheared $\mathbf{E} \times \mathbf{B}$ Flows

5.1 Motivation

Section 1.4 discusses the Kelvin-Helmholtz instability (KHI), which occurs at locations of high velocity shear. The results from Chapter 4 show that the GDI becomes damped by velocity shear. Furthermore, the KHI does not develop at the high velocity shear interfaces in Figures 4.10 and 4.12. However, there are similar background velocity conditions, such as the benchmark problems shown in Figures 3.13 and 3.14, in which the KHI grows.

This chapter examines several parameters to better understand the regions in which the GDI or the KHI may be the dominant instability. The parameters considered are the collisionality, the magnitude of the velocity shear, the velocity shear location, and the neutral wind. The importance of this analysis is to better understand instability development within the SAPS framework (see Chapter 4) in different parameter regimes.

5.2 Optimal GDI Growth Direction

A useful diagnostic tool to aid in differentiating between the GDI and the KHI is to determine the optimal direction of GDI growth. From a simulation perspective, this will allow predictive capabilities about the instability development. From an observation perspective, this can be seen as an inverse problem to obtain some unknown knowledge of the larger background structure.

For much of the previous simulation work in the literature regarding the F region GDI, the driving electric field is perpendicular to the density gradient such that $\mathbf{V}_{\mathbf{E} \times \mathbf{B}} \parallel \nabla n$. This

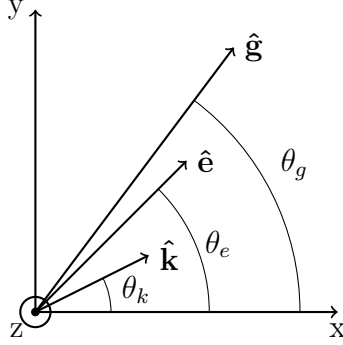


Figure 5.1: A diagram of unit vector definitions for $\hat{\mathbf{k}}$, $\hat{\mathbf{e}}$, and $\hat{\mathbf{g}}$ in the xy plane. Their angles, θ_k , θ_e , and θ_g , respectively, are defined as positive counter clockwise from the x axis. Note that this diagram only serves to define the angles and is not to scale by magnitude.

results in a growing GDI purely in the direction of the electric field vector.

The background electric field for SAPS, discussed in Chapter 4, is defined according to the geometry in Figure 4.1 in both $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$. In this case, the fastest growing mode is no longer in the direction of the electric field vector; there is instead some angular dependence. Recall that Eq. 1.15 is a growth rate formulation for a GDI with arbitrary geometry for the density gradient, magnetic field, electric field, and wavenumber [Makarevich, 2014]. Assuming that ψ and η are small, the equation simplifies to

$$\gamma = \frac{V_{Tot}}{L_N} \hat{\mathbf{k}} \cdot (\hat{\mathbf{b}} \times \hat{\mathbf{g}}) \left[\hat{\mathbf{k}} \cdot \hat{\mathbf{e}} - R \hat{\mathbf{k}} \cdot (\hat{\mathbf{e}} \times \hat{\mathbf{b}}) \right], \quad (5.1)$$

where V_{Tot} is the magnitude of total electric field driven velocity, L_N is the magnitude of the density gradient scale length, $\hat{\mathbf{g}}$ is the density gradient unit vector, $\hat{\mathbf{k}}$ is the wavenumber unit vector, $\hat{\mathbf{e}}$ is the electric field unit vector, $\hat{\mathbf{b}}$ is the magnetic field unit vector, and

$$R = \frac{\nu_{in}}{\Omega_{ci}} - \frac{\nu_{en}}{\Omega_{ce}}, \quad (5.2)$$

where ν is a collision frequency and Ω is a gyrofrequency. Eq. 5.1 is applied to the GDI in a slab geometry such that the magnetic field is purely in the $\hat{\mathbf{z}}$ direction. The wavenumber, electric field, and density gradient directions are allowed to be arbitrary vectors in the xy plane with their angles, θ_k , θ_e , and θ_g , respectively, defined as positive counterclockwise from the x axis, as shown in Figure 5.1. Therefore, the vector components of $\hat{\mathbf{k}}$, $\hat{\mathbf{e}}$, and $\hat{\mathbf{g}}$ in terms of θ_k , θ_e , and θ_g are

$$\hat{\mathbf{k}} = \begin{bmatrix} \cos \theta_k \\ \sin \theta_k \\ 0 \end{bmatrix}, \quad \hat{\mathbf{e}} = \begin{bmatrix} \cos \theta_e \\ \sin \theta_e \\ 0 \end{bmatrix}, \quad \hat{\mathbf{g}} = \begin{bmatrix} \cos \theta_g \\ \sin \theta_g \\ 0 \end{bmatrix}. \quad (5.3)$$

Based on these definitions for all of the unit vectors and simplifications using angle summation trigonometric identities, the growth rate formulation from Eq 5.1 becomes

$$\gamma = \underbrace{\frac{V_{Tot}}{L_N}}_{\gamma_0} \underbrace{\sin(\theta_k - \theta_g) \left[\cos(\theta_e - \theta_k) - R \sin(\theta_e - \theta_k) \right]}_f, \quad (5.4)$$

where γ_0 is the base growth rate that is modified by the geometric factor f . A similar result is shown in Eq. 8 from Keskinen and Ossakow [1982], as well as in Eq. 27 from Makarevich [2014] with a fixed $\hat{\mathbf{e}}$.

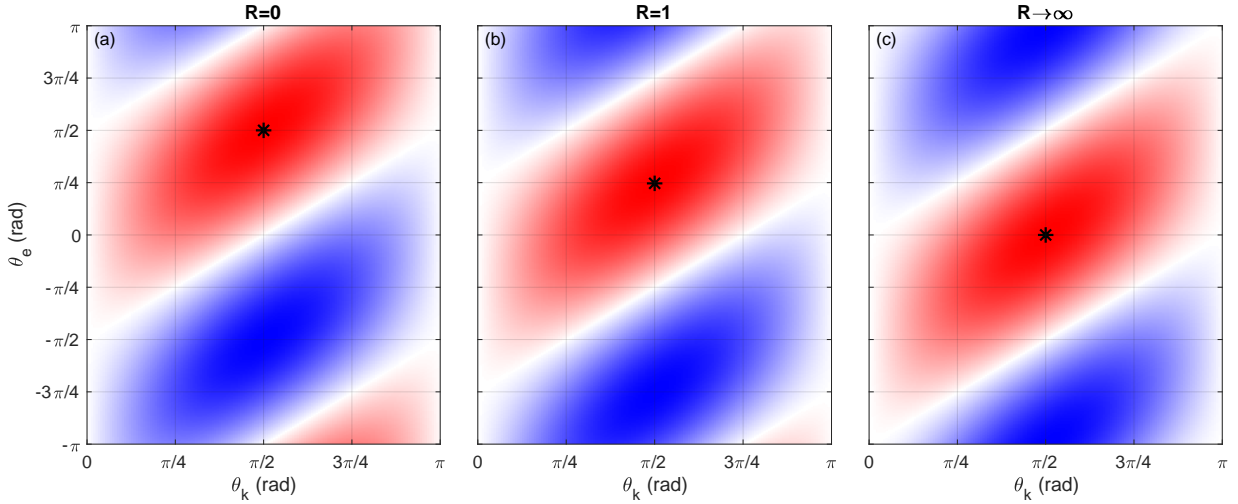


Figure 5.2: A plot of f as a function of θ_k and θ_e for $\theta_g = 0$ and $R = 0, 1$, and 10^6 . Since the goal is to understand which geometries are the most unstable to GDI growth, the magnitude of f is unimportant. This formulation is derived from a dispersion relation; therefore, it is symmetric in k such that $f(k) = f(-k)$. Thus, only one half of the domain in θ_k is shown. The absolute maximum is marked by a black asterisk for each case. In all cases, the maximum occurs at $\theta_k = \pi/2$. For $R = 0$, shown in Panel (a), the maximum occurs at $\theta_e = \pi/2$, i.e., $\hat{\mathbf{g}} \perp \hat{\mathbf{k}} \parallel \hat{\mathbf{e}}$, which is consistent with the F region GDI. For $R \rightarrow \infty$, shown in Panel (c), the maximum occurs at $\theta_e = 0$, i.e., $\hat{\mathbf{g}} \parallel \hat{\mathbf{e}} \perp \hat{\mathbf{k}}$, which is consistent with the E region GDI. The $R = 1$ case, shown in Panel (b), shows that the maximum occurs at $\theta_e = \pi/4$, which is between the two extremes.

Figure 5.2 shows how f behaves as a function of θ_k and θ_e with $\hat{\mathbf{g}} = \hat{\mathbf{x}}$ and $R = 0, 1$, and 10^6 . The 10^6 case approximates the limit as R approaches ∞ . Black asterisks mark the location of maximum f . For all cases, the maximum is at $\theta_k = \pi/2$.

In the $R = 0$ case, which is the F region limit, the maximum occurs at $\theta_e = \pi/2$, or $\hat{\mathbf{g}} \perp \hat{\mathbf{k}} \parallel \hat{\mathbf{e}}$, which is consistent with the F region GDI. In the $R \rightarrow \infty$ case, which is the E region limit, the maximum occurs at $\theta_e = 0$, or $\hat{\mathbf{g}} \parallel \hat{\mathbf{e}} \perp \hat{\mathbf{k}}$, which is consistent with the E region GDI. The $R = 1$ case has a maximum at $\theta_e = \pi/4$, which is between the two extremes.

While understanding how f behaves is useful in studying the general GDI behavior, R , $\hat{\mathbf{e}}$, and θ_g are typically pre-defined based on the initial and background conditions. When initializing a random perturbation, the fastest growing mode dominates. To determine which mode is dominant in k_x and k_y , the maximum of f needs to be found based on a constant θ_e , θ_g , and R . The derivative of f with respect to θ_k , simplified using trigonometric angle summation identities, is

$$\frac{df}{d\theta_k} = \cos(\theta_e + \theta_g - 2\theta_k) - R \sin(\theta_e + \theta_g - 2\theta_k). \quad (5.5)$$

Setting Eq. 5.5 equal to 0 and solving for θ_k yields periodic extrema of the form

$$\theta_{k_{\text{ext}}} = \frac{1}{2} \left(\theta_e + \theta_g - \arctan \frac{1}{R} \right) + l \frac{\pi}{2}, \quad l = 0, \pm 1, \pm 2, \dots, \quad (5.6)$$

where l is an integer describing the periodic nature of the extrema. The choice of l is found using the second derivative to look for the local maxima. The second derivative of f with respect to θ_k is

$$\frac{d^2 f}{d\theta_k^2} = 2 \left[\sin(\theta_e + \theta_g - 2\theta_k) + R \cos(\theta_e + \theta_g - 2\theta_k) \right]. \quad (5.7)$$

To find the local maxima, the condition

$$\left. \frac{d^2 f}{d\theta_k^2} \right|_{\theta_{k_{\text{ext}}}} < 0 \quad (5.8)$$

must be satisfied. First, the argument of the trigonometric functions in Eq. 5.7 is evaluated to be

$$\theta_e + \theta_g - 2\theta_{k_{\text{ext}}} = \theta_e - 2 \left(\frac{1}{2} \right) \left(\theta_e + \theta_g - \arctan \frac{1}{R} + l\pi \right) = \arctan \frac{1}{R} - l\pi = \beta - l\pi, \quad (5.9)$$

where $\beta = \arctan 1/R$. Using the trigonometric identities for phase shifts by $l\pi$,

$$\sin(\beta - l\pi) = (-1)^l \sin \beta \quad (5.10)$$

$$\cos(\beta - l\pi) = (-1)^l \cos \beta, \quad (5.11)$$

Eq. 5.8 becomes

$$\left. \frac{d^2 f}{d\theta_k^2} \right|_{\theta_{k_{\text{max}}}} = 2(-1)^l \left[\sin \beta + R \cos \beta \right] < 0. \quad (5.12)$$

The angle β describes the angle of a right triangle in which its opposite side has a length of 1, its adjacent side has a length of R , and therefore its hypotenuse has a length of $\sqrt{1 + R^2}$. Thus, the sine and cosine terms can be rewritten as

$$\sin \beta = \frac{1}{\sqrt{1 + R^2}} \quad (5.13)$$

$$\cos \beta = \frac{R}{\sqrt{1 + R^2}}. \quad (5.14)$$

Therefore, Eq. 5.12 can be simplified to

$$\left. \frac{d^2 f}{d\theta_k^2} \right|_{\theta_{k_{\max}}} = 2(-1)^l \sqrt{1 + R^2} < 0. \quad (5.15)$$

This inequality is satisfied when l is an odd number. Here the assumption is made that $R > 0$, when defining the triangle described by β . This is a reasonable and valid assumption to make in the ionosphere, but, for mathematical completeness, relaxing this assumption shifts the resulting value of l by 1 if $R < 0$. Therefore, l is even for $R < 0$. This is resolved by adding $(1 - \text{sgn } R)/2$ to l . Therefore, the values of θ_k which yield the maxima of f are

$$\theta_{k_{\max}} = \frac{1}{2} \left(\theta_e + \theta_g - \arctan \frac{1}{R} \right) + \left(l + \frac{1 - \text{sgn } R}{2} \right) \frac{\pi}{2}, \quad l = \pm 1, \pm 3, \pm 5 \dots \quad (5.16)$$

In the simulations, however, θ_e is not defined explicitly, but instead through a combination of the background neutral wind and the background velocity profile. The neutral wind driven electric field is $\mathbf{E}_u = \mathbf{u} \times \mathbf{B}$. The background velocity is defined through an $\mathbf{E} \times \mathbf{B}$ drift. By taking the right cross product of $\mathbf{V}_{\mathbf{E} \times \mathbf{B}}$ with \mathbf{B} , the \mathbf{E} from the velocity can be calculated.

$$\begin{aligned} \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \times \mathbf{B} &= \left(\frac{\mathbf{E} \times \mathbf{B}}{B^2} \right) \times \mathbf{B} \\ &= -\mathbf{E} \frac{B^2}{B^2} \\ \implies \mathbf{E} &= -\mathbf{V}_{\mathbf{E} \times \mathbf{B}} \times \mathbf{B} \end{aligned} \quad (5.17)$$

Thus, the total electric field is

$$\mathbf{E} = \left(\mathbf{u} - \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \right) \times \mathbf{B}. \quad (5.18)$$

If \mathbf{u} and $\mathbf{V}_{\mathbf{E} \times \mathbf{B}}$ are defined as

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \\ 0 \end{bmatrix}, \quad \mathbf{V}_{\mathbf{E} \times \mathbf{B}} = \begin{bmatrix} V_{0x} \\ V_{0y} \\ 0 \end{bmatrix}, \quad (5.19)$$

then

$$\mathbf{E} = B \begin{bmatrix} u_y - V_{0y} \\ V_{0x} - u_x \\ 0 \end{bmatrix}. \quad (5.20)$$

The angle θ_e is defined as

$$\theta_e = \arctan \frac{V_{0x} - u_x}{u_y - V_{0y}} = \text{atan2}(V_{0x} - u_x, u_y - V_{0y}). \quad (5.21)$$

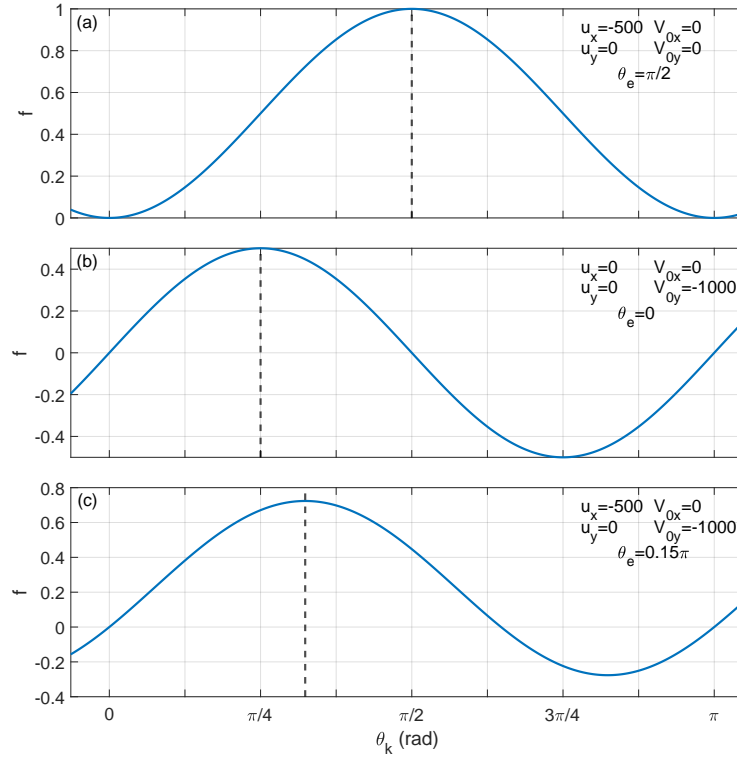


Figure 5.3: Plots of f with $\theta_g = 0$ and $R = 10^{-6}$ for different \mathbf{u} and $\mathbf{V}_{\mathbf{E} \times \mathbf{B}}$ and therefore different θ_e . The dashed line represents the θ_k for maximum growth as predicted by Eq. 5.16. For $\theta_e = \pi/2$, as shown in Panel (a), the maximum occurs at $\theta_k = \pi/2$. For $\theta_e = 0$, as shown in Panel (b), the maximum occurs at $\theta_k = \pi/4$. For θ_e somewhere in between, as shown in Panel (c), the maximum is between the other two cases.

Note that in practice, the atan2 function is used to allow for the electric field to be in an arbitrary direction as opposed to just from $-\pi/2$ to $\pi/2$. An example of a 1D slice of f for a constant θ_e with $\theta_g = 0$ and $R = 10^{-6}$ is shown in Figure 5.3 for different θ_e . When $\theta_e = \pi/2$, the maximum growth direction is $\theta_k = \pi/2$, and when $\theta_e = 0$, the maximum growth direction is $\theta_k = \pi/4$.

Several simulations are run using Listing C.1 with input parameters from Table 5.1 to show that Eq. 5.16 predicts the expected angle of GDI growth for different electric field and density gradient directions. The two density gradient directions considered are $\theta_g = 0$ and $\theta_g = \pi$. The electric field direction is modified in increments of $\pi/4$ from 0 to $7\pi/4$ through the neutral wind.

Figure 5.4 shows the resulting instability development. The times are presented non-dimensionally as $\tilde{t} = u f_{\max} t / L_g^g$ where L_g^g is the minimum density gradient scale length, u is the neutral wind magnitude, and f_{\max} is the larger geometric factor between the left and right density interfaces. The GDI grows as “fingers” extending from the density interfaces in all of plots.

The dashed cyan line indicates the direction of optimal visual growth predicted by Eq. 5.16. Note that the GDI visually appears to grow perpendicular to the wavenumber. Therefore, the dashed cyan lines correspond to the angle of $\theta_{k_{\max}} \pm \pi/2$. In all cases, the GDI matches the dashed cyan lines and extends in the predicted direction. For some panels, the GDI is only seen growing at a particular interface at this time because the geometric factor f is different for each interface; this is because θ_g is different for each interface. Therefore, one side may dominate the other in terms of GDI growth rate magnitude. This is why the GDI appears at the right interface for Figures 5.4(b-d) and at the left interface for Figures 5.4(f-h). Note that Figures 5.4(c) and 5.4(g) are special exceptions in which the geometry, for an F region GDI, is such that $f = 1$ at one interface and $f = 0$ at the other interface.

Keskinen and Ossakow [1982] conducted a similar analysis for one example of an angled electric field showing the GDI tilt in the generally expected direction. The analysis of Figure 5.4 provides a more comprehensive and quantitative validation of the theory.

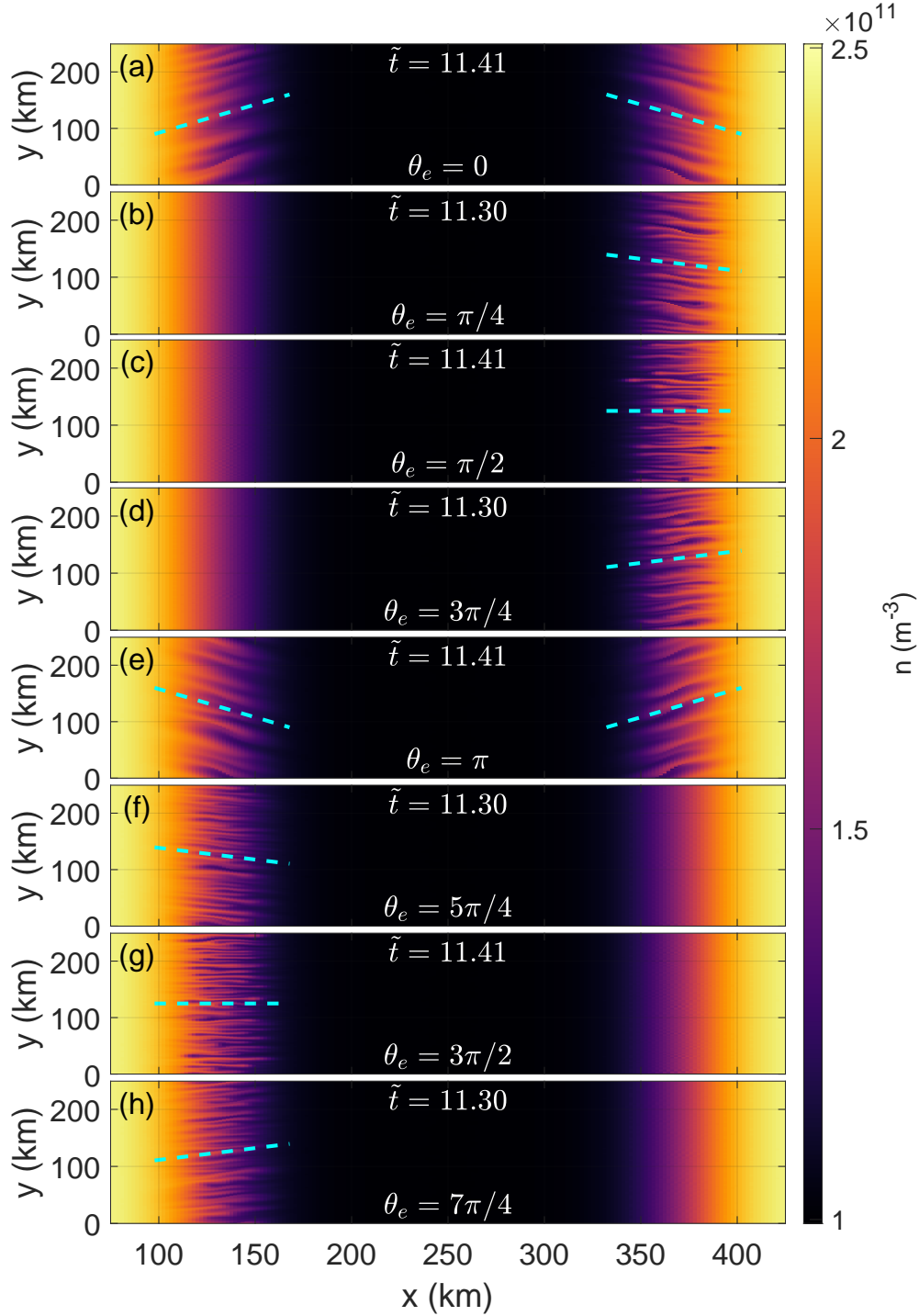


Figure 5.4: Plots of plasma density for different θ_e from 0 to $7\pi/4$. The dashed cyan lines represent the direction of expected visual growth based on Eq. 5.16. Note that the GDI visually grows perpendicular to \mathbf{k} . The GDI extends in the predicted directions for all of the cases. Based on the values of θ_e and θ_g , the magnitude of the growth rate at one of the interfaces may be stronger than at the other interface. This is why the GDI is only visible at the right interface for Panels (b-d) and the left interface for Panels (f-h) for these times.

Table 5.1: A table of the input parameters testing expected growth direction using Listing C.1 as the input file. The electric field direction is modified through the neutral wind. The angle θ_e is varied from 0 to $7\pi/4$ in increments of $\pi/4$.

Parameter	Value
$[L_x, L_y]$	[500, 250] km
$[n_x, n_y]$	[256, 512]
D	10^3 m ² /s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
B	[0, 0, 5×10^{-5} T]
$[u_x, u_y, u_z]$	$500[\cos(\theta_e + \pi/2), \sin(\theta_e + \pi/2), 0]$ m/s
levels	[2.5, 1, 2.5]
$[x_{g_N}^L, x_{g_N}^R]$	[125, 375] km
$[L_{g_N}^L, L_{g_N}^R]$	[30, 30] km
n_0	10^{11} m ⁻³
T_{i0}	1000 K
T_{e0}	1000 K
n_n	10^{14} m ⁻³
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-6} n_0$
pert Domain	0: Entire Domain
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

5.3 Effect of Collisionality

One of the primary differences between Figures 3.12 and 4.10 is collision frequency. These two figures have the same general density and velocity profile structure. For Figure 3.12, the collisions are low and therefore the KHI is the dominating instability. For Figure 4.10, the collisions are sufficiently large and therefore the GDI is the dominating instability. It is then hypothesized that there is a transition between the KHI and the GDI that depends on the collision frequency. One method of ascertaining this is by conducting a non-local linear analysis on the governing equations, which is a highly non-trivial problem. Instead, several parameter sweeps are considered to get an understanding of the instability development in different parameter regimes.

Table 5.2: A table of the input parameters examining instability growth for different collisionalities and background velocities using Listing C.4 as the input file. The collision frequencies are modified by changing the neutral number density. Two different background velocity magnitudes are considered. The higher numerical diffusion is used for the higher velocity.

Parameter	Value
$[L_x, L_y]$	$[1.5 \times 10^3, 1 \times 10^3]$ km
$[n_x, n_y]$	[1024, 256]
D	10^4 or 4×10^4 m ² /s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
\mathbf{B}	$[0, 0, 5 \times 10^{-5}$ T]
\mathbf{u}	[0, 0, 0] m/s
$[x_{g_\phi}^L, x_{g_\phi}^R]$	[220, 700] km
$[L_{g_\phi}^L, L_{g_\phi}^R]$	[10, 10] km
V_0	-1,000 or -2,000 m/s
m	1
o	2.5
$[x_{g_N}^L, x_{g_N}^R]$	[220, 1200] km
$[L_{g_N}^L, L_{g_N}^R]$	[50, 50] km
n_0	10^{11} m ⁻³
T_{i0}	1000 K
T_{e0}	1000 K
n_n	$5 \times 10^{12}, 7 \times 10^{12}, 2 \times 10^{13}, 3 \times 10^{13}, 4 \times 10^{13}, 6 \times 10^{13}$ m ⁻³
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-6} n_0$
pert Domain	1: Left Half of Domain
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

Several simulations are run using Listing C.4 with input parameters from Table 5.2. The effect of collisions is changed by using different neutral number densities, n_n . The parameter R , defined in Eq. 5.2, is used to determine the strength of the collisions, e.g., for high R , the collisional effects are stronger. All of the times are presented non-dimensionally based on the maximum background velocity, V_0 , and the minimum density gradient scale length, L_N^g , as $\tilde{t} = |tV_0/L_N^g|$.

Figure 5.5 shows that the resulting density color plots with $V_0 = -1000$ m/s and R increasing to the right. The white curve represents the velocity profile such that there is a velocity of 0 m/s to the left of the interface and a velocity of -1000 m/s to the right of the interface. The top panels are early in time and the bottom panels are late in time.

Figures 5.5(a-b) show development of the classical KHI vortices at the velocity shear interface. The linear growth rate of Figure 5.5(a) reasonably agrees with Figure 3 from Keskinen et al. [1988]. The amplitude of the KHI vortices from Figure 5.5(b) is smaller than that of Figure 5.5(a) because the higher collision frequency has a damping effect on the KHI [Keskinen et al., 1988]. For both of these cases, it is clear that the KHI is the dominating instability early in time. Figure 5.5(c) shows that the GDI is the dominating instability with the “finger-like” structure growing in the optimal GDI growth direction predicted by Eq. 5.16.

Figure 5.5(d) shows the growth developing into nonlinear KHI vortices undergoing a turbulence cascade but with no indication of the development of the GDI. Figure 5.5(e), however, shows that at late time, the GDI begins to develop to the right of the collisional KHI vortices. Figure 5.5(f) shows the development of the GDI into nonlinearity. The results show that the KHI is the dominating instability at low R and the GDI is the dominating instability at high R . The limits for R for the fully KHI and fully GDI cases, from Figures 5.5(d) and 5.5(f), respectively, are found using a parameter sweep.

The parameter R is shown to have an important role in affecting the dominance of the GDI or the KHI. The velocity, naturally, is the free energy source for the KHI. However, it is also the free energy source for the GDI in this case. Figure 5.5 does not have a neutral wind. Therefore, the GDI that develops in Figure 5.5(c) is from the electric field that drives the background $\mathbf{E} \times \mathbf{B}$ driven velocity profile. Therefore, the velocity plays a role in the growth of both instabilities. To better understand the sensitivity of each instability to the velocity, a set of simulations is run with the same velocity profiles as Figure 5.5 but with $V_0 = -2000$ m/s. Velocities this high are rare in the ionosphere but have been observed to occur in subauroral ion drifts (SAID) [Anderson et al., 1991]. A set of simulations is run using different neutral number densities with parameters from Table 5.2 using Listing C.4 as the input file (note that for this case, $D = 4 \times 10^4$ m²/s).

Figure 5.6 shows the resulting density color plots with R increasing to the right. The white curve indicates the velocity profile. Qualitatively, Figure 5.6 matches the behavior of Figure 5.5. The instabilities are seen to transition from the KHI to the GDI as R increases. The key difference is that the transition point in R is higher for a higher velocity. Thus, a higher shear velocity results in the transition occurring at a higher value of R , which suggests that the KHI is more sensitive to changes in the velocity shear than the GDI is.

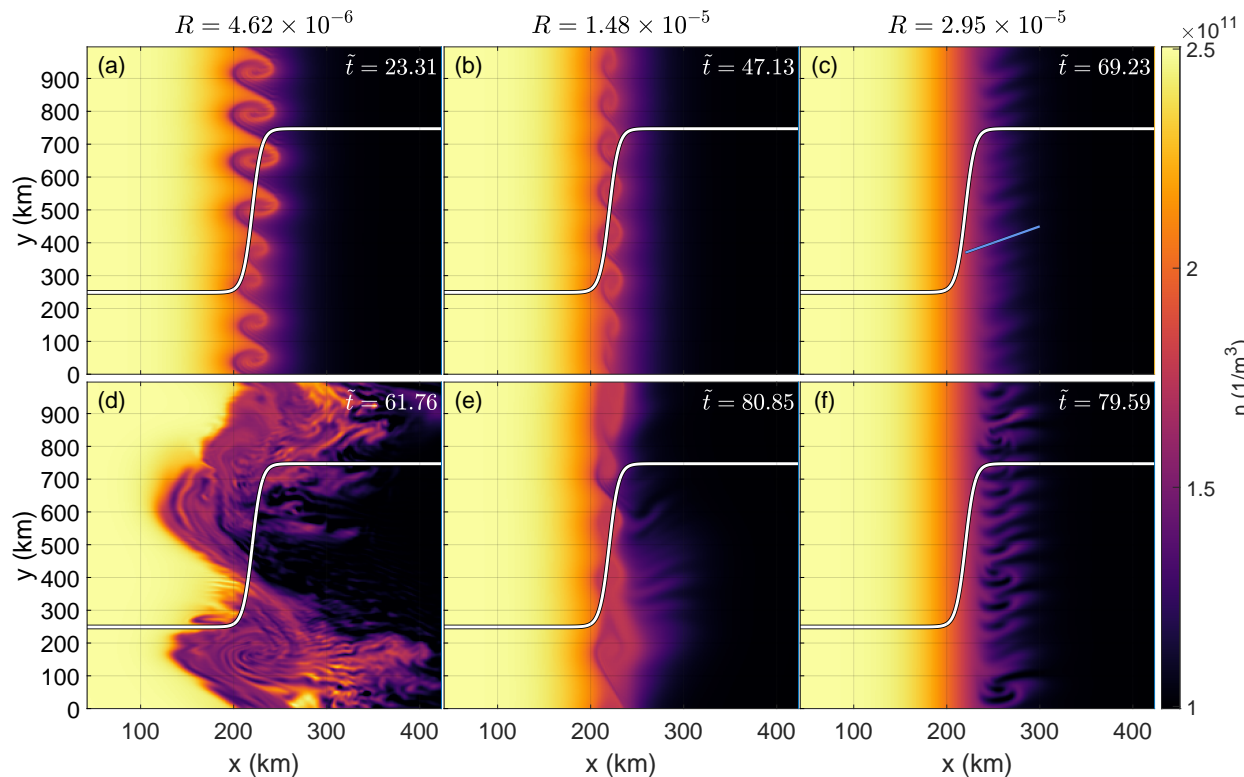


Figure 5.5: Density color plots showing instability development for different collisionalities increasing to the right. The top panels show the early time evolution and the bottom panels show the late stage development. The white curve represents the background velocity profile with $V_0 = -10^3$ m/s such that there is no velocity on the left of the interface and high velocity on the right of the interface. The transition from a pure KHI to a pure GDI is shown as collisionality increases. Note how the vortical structures gradually disappear and how the finger-like structures gradually increase as collisionality increases. The light blue line in Panel (c) indicates the predicted direction of optimal GDI growth based on Eq. 5.16.

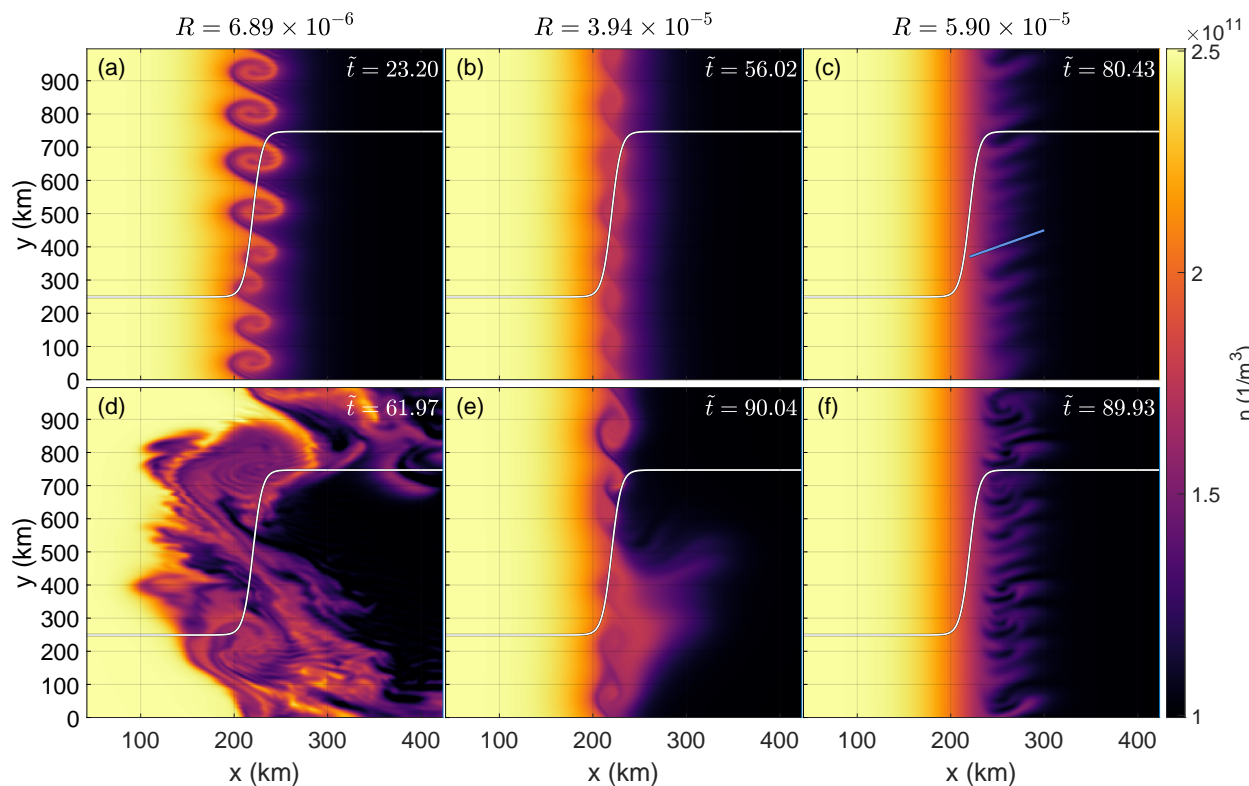


Figure 5.6: Density color plots showing instability development for different collisionalities increasing to the right for a higher velocity case. The white curve represents the background velocity profile with $V_0 = -2 \times 10^3$ m/s. Due to the higher background velocity, the transition from the KHI to the GDI occurs at a higher collisionality than that of Figure 5.5. Thus, the KHI is more sensitive to increases in the velocity than the GDI is. The light blue line in Panel (c) indicates the direction of optimal GDI growth.

5.4 Effect of Velocity Shear Location

The velocity shear that induces the KHI has a stabilizing effect on the GDI [Perkins and Doles III, 1975, Huba et al., 1983, Huba and Lee, 1983]. If the location of the velocity shear is translated away from the location of minimum density gradient scale length, the geometry may be more suitable for GDI growth even in the low collisionality regime. Note that the density gradient scale length location still needs to be within the velocity profile for the GDI to grow. Therefore, in this case, the location of velocity shear should be translated to the left. A set of simulations is run using parameters from Table 5.3 with Listing C.4 as the input file.

Figure 5.7 shows the late time results for two different velocity shear locations with collisionality parameter, $R = 4.62 \times 10^{-6}$, which is equal to that of Figure 5.5(a). The early time growth is similar to the KHI that develops in Figure 5.5(a). The velocity shear location of Figure 5.7(a) is slightly to the left of that of Figure 5.5. This change in the optimal direction is not enough to induce any growth of the GDI. The small structures that extend from the bottom right of the KHI vortex are due to a turbulence cascade. The velocity shear location of Figure 5.7(b) is sufficiently far away from the location of minimum density gradient scale length. Two longer wavelength instabilities are seen growing from the right of the KHI vortices. These structures are in the direction of optimal GDI growth, represented by the light blue lines. Note that the KHI vortex is generally circular. Therefore, the direction of the density gradient scale length can vary from $\pi/2$ to π . Thus, the top blue line uses $\theta_g = 3\pi/4$ and the bottom blue line uses $\theta_g = \pi$; for both lines, $\theta_e = 0$. Therefore, it is possible for the GDI to grow if the velocity shear location is sufficiently far from the minimum density gradient scale length.

Based on the geometry of the density gradients and magnetic field in Figures 5.5 and 5.6, a background neutral wind in the \hat{x} direction would increase the growth rate for the F region GDI, as per Eq. 1.15. Figure 5.8 shows the results using the same parameters as those of Figure 5.7, but with a background \hat{x} direction neutral wind of 500 m/s. The plots in Figure 5.8 are all at approximately the same time as their equivalents in Figure 5.7. However, the amplitudes are larger, suggesting that the neutral wind has an enhancing effect on the KHI growth for each case. As in Figure 5.7(a), the instability development in Figure 5.8(a) is dominated by the KHI. The increased amplitude due to the neutral wind shows the GDI growth in greater detail in Figure 5.8(b) with the blue lines showing the predicted directions. Similar to Figure 5.7(b), the circular geometry of the KHI causes the density gradient direction to vary. Starting from the top line, the density gradient directions are $\theta_g = 3\pi/4$, $\pi/2$, and π . Because of the inclusion of the neutral wind, $\theta_e = -0.4636$. Thus, while the neutral wind has an enhancing effect on both instabilities, it does not play a role in determining the dominance of either of the instabilities.

The GDI that grow from the KHI vortices in Figures 5.7(b) and 5.8(b) appear to be secondary instabilities. This is tested by running the equivalent simulations of Figures 5.7(b) and 5.8(b)

Table 5.3: A table of the input parameters testing if the GDI may develop in the low collisional KHI regime using Listing C.4 as the input file. Two different $\hat{\mathbf{x}}$ direction neutral wind speeds are considered. The velocity shear location is changed through x_{gV}^L .

Parameter	Value
$[L_x, L_y]$	$[1.5 \times 10^3, 1 \times 10^3]$ km
$[n_x, n_y]$	[2048, 512]
D	10^4 or 4×10^4 m ² /s
FourierMeshType	1: Approximation
aliasType	0: No de-aliasing
\mathbf{B}	$[0, 0, 5 \times 10^{-5}$ T]
\mathbf{u}	[0 or 500, 0 or 1000, 0] m/s
x_{gV}^L	150 or 200 km
x_{gV}^R	700 km
$[L_{gN}^L, L_{gN}^R]$	[10, 10] km
V_0	0 or -1000 m/s
m	1
o	2.5
$[x_{gN}^L, x_{gN}^R]$	$[2.2 \times 10^2, 1.2 \times 10^3]$ km
$[L_{gN}^L, L_{gN}^R]$	[50, 50] km
n_0	10^{11} m ⁻³
T_{i0}	1000 K
T_{e0}	1000 K
n_n	5×10^{12} m ⁻³
r_n	152 pm
r_i	152 pm
m_i	16 amu
m_n	16 amu
pertType	4: Random Noise
Amplitude	$\pm 10^{-6} n_0$
pert Domain	1: Left Half of Domain
constCollSwitch	0: Self-Consistent Collision Frequencies
inHeatingSwitch	0: No ion-neutral frictional heating
ieHeatingSwitch	0: No ion-electron frictional heating
condSwitch	0: No thermal conduction

with the equivalent neutral wind driven electric field. Figure 5.9 shows the results with a normalized time of $\tilde{t} = |tu_y/L_N^g|$, which is effectively the same normalization used for Figures 5.7(b) and 5.8(b). It can be shown that the GDI grows for both cases, but at a much slower rate. Therefore, the GDI that grows in Figures 5.7(b) and 5.8(b) is a secondary instability that is induced to grow by the growing primary KHI.

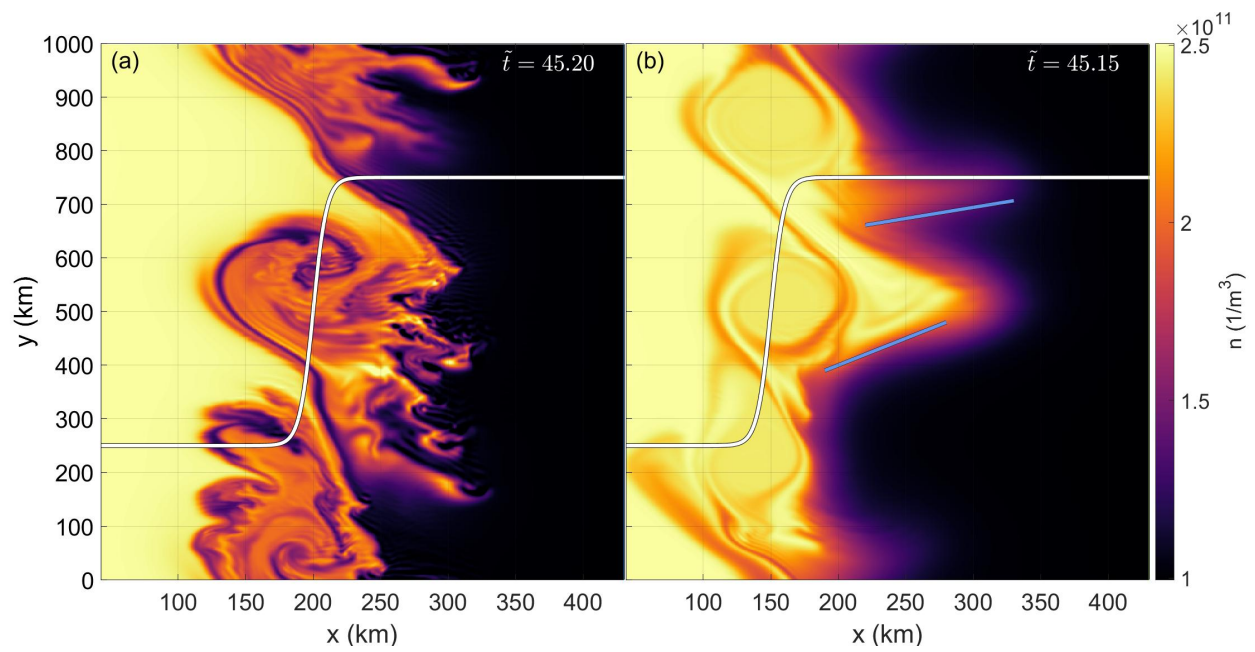


Figure 5.7: Density color plots with different velocity shear locations using $R = 4.62 \times 10^{-6}$. Panel (a) shows the development of the KHI similar to that of Figure 5.5(a). Panel (b) also shows similar vortical KHI development as well as the structures growing to the right that can be attributed to the GDI, which grows in the expected direction (blue lines).

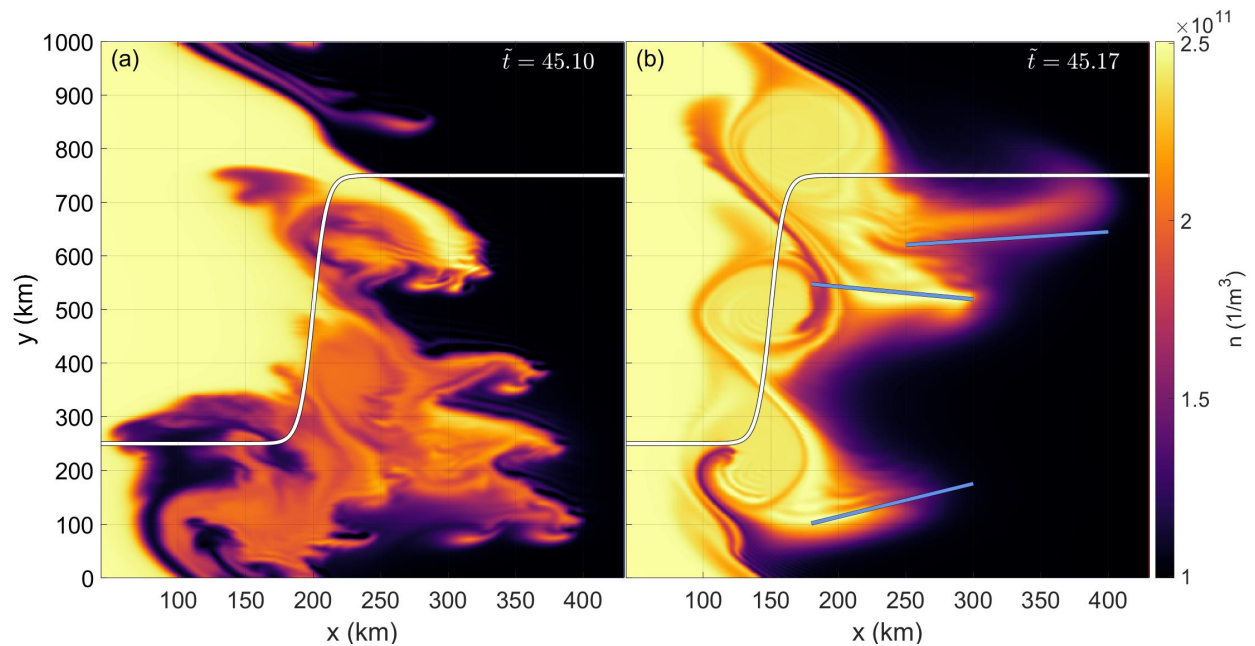


Figure 5.8: Density color plots with different velocity shear locations using $R = 4.92 \times 10^{-6}$ and $u_x = 500$ m/s. All of the panels show the instabilities with a larger amplitude due to the enhancing effect of the background neutral wind. Panel (a) shows only the growth of the KHI in the nonlinear regime. Panel (b) shows the growth of the GDI “fingers” on the right side of the vortical KHI structure. The blue lines indicate the expected directions of GDI development.

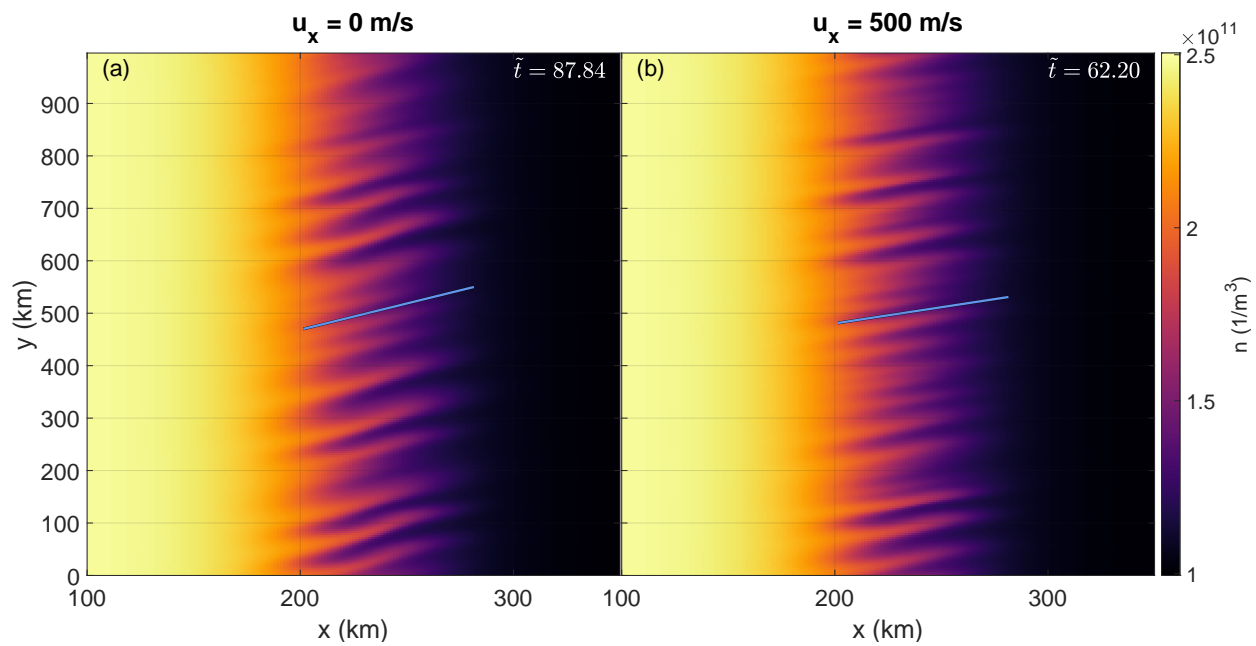


Figure 5.9: Density color plots using the equivalent neutral wind driven electric field to Figures 5.7(b) and 5.8(b). The blue line indicates the direction of optimal GDI growth. The growth of the GDI takes significantly longer than the growth of the GDI shown in Figures 5.7(b) and 5.8(b).

5.5 Altitude Observation Implications

Based on these results, the neutral wind has no impact on the transition between the GDI and the KHI. If the velocity shear is sufficiently far away from the minimum density gradient scale length location, the GDI may grow as a secondary instability in a low collisionality regime. Thus, the main determining factor for the dominance of the GDI or the KHI is the collisionality parameter, R . From a physical perspective, changing the parameter R can be thought of as looking at the ionosphere at a different altitude. Because the neutral density decreases at higher altitudes, the effect of collisions with neutrals and therefore R decreases with increasing altitude. Data from the IRI [Bilitza, 2018], NRLMSISE-00 [Picone et al., 2002], and IGRF [Thébault et al., 2015] are used to calculate R as a function of altitude. The input parameters are May 2, 2013 at 20:00 local time at a geographic latitude and longitude of 57.5° and 40° , respectively. The collision frequencies are calculated using

$$\nu_{\alpha n} = n_n V_{Th_\alpha} \pi (r_\alpha + r_n)^2, \quad (5.22)$$

where n_n is the neutral number density, V_{Th} is the thermal velocity, r is the collision radius, the subscript n denotes neutrals, and the subscript α denotes either the ions or the electrons. The collision radii are assumed to be the Van der Waals radii and are obtained from Bondi [1964] and Batsanov [2001]. Note that the radius for the molecule NO could not be found and is assumed to be approximately that of O_2 . Since the change in collision frequency with altitude is more dependent on the neutral number density and this is intended to be a rough calculation, the results should not change much with this crude approximation. Additionally, the electron radius is assumed to be negligible compared the neutral radii. Instead of accounting for each of the different species of ions and neutrals individually, a weighted average, based on the species's fraction of the total density, is used to obtain an effective species, and therefore effective masses and collision radii. With these values, R can be calculated as a function of altitude.

Figure 5.10 shows the results of this example calculation. As expected, R decreases with increasing altitude. The red dashed line indicates the region above which the KHI dominates. The green dashed line indicates the region below which the GDI dominates. In the intermediate region, both instabilities exist in tandem. Figure 5.10(a) determines the regions using the collisionality parameters from Figures 5.5(a) and 5.5(c). Figure 5.10(b) determines the regions using the collisionality parameters from Figures 5.6(a) and 5.6(c). The increased velocity shear causes the transition to occur at a lower altitude.

Note that Figure 5.10 is just an example and the interpretations are based on the specific parameters chosen in Figures 5.5 and 5.6. It makes the case that, generally, there exist regions in altitude where the KHI and GDI individually dominate and a region where they co-exist and grow together. This is a useful result because SAPS have been measured by SuperDARN radars at altitudes of about 300 km [Chisham et al., 2007] and by DMSP satellites [Mishin and Blaunstein, 2008] at altitudes of about 800 km. Based on Figure 5.10, the satellites would observe the KHI, whereas SuperDARN would observe the GDI despite

looking at the same ionospheric phenomenon¹. In particular, this can be applied to the density irregularities observed in subauroral polarization streams (SAPS).

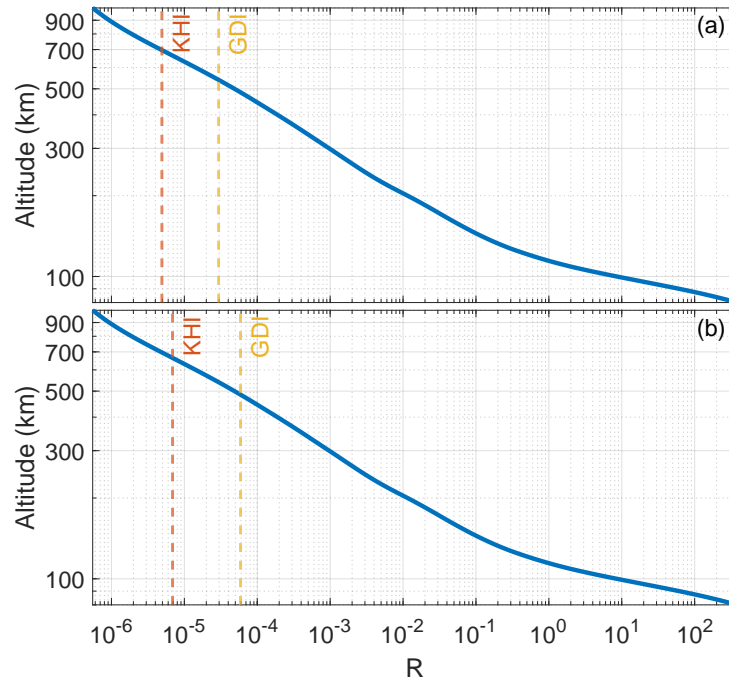


Figure 5.10: An example plot of R as a function of altitude. Data from the IRI [Bilitza, 2018], NRLMSISE-00 [Picone et al., 2002], and IGRF [Thébault et al., 2015] models are used for this calculation. As altitude increases, the value of R decreases. For all altitudes above the red dashed line, the KHI will be the expected dominant instability. For all altitudes below the orange dashed line, the GDI will be the expected dominant instability. For values in the middle, there will be some combination of the two instabilities co-existing. Panel (a) uses collisionality parameters from Figures 5.5(a) and 5.5(c). Panel (b) uses collisionality parameters from Figures 5.6(a) and 5.6(c). The higher velocity pushes the bounds lower in altitude. Note that this is a rough approximation and will change depending on other ionospheric conditions.

As an example, Figures 5.11 and 5.12 show the x and y direction spectra, respectively, for Figures 5.5 and 5.6. The top panels show the spectra of the perturbed density normalized by the total density. The bottom panels show the perturbed potential spectra. The left panels show the low collisionality KHI cases. The middle panels show the mid-collisionality cases that have a combination of the two instabilities. The right panels show the high collisionality GDI cases. Based on Figure 5.10, the left panels are what would be observed by DMSP satellites and the right panels are what would be observed by SuperDARN (if SuperDARN could obtain power spectra). The black and green lines correspond to the KHI

¹Note that the phenomenon needs to have an altitudinal extent large enough to be observed by both types of instruments.

and GDI power laws, respectively. For Figures 5.11 and 5.12, the increase in velocity does not impact the power spectra.

Figure 5.11 shows that the normalized density spectrum changes from a power law of -6 to a power law of -8 as the KHI transitions to the GDI. Similarly, the perturbed potential spectrum power law transitions from -1 to -3. Figures 5.11(b) and 5.11(e) show that, in the middle region, the power law is somewhere in between the two extremes.

Figure 5.12 shows that the normalized density spectrum changes from a power law of -4 to -8 as the KHI transitions to the GDI. The perturbed potential spectrum power law changes from -2 to -9 for the same instability transition. Figures 5.12(b) and 5.12(e) show that the power law exists somewhere in between the two extremes for the middle case. The energy in Figure 5.12(e) may not have run far enough in time to have cascaded to the smallest scales; however, the general trend appears to hold.

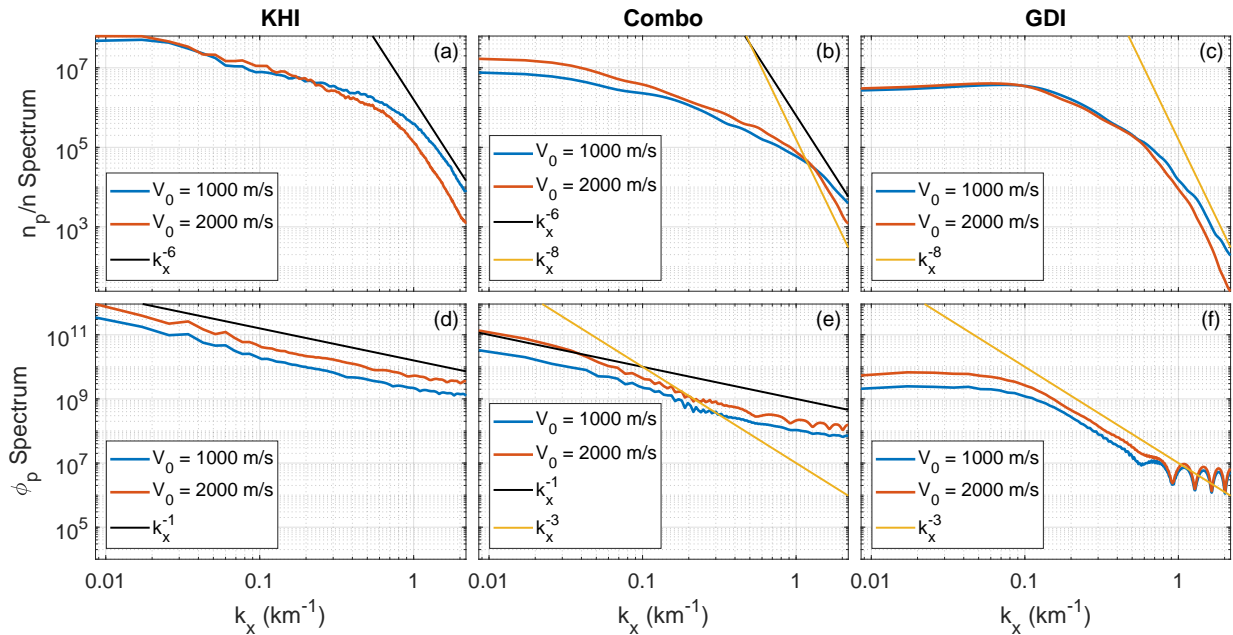


Figure 5.11: The x direction normalized density (top) and perturbed potential (bottom) power spectra for the cases in Figures 5.5 and 5.6. The black and green lines correspond to the KHI and GDI power laws, respectively. The increase in the velocity does not impact the turbulence cascade. For the normalized density spectrum, the power law transitions from -6 to -8 as the KHI transition to the GDI. For the perturbed potential spectrum, the power law transitions from -1 to -3 for the same transition.

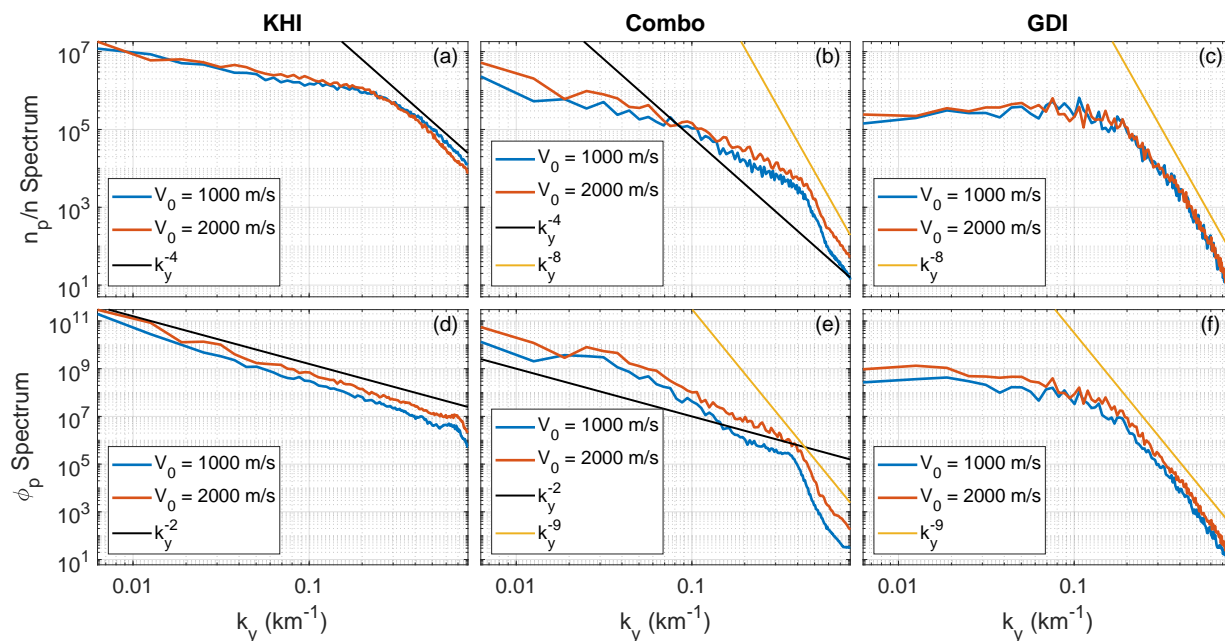


Figure 5.12: The y direction normalized density (top) and perturbed potential (bottom) power spectra for the cases in Figures 5.5 and 5.6. The black and green lines correspond to the KHI and GDI power laws, respectively. The increase in the velocity does not impact the turbulence cascade. For the normalized density spectrum, the power law transitions from -4 to -8 as the KHI transition to the GDI. For the perturbed potential spectrum, the power law transitions from -2 to -9 for the same transition.

Chapter 6

Conclusion

Ionospheric turbulence is important to understand due to its negative impact on communication signals. The novel model developed in this dissertation is motivated by this need. The model is developed for the F region ionosphere to study low frequency electrostatic phenomena using a quasineutral fluid approximation and considers solely the motion perpendicular to the magnetic field. The governing equations are split into background and perturbed terms. Only the perturbed terms are evolved so that the ionospheric turbulence can be better understood. The individual algorithms are verified to work at the formal orders of accuracy. The model is demonstrated to be able to accurately capture the gradient drift instability (GDI) and the Kelvin-Helmholtz instability (KHI) through several benchmark problems. The GDI growth rate is found to tend to the appropriate limit as the wavenumber tends to 0 or ∞ . A consideration of the plasma cloud problem in collisional and inertial regimes provides valuable insight into the importance of the GDI and the KHI in those regimes, respectively.

A brief study on the possibility of GDI development during the August 17, 2017 solar eclipse is presented. The NRL-developed model Sami3 is Also a Model of the Ionosphere (SAMI3) is used to study the ionospheric response to the solar eclipse. An obscuration mask is applied to the EUV radiation flux to model the solar eclipse. Eq. 2.1 is used to calculate the GDI growth rate based on data from the SAMI3 simulation, the Horizontal Wind Model (HWM14), and the International Geomagnetic Reference Field (IGRF) model. It is shown that the growth rate is on the order of 10^{-5} s^{-1} , while the eclipse time rate (think of this as the inverse of a temporal density gradient scale “length”) is on the order of 10^{-4} s^{-1} . Therefore, it is concluded that the GDI is not expected to generate ionospheric turbulence during a solar eclipse. By the time any GDI might grow, the eclipse conditions that would have caused the GDI would have ended and the ionosphere would have returned to nominal conditions [Rathod et al., 2020].

The ionospheric phenomena called subauroral polarization streams (SAPS) are examined using the model described in Chapter 3. SAPS are latitudinally narrow regions in the

ionosphere of large westward plasma flow driven by a poleward electric field. Within SAPS, density irregularities have been observed [Mishin et al., 2003, Mishin and Blaunstein, 2008]. Super Dual Auroral Radar Network (SuperDARN) radar data and Global Positioning System (GPS) scintillation data of 10 different SAPS events are used to provide the background ionospheric conditions. Keeping the background density profile the same, the background velocity profile is changed to understand how the GDI forms under different conditions. It is found that the GDI becomes damped in regions of high velocity shear, which is consistent with nonlocal linear theory for the GDI [Perkins and Doles III, 1975, Huba et al., 1983, Huba and Lee, 1983]. Additionally, the latitudinal location where the GDI grows does not coincide with the expected location predicted by the purely neutral wind driven case. Therefore, the GDI growth location is a function of the density profile, velocity profile, and neutral wind direction. Furthermore, as the GDI grows, it cannot extend through regions of high velocity shear. Therefore, the density irregularities that develop inside (outside) the SAPS remain inside (outside) the SAPS. This suggests that if density irregularities are observed on either side of a velocity sheared region, then they are likely caused by multiple sources. Turbulence generated outside of the SAPS follows a power law of about $-5/3$ or -2 , which is consistent with DMSP satellite observations [Mishin and Blaunstein, 2008] and nonlinear GDI theory [Keskinen, 1984]. Turbulence generated inside the SAPS instead follows power laws of -6 for the normalized density and -8 for the electric potential. Therefore, observations of turbulence spectra can provide insight into the larger overall latitudinal velocity profile.

As per the simulations in Chapter 4, the density irregularities that develop in SAPS propagate at oblique angles. The growth rate from Makarevich [2014] is used to determine the optimal wavenumber direction for GDI growth (Eq. 5.16), which agrees with the simulations.

The velocity shear in SAPS is shown to stabilize the GDI; velocity shear can also cause the KHI to develop. The instability development is studied for varying collisionality. It is found that for high collisionality, as in all of the results in Chapter 4, the GDI is the dominant instability. For low collisionality regimes, the KHI is the dominant instability. For regions in between these extremes, both instabilities exist together and grow at different rates. The KHI is found to be more sensitive to changes in the background velocity than the GDI. Changing collisionality can be seen as analogous to changing altitude. At higher altitudes, the effect of collisions is lower due to a smaller density of neutral particles. Using an example dataset from the IRI, IGRF, and NRLMSISE-00 models, the collisional parameter R is plotted against altitude. It shows that there exists a point in altitude above which the KHI is dominant and another point in altitude below which the GDI is dominant. This predicts that satellites may be measuring irregularities caused by the KHI and that SuperDARN radars may be measuring irregularities caused by the GDI, despite observing the same large scale phenomenon.

Future work should consider an observational study of where the density irregularities occur latitudinally. This will shed light on whether they are caused by the GDI. Additionally a nonlocal linear analysis should be conducted on the governing equations to be able to predict where the maximum growth location for the GDI will occur latitudinally as a function of

the density profile, velocity profile, and neutral wind direction. This same analysis can also determine the importance of collisionality and provide a more accurate transition point between the KHI and the GDI. The variation of SAPS velocity profiles with altitude should be understood to study if density irregularities might occur due to the KHI at higher altitudes.

Bibliography

- Ercha Aa, Philip J Erickson, Shun-Rong Zhang, Shasha Zou, Anthea J Coster, Larisa P Goncharenko, and John C Foster. A statistical study of the subauroral polarization stream over north american sector using the millstone hill incoherent scatter radar 1979–2019 measurements. *Journal of Geophysical Research: Space Physics*, 125(10):e2020JA028584, 2020.
- PC Anderson, RA Heelis, and WB Hanson. The ionospheric signatures of rapid subauroral ion drifts. *Journal of Geophysical Research: Space Physics*, 96(A4):5785–5792, 1991.
- PC Anderson, WB Hanson, RA Heelis, JD Craven, DN Baker, and LA Frank. A proposed production model of rapid subauroral ion drifts and their relationship to substorm evolution. *Journal of Geophysical Research: Space Physics*, 98(A4):6069–6078, 1993.
- EB Armstrong. Observations of luminous clouds produced in the upper atmosphere by exploding grenades—i. atomic emissions. *Planetary and Space Science*, 11(7):733–742, 1963.
- Sunanda Basu, Santimay Basu, CE Valladares, H-C Yeh, S-Y Su, E MacKenzie, PJ Sultan, J Aarons, FJ Rich, P Doherty, et al. Ionospheric effects of major magnetic storms during the international space weather period of september and october 1999: Gps observations, vhf/uhf scintillations, and in situ density structures at middle and equatorial latitudes. *Journal of Geophysical Research: Space Physics*, 106(A12):30389–30413, 2001.
- SS Batsanov. Van der waals radii of elements. *Inorganic materials*, 37(9):871–885, 2001.
- Paul A Bernhardt. Cross-b convection of artificially created, negative-ion clouds and plasma depressions: Low-speed flow. *Journal of Geophysical Research: Space Physics*, 93(A8):8696–8704, 1988.
- Christophe Besse, Jean Claudel, Pierre Degond, Fabrice Deluzet, Gérard Gallice, and Christian Tessieras. Instability of the ionospheric plasma: Modeling and analysis. *SIAM Journal on Applied Mathematics*, 65(6):2178–2198, 2005.
- Christophe Besse, Jean Claudel, Pierre Degond, Fabrice Deluzet, Gérard Gallice, and Christian Tessieras. Numerical simulations of the ionospheric striation model in a non-uniform magnetic field. *Computer physics communications*, 176(2):75–90, 2007.

- Dieter Bilitza. Iri the international standard for the ionosphere. *Advances in Radio Science*, 16, 2018.
- CK Birdsall and AB Langdon. Plasma physics via computer simulation (reprinted), 2005.
- A van Bondi. van der waals volumes and radii. *The Journal of physical chemistry*, 68(3):441–451, 1964.
- KA Browning. Structure of the atmosphere in the vicinity of large-amplitude kelvin-helmholtz billows. *Quarterly Journal of the Royal Meteorological Society*, 97(413):283–299, 1971.
- O Buneman. Excitation of field aligned sound waves by electron streams. *Physical Review Letters*, 10(7):285, 1963.
- M Ji Buonsanto. Ionospheric storms—a review. *Space Science Reviews*, 88(3-4):563–601, 1999.
- Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, A Thomas Jr, et al. *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.
- Subrahmanyan Chandrasekhar. *Hydrodynamic and hydromagnetic stability*. Oxford University Press, 1961.
- PK Chaturvedi and SL Ossakow. Nonlinear stabilization of the $e \times b$ gradient drift instability in ionospheric plasma clouds. *Journal of Geophysical Research: Space Physics*, 84(A2):419–421, 1979.
- Francis F Chen et al. *Introduction to plasma physics and controlled fusion*, volume 1. Springer, 1984.
- G Chimonas. Wind-driven instability in the lower e region. *Journal of Geophysical Research*, 74(16):4091–4098, 1969.
- G Chimonas. Internal gravity-wave motions induced in the earth’s atmosphere by a solar eclipse. *Journal of Geophysical Research*, 75(28):5545–5551, 1970.
- G Chisham, M Lester, SE Milan, MP Freeman, WA Bristow, A Grocott, KA McWilliams, JM Ruohoniemi, TK Yeoman, Peter Lawrence Dyson, et al. A decade of the super dual auroral radar network (superdarn): Scientific achievements, new techniques and future directions. *Surveys in geophysics*, 28(1):33–109, 2007.
- Tong Dang, Binzheng Zhang, Jiuhou Lei, Wenbin Wang, Alan Burns, Han-li Liu, Kevin Pham, and Kareem A Sorathia. Development of high-resolution thermosphere–ionosphere electrodynamics general circulation model (tie-gcm) using ring average technique. *Geoscientific Model Development Discussions*, pages 1–30, 2020.

- JH Doles III, No J Zabusky, and FW Perkins. Deformation and striation of plasma clouds in the ionosphere, 3. numerical simulations of a multilevel model with recombination chemistry. *Journal of Geophysical Research*, 81(34):5987–6004, 1976.
- Douglas P Drob, John T Emmert, John W Meriwether, Jonathan J Makela, Eelco Doornbos, Mark Conde, Gonzalo Hernandez, John Noto, Katherine A Zawdie, Sarah E McDonald, et al. An update to the horizontal wind model (hwm): The quiet time thermosphere. *Earth and Space Science*, 2(7):301–319, 2015.
- Dale R Durran. *Numerical methods for fluid dynamics: With applications to geophysics*, volume 32. Springer Science & Business Media, 2010.
- A Eltrass and WA Scales. Nonlinear evolution of the temperature gradient instability in the midlatitude ionosphere. *Journal of Geophysical Research: Space Physics*, 119(9):7889–7901, 2014.
- A Eltrass, A Mahmoudian, WA Scales, S de Larquier, JM Ruohoniemi, JBH Baker, RA Greenwald, and PJ Erickson. Investigation of the temperature gradient instability as the source of midlatitude quiet time decameter-scale ionospheric irregularities: 2. linear analysis. *Journal of Geophysical Research: Space Physics*, 119(6):4882–4893, 2014.
- PJ Erickson, JC Foster, and JM Holt. Inferred electric field variability in the polarization jet from millstone hill e region coherent scatter observations. *Radio Science*, 37(2):1–15, 2002.
- PJ Erickson, F Beroz, and MZ Miskin. Statistical characterization of the american sector subauroral polarization stream using incoherent scatter radar. *Journal of Geophysical Research: Space Physics*, 116(A5), 2011.
- DT Farley Jr. Two-stream plasma instability as a source of irregularities in the ionosphere. *Physical Review Letters*, 10(7):279, 1963.
- Bela G Fejer and MC Kelley. Ionospheric irregularities. *Reviews of Geophysics*, 18(2):401–454, 1980.
- Banafsheh Ferdousi, Yukitoshi Nishimura, Naomi Maruyama, and Larry R Lyons. Subauroral neutral wind driving and its feedback to saps during the 17 march 2013 geomagnetic storm. *Journal of Geophysical Research: Space Physics*, 124(3):2323–2337, 2019.
- JC Foster and WJ Burke. Saps: A new categorization for sub-auroral electric fields. *Eos, Transactions American Geophysical Union*, 83(36):393–394, 2002.
- JC Foster and HB Vo. Average characteristics and activity dependence of the subauroral polarization stream. *Journal of Geophysical Research: Space Physics*, 107(A12):SIA–16, 2002.

- JC Foster, PJ Erickson, FD Lind, and W Rideout. Millstone hill coherent-scatter radar observations of electric field variability in the sub-auroral polarization stream. *Geophysical research letters*, 31(21), 2004.
- Yu I Galperin. Polarization jet: Characteristics and a model. In *Annales Geophysicae*, volume 20, pages 391–404. Copernicus GmbH, 2002.
- G Ganguli, MJ Keskinen, H Romero, R Heelis, T Moore, and C Pollock. Coupling of microprocesses and macroprocesses due to velocity shear: An application to the low-altitude ionosphere. *Journal of Geophysical Research: Space Physics*, 99(A5):8873–8889, 1994.
- J Goldstein, JL Burch, and BR Sandel. Magnetospheric model of subauroral polarization stream. *Journal of Geophysical Research: Space Physics*, 110(A9), 2005.
- NA Gondarenko and PN Guzdar. Gradient drift instability in high latitude plasma patches: Ion inertial effects. *Geophysical research letters*, 26(22):3345–3348, 1999.
- NA Gondarenko and PN Guzdar. Nonlinear three-dimensional simulations of mesoscale structuring by multiple drives in high-latitude plasma patches. *Journal of Geophysical Research: Space Physics*, 111(A8), 2006.
- RA Greenwald. Diffuse radar aurora and the gradient drift instability. *Journal of Geophysical Research*, 79(31):4807–4810, 1974.
- Raymond A Greenwald, Kjellmar Oksavik, Philip J Erickson, Frank D Lind, J Michael Ruohoniemi, Joseph BH Baker, and Jesper W Gjerloev. Identification of the temperature gradient instability as the source of decameter-scale ionospheric irregularities on plasma-pause field lines. *Geophysical research letters*, 33(18), 2006.
- Gerhard Haerendel. Experiments with plasmas artificially injected into near-earth space. *Frontiers in Astronomy and Space Sciences*, 6:29, 2019.
- Gerhard Haerendel and Reimar Lüst. Artificial plasma clouds in space. *Scientific American*, 219(5):80–95, 1968.
- Gerhard Haerendel, Reimar Lüst, and Erich Rieger. Motion of artificial ion clouds in the upper atmosphere. *Planetary and Space Science*, 15(1):1–18, 1967.
- M Harel, RA Wolf, RW Spiro, PH Reiff, C-K Chen, WJ Burke, FJ Rich, and M Smiddy. Quantitative simulation of a magnetospheric substorm 2. comparison with observations. *Journal of Geophysical Research: Space Physics*, 86(A4):2242–2260, 1981.
- Fei He, Xiao-Xin Zhang, and Bo Chen. Solar cycle, seasonal, and diurnal variations of subauroral ion drifts: Statistical results. *Journal of Geophysical Research: Space Physics*, 119(6):5076–5086, 2014.

- Fei He, Xiao-Xin Zhang, Wenbin Wang, and Weixing Wan. Different evolution patterns of subauroral polarization streams (saps) during intense storms and quiet time substorms. *Geophysical Research Letters*, 44(21):10–796, 2017.
- Fei He, Xiao-Xin Zhang, Wenbin Wang, Libo Liu, Zhi-Peng Ren, Xinan Yue, Lianhuan Hu, Weixing Wan, and Hui Wang. Large-scale structure of subauroral polarization streams during the main phase of a severe geomagnetic storm. *Journal of Geophysical Research: Space Physics*, 123(4):2964–2973, 2018.
- Charles Hirsch. *Numerical Computation of Internal and External Flows: Volume 2: Computational Methods for Inviscid and Viscous Flow*. Wiley & Sons, 1990.
- FC Hoh. Instability of penning-type discharges. *The Physics of Fluids*, 6(8):1184–1191, 1963.
- J Huba and J Krall. Modeling the plasmasphere with sami3. *Geophysical research letters*, 40(1):6–10, 2013.
- JD Huba and D Drob. Sami3 prediction of the impact of the 21 august 2017 total solar eclipse on the ionosphere/plasmasphere system. *Geophysical Research Letters*, 44(12):5928–5935, 2017.
- JD Huba and G Joyce. Global modeling of equatorial plasma bubbles. *Geophysical research letters*, 37(17), 2010.
- JD Huba and ST Zalesak. Long-wavelength limit of the $e \times b$ instability. *Journal of Geophysical Research: Space Physics*, 88(A12):10263–10265, 1983.
- JD Huba, SL Ossakow, P Satyanarayana, and PN Guzdar. Linear theory of the $e \times b$ instability with an inhomogeneous electric field. *Journal of Geophysical Research: Space Physics*, 88(A1):425–434, 1983.
- JD Huba, G Joyce, and JA Fedder. Sami2 is another model of the ionosphere (sami2): A new low-latitude ionosphere model. *Journal of Geophysical Research: Space Physics*, 105(A10):23035–23053, 2000.
- JD Huba, J Krall, and D Drob. Global ionospheric metal ion transport with sami3. *Geophysical Research Letters*, 46(14):7937–7944, 2019.
- Joseph D Huba and LC Lee. Short wavelength stabilization of the gradient drift instability due to velocity shear. *Geophysical research letters*, 10(4):357–360, 1983.
- Mary K Hudson and Michael C Kelley. The temperature gradient drift instability at the equatorward edge of the ionospheric plasma trough. *Journal of Geophysical Research*, 81(22):3913–3918, 1976.
- DL Hysell and E Kudeki. Collisional shear instability in the equatorial f region ionosphere. *Journal of Geophysical Research: Space Physics*, 109(A11), 2004.

- DL Hysell, J Chun, and JL Chau. Bottom-type scattering layers and equatorial spread f. In *Annales Geophysicae*, volume 22, pages 4061–4069, 2004.
- George Kaplan, Jennifer Lynn Bartlett, Alice Monet, John Bangert, Wendy Puatua, William Harris, Amy Fredericks, Eric G Barron, and Paul Barrett. Novas: Naval observatory vector astrometry software. *ascl*, pages ascl-1202, 2012.
- MC Kelley, MF Larsen, C LaHoz, and JP McClure. Gravity wave initiation of equatorial spread f: A case study. *Journal of Geophysical Research: Space Physics*, 86(A11):9087–9100, 1981.
- Michael C Kelley. *The Earth's ionosphere: plasma physics and electrodynamics*. Academic press, 2009.
- Michael C Kelley, KD Baker, and JC Ulwick. Late time barium cloud striations and their possible relationship to equatorial spread f. *Journal of Geophysical Research: Space Physics*, 84(A5):1898–1904, 1979.
- Michael John Keskinen and SL Ossakow. On the spatial power spectrum of the $e \times b$ gradient drift instability in ionospheric plasma clouds. *Journal of Geophysical Research: Space Physics*, 86(A8):6947–6950, 1981.
- MJ Keskinen. Nonlinear theory of the $e \times b$ instability with an inhomogeneous electric field. *Journal of Geophysical Research: Space Physics*, 89(A6):3913–3920, 1984.
- MJ Keskinen. Nonlinear evolution of a narrow stratified velocity-shear layer. *Physics of Plasmas*, 3(4):1259–1262, 1996.
- MJ Keskinen and JD Huba. Nonlinear evolution of high-latitude ionospheric interchange instabilities with scale-size-dependent magnetospheric coupling. *Journal of Geophysical Research: Space Physics*, 95(A9):15157–15166, 1990.
- MJ Keskinen and SL Ossakow. Nonlinear evolution of plasma enhancements in the auroral ionosphere, 1, long wavelength irregularities. *Journal of Geophysical Research: Space Physics*, 87(A1):144–150, 1982.
- MJ Keskinen and SL Ossakow. Nonlinear evolution of convecting plasma enhancements in the auroral ionosphere: 2. small scale irregularities. *Journal of Geophysical Research: Space Physics*, 88(A1):474–482, 1983a.
- MJ Keskinen and SL Ossakow. Theories of high-latitude ionospheric irregularities: A review. *Radio science*, 18(06):1077–1091, 1983b.
- MJ Keskinen, HG Mitchell, JA Fedder, P Satyanarayana, ST Zalesak, and JD Huba. Nonlinear evolution of the kelvin-helmholtz instability in the high-latitude ionosphere. *Journal of Geophysical Research: Space Physics*, 93(A1):137–152, 1988.

- MJ Keskinen, Su Basu, and Santimay Basu. Midlatitude sub-auroral ionospheric small scale structure during a magnetic storm. *Geophysical research letters*, 31(9), 2004.
- Kyung Sung Kim and Moo-Hyun Kim. Simulation of the kelvin–helmholtz instability using a multi-liquid moving particle semi-implicit method. *Ocean Engineering*, 130:531–541, 2017.
- Paul M Kintner and Brent M Ledvina. The ionosphere, radio navigation, and global navigation satellite systems. *Advances in Space Research*, 35(5):788–811, 2005.
- BSR Kunduri, JBH Baker, JM Ruohoniemi, EG Thomas, SG Shepherd, and KT Sterne. Statistical characterization of the large-scale structure of the subauroral polarization stream. *Journal of Geophysical Research: Space Physics*, 122(6):6035–6048, 2017.
- BSR Kunduri, JBH Baker, JM Ruohoniemi, N Nishitani, K Oksavik, PJ Erickson, AJ Coster, SG Shepherd, WA Bristow, and ES Miller. A new empirical model of the subauroral polarization stream. *Journal of Geophysical Research: Space Physics*, 123(9):7342–7357, 2018.
- Leslie J Lamarche and Roman A Makarevich. Radar observations of density gradients, electric fields, and plasma irregularities near polar cap patches in the context of the gradient-drift instability. *Journal of Geophysical Research: Space Physics*, 122(3):3721–3736, 2017.
- BM Ledvina, Jonathan J Makela, and PM Kintner. First observations of intense gps ll amplitude scintillations at midlatitude. *Geophysical Research Letters*, 29(14):4–1, 2002.
- James R Lemen, David J Akin, Paul F Boerner, Catherine Chou, Jerry F Drake, Dexter W Duncan, Christopher G Edwards, Frank M Friedlaender, Gary F Heyman, Neal E Hurlburt, et al. The atmospheric imaging assembly (aia) on the solar dynamics observatory (sdo). In *The solar dynamics observatory*, pages 17–40. Springer, 2011.
- Dong Lin, Wenbin Wang, Wayne A Scales, Kevin Pham, Jing Liu, Binzheng Zhang, Viacheslav Merkin, Xueling Shi, Bharat Kunduri, and Maimaitirebike Maimaiti. Saps in the 17 march 2013 storm event: Initial results from the coupled magnetosphere-ionosphere-thermosphere model. *Journal of Geophysical Research: Space Physics*, 124(7):6212–6225, 2019.
- Lewis M Linson and Joseph B Workman. Formation of striations in ionospheric plasma clouds. *Journal of Geophysical Research*, 75(16):3211–3219, 1970.
- KH Lloyd and Gerhard Haerendel. Numerical modeling of the drift and deformation of ionospheric plasma clouds and of their interaction with other layers of the ionosphere. *Journal of geophysical research*, 78(31):7389–7415, 1973.
- Elizabeth A MacDonald, Eric Donovan, Yukitoshi Nishimura, Nathan A Case, D Megan Gillies, Bea Gallardo-Lacourt, William E Archer, Emma L Spanswick, Notanee Bourassa, Martin Connors, et al. New science in plain sight: Citizen scientists lead to the discovery of optical structure in the upper atmosphere. *Science advances*, 4(3):eaq0030, 2018.

- Roman A Makarevich. Symmetry considerations in the two-fluid theory of the gradient drift instability in the lower ionosphere. *Journal of Geophysical Research: Space Physics*, 119(9):7902–7913, 2014.
- Roman A Makarevich. Toward an integrated view of ionospheric plasma instabilities: Altitudinal transitions and strong gradient case. *Journal of Geophysical Research: Space Physics*, 121(4):3634–3647, 2016a.
- Roman A Makarevich. Toward an integrated view of ionospheric plasma instabilities: 2. three inertial modes of a cubic dispersion relation. *Journal of Geophysical Research: Space Physics*, 121(7):6855–6869, 2016b.
- Roman A Makarevich. Toward an integrated view of ionospheric plasma instabilities: 3. explicit growth rate and oscillation frequency for arbitrary altitude. *Journal of Geophysical Research: Space Physics*, 124(7):6138–6155, 2019a.
- Roman A Makarevich. Toward an integrated view of ionospheric plasma instabilities: 4. behavior in the transitional valley region. *Journal of Geophysical Research: Space Physics*, 124(11):9709–9724, 2019b.
- Roman A Makarevich. Toward an integrated view of ionospheric plasma instabilities: 5. ion-thermal instability for arbitrary ion magnetization, density gradient, and wave propagation. *Journal of Geophysical Research: Space Physics*, 125(9):e2020JA028349, 2020.
- RT Marriott, DE St John, RM Thorne, and SV Venkateswaran. Xuv image of the sun from eclipse observations of the ionospheric e-region. *Solar Physics*, 21(2):483–494, 1971.
- BE McDonald, MJ Keskinen, SL Ossakow, and ST Zalesak. Computer simulation of gradient drift instability processes in operation avefria. *Journal of Geophysical Research: Space Physics*, 85(A5):2143–2154, 1980.
- EV Mishin and WJ Burke. Stormtime coupling of the ring current, plasmasphere, and topside ionosphere: Electromagnetic and plasma disturbances. *Journal of Geophysical Research: Space Physics*, 110(A7), 2005.
- EV Mishin, WJ Burke, CY Huang, and FJ Rich. Electromagnetic wave structures within subauroral polarization streams. *Journal of Geophysical Research: Space Physics*, 108(A8), 2003.
- Evgeny Mishin and Natan Blaunstein. Irregularities within subauroral polarization stream-related troughs and gps radio interference at midlatitudes. *Midlatitude Ionospheric Dynamics and Disturbances, Geophys. Monogr. Ser.*, 181:291–295, 2008.
- Evgeny Mishin, Yukitoshi Nishimura, and John Foster. Saps/said revisited: A causal relation to the substorm current wedge. *Journal of Geophysical Research: Space Physics*, 122(8):8516–8535, 2017.

- HG Mitchell Jr, JA Fedder, JD Huba, and ST Zalesak. Transverse motion of high-speed barium clouds in the ionosphere. *Journal of Geophysical Research: Space Physics*, 90(A11):11091–11095, 1985a.
- HG Mitchell Jr, JA Fedder, MJ Keskinen, and ST Zalesak. A simulation of high latitude f-layer instabilities in the presence of magnetosphere-ionosphere coupling. *Geophysical research letters*, 12(5):283–286, 1985b.
- J Moen, K Oksavik, T Abe, Mark Lester, Y Saito, TA Bekkeng, and KS Jacobsen. First in-situ measurements of hf radar echoing targets. *Geophysical Research Letters*, 39(7), 2012.
- Sebastijan Mrak, Joshua Semeter, Douglas Drob, and JD Huba. Direct euv/x-ray modulation of the ionosphere during the august 2017 total solar eclipse. *Geophysical Research Letters*, 45(9):3820–3828, 2018.
- William L Oberkampf and Christopher J Roy. *Verification and validation in scientific computing*. Cambridge University Press, 2010.
- K Oksavik, RA Greenwald, JM Ruohoniemi, MR Hairston, LJ Paxton, JBH Baker, JW Gjerloev, and RJ Barnes. First observations of the temporal/spatial variation of the sub-auroral polarization stream from the superdarn wallops hf radar. *Geophysical research letters*, 33(12), 2006.
- Steven A Orszag. Numerical simulation of incompressible flows within simple boundaries: accuracy. *Journal of Fluid Mechanics*, 49(1):75–112, 1971.
- SL Ossakow, PK Chaturvedi, and JB Workman. High-altitude limit of the gradient drift instability. *Journal of Geophysical Research: Space Physics*, 83(A6):2691–2693, 1978.
- GS Patterson Jr and Steven A Orszag. Spectral calculations of isotropic turbulence: Efficient removal of aliasing interactions. *The Physics of Fluids*, 14(11):2538–2541, 1971.
- FW Perkins and JH Doles III. Velocity shear and the $e \times b$ instability. *Journal of Geophysical Research*, 80(1):211–214, 1975.
- FW Perkins, NJ Zabusky, and JH Doles III. Deformation and striation of plasma clouds in the ionosphere: 1. *Journal of Geophysical Research*, 78(4):697–709, 1973.
- JM Picone, AE Hedin, D Pj Drob, and AC Aikin. Nrlmsise-00 empirical model of the atmosphere: Statistical comparisons and scientific issues. *Journal of Geophysical Research: Space Physics*, 107(A12):SIA–15, 2002.
- Joachim Raeder, William D Cramer, Joseph Jensen, Timothy Fuller-Rowell, Naomi Maruyama, Frank Toffoletto, and Hien Vo. Sub-auroral polarization streams: A complex interaction between the magnetosphere, ionosphere, and thermosphere. In *Journal of Physics: Conference Series*, volume 767, page 012021. IOP Publishing, 2016.

- Chirag Rathod, Lee Kordella, Wayne Scales, Greg Earle, and Bhuvana Srinivasan. Likelihood of gradient drift instability development during the august 21, 2017 solar eclipse. *Radiation Effects and Defects in Solids*, 175(1-2):136–140, 2020.
- PG Richards, JA Fennelly, and DG Torr. Euvac: A solar euv flux model for aeronomic calculations. *Journal of Geophysical Research: Space Physics*, 99(A5):8981–8992, 1994.
- AD Richmond. Ionospheric electrodynamics using magnetic apex coordinates. *Journal of geomagnetism and geoelectricity*, 47(2):191–212, 1995.
- William Rideout and Anthea Coster. Automated gps processing for global total electron content data. *GPS solutions*, 10(3):219–228, 2006.
- H Rishbeth. Solar eclipses and ionospheric theory. *Space Science Reviews*, 8(4):543–554, 1968.
- A Rogister and N d’Angelo. Type ii irregularities in the equatorial electrojet. *Journal of Geophysical Research*, 75(19):3879–3887, 1970.
- JM Ruohoniemi, RA Greenwald, J-P Villain, KB Baker, PT Newell, and C-I Meng. Coherent hf radar backscatter from small-scale irregularities in the dusk sector of the subauroral ionosphere. *Journal of Geophysical Research: Space Physics*, 93(A11):12871–12882, 1988.
- John D Sahr and Bela G Fejer. Auroral electrojet plasma irregularity theory and experiment: A critical review of present understanding and future directions. *Journal of Geophysical Research: Space Physics*, 101(A12):26893–26909, 1996.
- Tetsuya Sato, Takao Tsuda, and Ken-ichi Maeda. Fully developed turbulent irregularities in the ionosphere due to cross-field plasma instability. *Radio Science*, 3(6):529–534, 1968.
- WA Scales and PA Bernhardt. Simulation of high-speed (orbital) releases of electron attachment materials in the ionosphere. *Journal of Geophysical Research: Space Physics*, 96(A8):13815–13828, 1991.
- Ro Wo Schunk, P Mo Banks, and W Jo Raitt. Effects of electric fields and other processes upon the nighttime high-latitude f layer. *Journal of Geophysical Research*, 81(19):3271–3282, 1976.
- Joshua Semeter, Michael Hunnekuhl, Elizabeth MacDonald, Michael Hirsch, Neil Zeller, Alexei Chernenkoff, and Jun Wang. The mysterious green streaks below steve. *AGU Advances*, page e2020AV000183, 2020.
- Uri Shumlak and J Loverich. Approximate riemann solver for the two-fluid plasma model. *Journal of Computational Physics*, 187(2):620–638, 2003.
- Albert Simon. Instability of a partially ionized plasma in crossed electric and magnetic fields. *The physics of fluids*, 6(3):382–388, 1963.

- A Spicher, T Cameron, EM Grono, KN Yakymenko, Stephan C Buchert, LBN Clausen, DJ Knudsen, KA McWilliams, and JI Moen. Observation of polar cap patches and calculation of gradient drift instability growth times: A swarm case study. *Geophysical Research Letters*, 42(2):201–206, 2015.
- Andres Spicher, Kshitija Deshpande, Yaqi Jin, Kjellmar Oksavik, Matthew D Zettergren, Lasse BN Clausen, Jøran I Moen, Marc R Hairston, and Lisa Baddeley. On the production of ionospheric irregularities via kelvin-helmholtz instability associated with cusp flow channels. *Journal of Geophysical Research: Space Physics*, 125(6):e2019JA027734, 2020.
- RW Spiro, RA Heelis, and WB Hanson. Rapid subauroral ion drifts observed by atmosphere explorer c. *Geophysical Research Letters*, 6(8):657–660, 1979.
- AV Streltsov and JC Foster. Electrodynamics of the magnetosphere–ionosphere coupling in the nightside subauroral zone. *Physics of Plasmas*, 11(4):1260–1267, 2004.
- AV Streltsov and EV Mishin. Numerical modeling of localized electromagnetic waves in the nightside subauroral zone. *Journal of Geophysical Research: Space Physics*, 108(A8), 2003.
- RN Sudan, J Akinrimisi, and DT Farley. Generation of small-scale irregularities in the equatorial electrojet. *Journal of Geophysical Research*, 78(1):240–248, 1973.
- Hiromitsu Takeuchi, Naoya Suzuki, Kenichi Kasamatsu, Hiroki Saito, and Makoto Tsubota. Quantum kelvin-helmholtz instability in phase-separated two-component bose-einstein condensates. *Physical Review B*, 81(9):094517, 2010.
- Erwan Thébault, Christopher C Finlay, Ciarán D Beggan, Patrick Alken, Julien Aubert, Olivier Barrois, Francois Bertrand, Tatiana Bondar, Axel Boness, Laura Brocco, et al. International geomagnetic reference field: the 12th generation. *Earth, Planets and Space*, 67(1):79, 2015.
- EG Thomas, Joseph BH Baker, J Michael Ruohoniemi, Lasse BN Clausen, AJ Coster, JC Foster, and PJ Erickson. Direct observations of the role of convection electric field in the formation of a polar tongue of ionization from storm enhanced density. *Journal of Geophysical Research: Space Physics*, 118(3):1180–1189, 2013.
- Roland T Tsunoda. High-latitude f region irregularities: A review and synthesis. *Reviews of Geophysics*, 26(4):719–760, 1988.
- Roland T Tsunoda and James F Vickrey. Evidence of east-west structure in large-scale f-region plasma enhancements in the auroral zone. Technical report, SRI INTERNATIONAL MENLO PARK CA, 1982.
- Takayuki Umeda and Yasutaka Wada. Secondary instabilities in the collisionless rayleigh-taylor instability: Full kinetic simulation. *Physics of Plasmas*, 23(11):112117, 2016.

- Mario Vietri, Andrea Ferrara, and Francesco Miniati. The survival of interstellar clouds against kelvin-helmholtz instabilities. *The Astrophysical Journal*, 483(1):262, 1997.
- Hui Wang, Aaron J Ridley, Hermann Lühr, Michael W Liemohn, and Shu Y Ma. Statistical study of the subauroral polarization stream: Its dependence on the cross-polar cap potential and subauroral conductance. *Journal of Geophysical Research: Space Physics*, 113(A12), 2008.
- Hui Wang, Hermann Lühr, Kathrin Häusler, and Patricia Ritter. Effect of subauroral polarization streams on the thermosphere: A statistical study. *Journal of Geophysical Research: Space Physics*, 116(A3), 2011.
- Hui Wang, Hermann Lühr, and Shu Y Ma. The relation between subauroral polarization streams, westward ion fluxes, and zonal wind: Seasonal and hemispheric variations. *Journal of Geophysical Research: Space Physics*, 117(A4), 2012a.
- Hui Wang, Kedeng Zhang, Zhichao Zheng, and Aaron James Ridley. The effect of subauroral polarization streams on the mid-latitude thermospheric disturbance neutral winds: a universal time effect. *Annales Geophysicae (09927689)*, 36(2), 2018.
- LF Wang, WH Ye, and YJ Li. Combined effect of the density and velocity gradients in the combination of kelvin-helmholtz and rayleigh-taylor instabilities. *Physics of Plasmas*, 17(4):042103, 2010.
- Wenbin Wang, Elsayed R Talaat, Alan G Burns, Barbary Emery, Syau-yun Hsieh, Jiuhou Lei, and Jiyao Xu. Thermosphere and ionosphere response to subauroral polarization streams (saps): Model simulations. *Journal of Geophysical Research: Space Physics*, 117(A7), 2012b.
- Edward J Weber, Jürgen Buchau, JG Moore, JR Sharber, RC Livingston, John David Winningham, and BW Reinisch. F layer ionization patches in the polar cap. *Journal of Geophysical Research: Space Physics*, 89(A3):1683–1694, 1984.
- Heng Yang, Enrique Monte Moreno, and Manuel Hernández-Pajares. Detection and description of the different ionospheric disturbances that appeared during the solar eclipse of 21 august 2017. *Remote Sensing*, 10(11):1710, 2018.
- Yiqun Yu, Vania Jordanova, Shasha Zou, Roderick Heelis, Mike Ruohoniemi, and John Wygant. Modeling subauroral polarization streams during the 17 march 2013 storm. *Journal of Geophysical Research: Space Physics*, 120(3):1738–1750, 2015.
- NJ Zabusky, JH Doles III, and FW Perkins. Deformation and striation of plasma clouds in the ionosphere: 2. numerical simulation of a nonlinear two-dimensional model. *Journal of Geophysical Research*, 78(4):711–724, 1973.

- ST Zalesak, JF Drake, and JD Huba. Three-dimensional simulation study of ionospheric plasma clouds. *Geophysical research letters*, 17(10):1597–1600, 1990.
- Huan-Hao Zhang, Chun Zheng, Nadine Aubry, Wei-Tao Wu, and Zhi-Hua Chen. Numerical analysis of richtmyer–meshkov instability of circular density interface in presence of transverse magnetic field. *Physics of Fluids*, 32(11):116104, 2020.
- KeDeng Zhang, Hui Wang, WenBin Wang, Jing Liu, ShunRong Zhang, and Cheng Sheng. Nighttime meridional neutral wind responses to saps simulated by the tiegcm: a universal time effect. *Earth and Planetary Physics*, 5(1):1–11, 2021.
- Shun-Rong Zhang, Philip J Erickson, John C Foster, John M Holt, Anthea J Coster, Jonathan J Makela, John Noto, John W Meriwether, Brian J Harding, Juanita Riccobono, et al. Thermospheric poleward wind surge at midlatitudes during great storm intervals. *Geophysical Research Letters*, 42(13):5132–5140, 2015.
- Shun-Rong Zhang, Philip J Erickson, Larisa P Goncharenko, Anthea J Coster, William Rideout, and Juha Vierinen. Ionospheric bow waves and perturbations induced by the 21 august 2017 solar eclipse. *Geophysical Research Letters*, 44(24):12–067, 2017.
- Yihua Zheng, Pontus C Brandt, Anthony TY Lui, and Mei-Ching Fok. On ionospheric trough conductance and subauroral polarization streams: Simulation results. *Journal of Geophysical Research: Space Physics*, 113(A4), 2008.

Appendix A

Linear Theory

A.1 Geometry

The current closure and continuity equations are considered separately by looking at individual terms and then putting them back together in the end. Note that there is an additional coefficient, I , in front of the inertial terms. This is either 0 or 1 and acts as a switch to turn the inertial terms on or off. The density and electric potential are assumed to be a background (subscript 0 terms) plus a plane wave perturbation (subscript 1 terms) of the form

$$n = n_0 + \underbrace{\tilde{n} \exp [i(\mathbf{k} \cdot \mathbf{r} - \omega t)]}_{n_1} \quad (\text{A.1})$$

$$\phi = \phi_0 + \underbrace{\tilde{\phi} \exp [i(\mathbf{k} \cdot \mathbf{r} - \omega t)]}_{\phi_1}, \quad (\text{A.2})$$

where \mathbf{k} is the wavenumber, \mathbf{r} is the position vector in Cartesian coordinates, ω is the wave frequency, and the subscripts correspond to the order of the terms. Based on the definition, the derivatives of the first order terms become algebraic expressions. The problem is further simplified by assuming a slab geometry such that

$$\mathbf{k} = k_x \hat{\mathbf{x}} + k_y \hat{\mathbf{y}} \quad (\text{A.3})$$

$$\mathbf{u} = u_x \hat{\mathbf{x}} + u_y \hat{\mathbf{y}} \quad (\text{A.4})$$

$$\mathbf{B} = B \hat{\mathbf{z}} \quad (\text{A.5})$$

$$n = n_0(x) + n_1(x, y, t) \quad (\text{A.6})$$

$$\phi = \phi_0(x) + \phi_1(x, y, t) \quad (\text{A.7})$$

$$T_i = \text{const} \quad (\text{A.8})$$

$$T_e = \text{const.} \quad (\text{A.9})$$

A.2 Current Closure

The current closure, Eq. 3.82, rewritten here for convenience, with the source term, Eq. 3.83, and the factor I to switch the inertial terms on and off is

$$I\nabla \cdot \left(n \frac{\nabla \phi}{\partial t} \right) = \frac{1}{e} \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left(\frac{\nu_{en}}{\Omega_{ce}} \nabla^2 P_e - \frac{\nu_{in}}{\Omega_{ci}} \nabla^2 P_i \right) - I\nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right) \\ + \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right)^{-1} \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left[\mathbf{u} \times \mathbf{B} \cdot \nabla n - \nabla \cdot (n \nabla \phi) \right] \quad (\text{A.10})$$

All terms are multiplied by $e(1/\Omega_{ci} + 1/\Omega_{ce})$ and brought over to the left hand side, resulting in

$$Ie \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left[\underbrace{\nabla \cdot \left(n \frac{\nabla \phi}{\partial t} \right)}_1 + \underbrace{\nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right)}_2 \right] \underbrace{- \frac{\nu_{en}}{\Omega_{ce}} \nabla^2 P_e}_3 + \underbrace{\frac{\nu_{in}}{\Omega_{ci}} \nabla^2 P_i}_4 \\ + \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left[\underbrace{\nabla \cdot (n \nabla \phi)}_5 \underbrace{- \mathbf{u} \times \mathbf{B} \cdot \nabla n}_6 \right] = 0, \quad (\text{A.11})$$

which is split into terms 1 through 6. Each term is considered individually and then summed together at the end. Linearizing term 1 results in

$$1: \quad \nabla \cdot \left(n \frac{\partial \nabla \phi}{\partial t} \right) = \nabla \cdot \left[\left(n_0 + n_1 \right) \left(\cancel{\nabla \frac{\partial \phi_0}{\partial t}} + \nabla \frac{\partial \phi_1}{\partial t} \right) \right] \\ = -i\omega \nabla \cdot \left(n_0 \nabla \phi_1 \right) \\ = -i\omega \left(n_0 \nabla^2 \phi_1 + \nabla n_0 \cdot \nabla \phi_1 \right) \\ = -i\omega \left(-n_0 \left[k_x^2 + k_y^2 \right] \phi_1 + ik_x \frac{\partial n_0}{\partial x} \phi_1 \right) \\ = \left[\left(k_x \frac{\partial n_0}{\partial x} \right) + i \left(\omega \left[k_x^2 + k_y^2 \right] n_0 \right) \right] \phi_1. \quad (\text{A.12})$$

The $\mathbf{E} \times \mathbf{B}$ drift and $\nabla \nabla \phi$, in two dimensions, are

$$\mathbf{V}_{\mathbf{E} \times \mathbf{B}} = \frac{1}{B} \begin{bmatrix} -\frac{\partial \phi}{\partial y} \\ \frac{\partial \phi}{\partial x} \end{bmatrix} = \frac{1}{B} \begin{bmatrix} -ik_y \phi_1 \\ \frac{\partial \phi_0}{\partial x} + ik_x \phi_1 \end{bmatrix} \quad (\text{A.13})$$

$$\nabla \nabla \phi = \begin{bmatrix} \frac{\partial^2 \phi}{\partial x^2} & \frac{\partial^2 \phi}{\partial x \partial y} \\ \frac{\partial^2 \phi}{\partial x \partial y} & \frac{\partial^2 \phi}{\partial y^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial^2 \phi_0}{\partial x^2} - k_x^2 \phi_1 & -k_x k_y \phi_1 \\ -k_x k_y \phi_1 & -k_y^2 \phi_1 \end{bmatrix}. \quad (\text{A.14})$$

Therefore, term 2 is

$$\begin{aligned}
2: \quad \nabla \cdot \left(n \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla \nabla \phi \right) &= \frac{1}{B} \nabla \cdot \left(n \begin{bmatrix} -ik_y \phi_1 \\ \frac{\partial \phi_0}{\partial x} + ik_x \phi_1 \end{bmatrix} \begin{bmatrix} \frac{\partial^2 \phi_0}{\partial x^2} - k_x^2 \phi_1 & -k_x k_y \phi_1 \\ -k_x k_y \phi_1 & -k_y^2 \phi_1 \end{bmatrix} \right) \\
&= \frac{1}{B} \nabla \cdot \left(n \begin{bmatrix} -ik_y \phi_1 \left(\frac{\partial^2 \phi_0}{\partial x^2} - k_x^2 \phi_1 \right) - k_x k_y \phi_1 \left(\frac{\partial \phi_0}{\partial x} + ik_x \phi_1 \right) \\ -ik_y \phi_1 \left(-k_x k_y \phi_1 \right) - k_y^2 \phi_1 \left(\frac{\partial \phi_0}{\partial x} + ik_x \phi_1 \right) \end{bmatrix} \right) \\
&= -\frac{k_y}{B} \nabla \cdot \left(\left[n_0 + n_1 \right] \phi_1 \begin{bmatrix} i \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial \phi_0}{\partial x} \\ k_y \frac{\partial \phi_0}{\partial x} \end{bmatrix} \right) \\
&= -\frac{k_y}{B} \left(\nabla [n_0 \phi_1] \cdot \begin{bmatrix} i \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial \phi_0}{\partial x} \\ k_y \frac{\partial \phi_0}{\partial x} \end{bmatrix} + n_0 \phi_1 \nabla \cdot \begin{bmatrix} i \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial \phi_0}{\partial x} \\ k_y \frac{\partial \phi_0}{\partial x} \end{bmatrix} \right) \\
&= -\frac{k_y}{B} \left([n_0 \nabla \phi_1 + \phi_1 \nabla n_0] \cdot \begin{bmatrix} i \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial \phi_0}{\partial x} \\ k_y \frac{\partial \phi_0}{\partial x} \end{bmatrix} + n_0 \phi_1 \left[i \frac{\partial^3 \phi_0}{\partial x^3} + k_x \frac{\partial^2 \phi_0}{\partial x^2} \right] \right) \\
&= -\frac{k_y}{B} \left(\begin{bmatrix} ik_x n_0 \phi_1 + \frac{\partial n_0}{\partial x} \phi_1 \\ ik_y n_0 \phi_1 \end{bmatrix} \cdot \begin{bmatrix} i \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial \phi_0}{\partial x} \\ k_y \frac{\partial \phi_0}{\partial x} \end{bmatrix} + n_0 \phi_1 \left[i \frac{\partial^3 \phi_0}{\partial x^3} + k_x \frac{\partial^2 \phi_0}{\partial x^2} \right] \right) \\
&= -\frac{k_y}{B} \left[\left(ik_x n_0 + \frac{\partial n_0}{\partial x} \right) \left(i \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial \phi_0}{\partial x} \right) \phi_1 + ik_y^2 n_0 \frac{\partial \phi_0}{\partial x} \phi_1 + n_0 \phi_1 \left(i \frac{\partial^3 \phi_0}{\partial x^3} + k_x \frac{\partial^2 \phi_0}{\partial x^2} \right) \right] \\
&= -\frac{k_y}{B} \left[-k_x n_0 \frac{\partial^2 \phi_0}{\partial x^2} + ik_x^2 n_0 \frac{\partial \phi_0}{\partial x} + i \frac{\partial n_0}{\partial x} \frac{\partial^2 \phi_0}{\partial x^2} + k_x \frac{\partial n_0}{\partial x} \frac{\partial \phi_0}{\partial x} + ik_y^2 n_0 \frac{\partial \phi_0}{\partial x} + in_0 \frac{\partial^3 \phi_0}{\partial x^3} + k_x n_0 \frac{\partial^2 \phi_0}{\partial x^2} \right] \phi_1 \\
&= \left[\left(-\frac{k_x k_y}{B} \frac{\partial n_0}{\partial x} \frac{\partial \phi_0}{\partial x} + i \left(-\frac{k_y}{B} \right) \left(n_0 \frac{\partial^3 \phi_0}{\partial x^3} + \frac{\partial n_0}{\partial x} \frac{\partial^2 \phi_0}{\partial x^2} + [k_x^2 + k_y^2] n_0 \frac{\partial \phi_0}{\partial x} \right) \right] \phi_1. \quad (\text{A.15})
\end{aligned}$$

Terms 3 through 6 are fairly straightforward to obtain and are

$$\begin{aligned}
3: \quad -\frac{\nu_{en}}{\Omega_{ce}} \nabla^2 P_e &= -\frac{\nu_{en}}{\Omega_{ce}} k_B T_e \nabla^2 (\mu_0 + n_1) \\
&= \left[\frac{\nu_{en}}{\Omega_{ce}} (k_x^2 + k_y^2) k_B T_e \right] n_1 \quad (\text{A.16})
\end{aligned}$$

$$\begin{aligned}
4: \quad \frac{\nu_{in}}{\Omega_{ci}} \nabla^2 P_i &= \frac{\nu_{in}}{\Omega_{ci}} k_B T_i \nabla^2 (\mu_0 + n_1) \\
&= \left[-\frac{\nu_{in}}{\Omega_{ci}} (k_x^2 + k_y^2) k_B T_i \right] n_1 \quad (\text{A.17})
\end{aligned}$$

$$\begin{aligned}
5: \quad & \nabla \cdot (n \nabla \phi) = \nabla n \cdot \nabla \phi + n \nabla^2 \phi \\
& = \begin{bmatrix} \frac{\partial n_0}{\partial x} + ik_x n_1 \\ ik_y n_1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \phi_0}{\partial x} + ik_x \phi_1 \\ ik_y \phi_1 \end{bmatrix} + (n_0 + n_1) \left[\frac{\partial^2 \phi_0}{\partial x^2} - (k_x^2 + k_y^2) \phi_1 \right] \\
& = \cancel{\frac{\partial n_0}{\partial x} \frac{\partial \phi_0}{\partial x}} + ik_x \frac{\partial n_0}{\partial x} \phi_1 + ik_x n_1 \frac{\partial \phi_0}{\partial x} + \cancel{i^2 k_x^2 n_1 \phi_1} + \cancel{n_0 \frac{\partial^2 \phi_0}{\partial x^2}} - n_0 (k_x^2 + k_y^2) \phi_1 + n_1 \frac{\partial^2 \phi_0}{\partial x^2} \\
& \quad - \cancel{n_1 (k_x^2 + k_y^2) \phi_1} \\
& = \left[-n_0 (k_x^2 + k_y^2) + i \left(k_x \frac{\partial n_0}{\partial x} \right) \right] \phi_1 + \left[\frac{\partial^2 \phi_0}{\partial x^2} + i \left(k_x \frac{\partial \phi_0}{\partial x} \right) \right] n_1 \quad (\text{A.18})
\end{aligned}$$

$$\begin{aligned}
6: \quad & -\mathbf{u} \times \mathbf{B} \cdot \nabla n = \begin{bmatrix} -u_y B \\ u_x B \end{bmatrix} \cdot \begin{bmatrix} \cancel{\frac{\partial n_0}{\partial x}} + ik_x n_1 \\ ik_y n_1 \end{bmatrix} \\
& = \left[i \left(B [k_y u_x - k_x u_y] \right) \right] n_1. \quad (\text{A.19})
\end{aligned}$$

A.2.1 Linearized Equation

Substituting Eqs. A.12 through A.19 into Eq. A.11 provides the linearized current closure equation, which is

$$\begin{aligned}
& \left[Ie \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(k_x \frac{\partial n_0}{\partial x} - \frac{k_x k_y}{B} \frac{\partial n_0}{\partial x} \frac{\partial \phi_0}{\partial x} \right) - \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) n_0 (k_x^2 + k_y^2) \right. \\
& \left. + i \left(Ie \left[\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right] \left[\omega (k_x^2 + k_y^2) n_0 - \frac{k_y}{B} \left(n_0 \frac{\partial^3 \phi_0}{\partial x^3} + \frac{\partial n_0}{\partial x} \frac{\partial^2 \phi_0}{\partial x^2} + [k_x^2 + k_y^2] n_0 \frac{\partial \phi_0}{\partial x} \right) \right] \right) \right] \phi_1 \\
& + \left[(k_x^2 + k_y^2) k_B \left(\frac{\nu_{en}}{\Omega_{ce}} T_e - \frac{\nu_{in}}{\Omega_{ci}} T_i \right) + \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \frac{\partial^2 \phi_0}{\partial x^2} \right. \\
& \left. + i \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(k_x \frac{\partial \phi_0}{\partial x} + B [k_y u_x - k_x u_y] \right) \right] n_1 = 0. \quad (\text{A.20})
\end{aligned}$$

A.3 Continuity

The continuity equation, Eq. 3.84, with all the terms moved to the left hand side is

$$\underbrace{\frac{\partial n}{\partial t}}_7 + \underbrace{\mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla n}_8 - \underbrace{D \nabla^2 n}_9 = 0, \quad (\text{A.21})$$

which is split into terms 7 through 9. Using Eq. A.13 for the linearized $\mathbf{E} \times \mathbf{B}$ drift, the linearized forms of terms 7 through 9 are

$$7: \quad \frac{\partial n}{\partial t} = \cancel{\frac{\partial n_0}{\partial t}} + \frac{\partial n_1}{\partial t} = \left[i \begin{pmatrix} -\omega \end{pmatrix} \right] n_1 \quad (\text{A.22})$$

$$8: \quad \mathbf{V}_{\mathbf{E} \times \mathbf{B}} \cdot \nabla n = \frac{1}{B} \begin{bmatrix} -ik_y \phi_1 \\ \frac{\partial \phi_0}{\partial x} + ik_x \phi_1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial n_0}{\partial x} + ik_x n_1 \\ ik_y n_1 \end{bmatrix} = \left[i \begin{pmatrix} -\frac{k_y}{B} \frac{\partial n_0}{\partial x} \end{pmatrix} \right] \phi_1 + \left[i \begin{pmatrix} \frac{k_y}{B} \frac{\partial \phi_0}{\partial x} \end{pmatrix} \right] n_1 \quad (\text{A.23})$$

$$9: \quad -D \nabla^2 n = -D \left(\cancel{\nabla^2 n_0} + \nabla^2 n_1 \right) = \left[D \left(k_x^2 + k_y^2 \right) \right] n_1. \quad (\text{A.24})$$

A.3.1 Linearized Equation

Eqs. A.22 through A.24 are substituted into Eq. A.21, resulting in the linearized continuity equation, which is

$$\left[i \begin{pmatrix} -\frac{k_y}{B} \frac{\partial n_0}{\partial x} \end{pmatrix} \right] \phi_1 + \left[D \left(k_x^2 + k_y^2 \right) + i \begin{pmatrix} -\omega + \frac{k_y}{B} \frac{\partial \phi_0}{\partial x} \end{pmatrix} \right] n_1 = 0. \quad (\text{A.25})$$

A.4 Matrix Form

The linearized equations, Eqs. A.20 and A.25, can be rewritten as a matrix equation of the form

$$\mathbf{A} \mathbf{x} = \mathbf{0}, \quad (\text{A.26})$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad (\text{A.27})$$

and

$$\mathbf{x} = \begin{bmatrix} \phi_1 \\ n_1 \end{bmatrix}. \quad (\text{A.28})$$

Based on Eqs. A.20 and A.25, the elements in \mathbf{A} are

$$a_{11} = Ie \left(\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right) \left(k_x \frac{\partial n_0}{\partial x} - \frac{k_x k_y}{B} \frac{\partial n_0}{\partial x} \frac{\partial \phi_0}{\partial x} \right) - \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) n_0 (k_x^2 + k_y^2) + i \left(Ie \left[\frac{1}{\Omega_{ci}} + \frac{1}{\Omega_{ce}} \right] \left[\omega (k_x^2 + k_y^2) n_0 - \frac{k_y}{B} \left(n_0 \frac{\partial^3 \phi_0}{\partial x^3} + \frac{\partial n_0}{\partial x} \frac{\partial^2 \phi_0}{\partial x^2} + [k_x^2 + k_y^2] n_0 \frac{\partial \phi_0}{\partial x} \right) \right] \right) \quad (\text{A.29})$$

$$a_{12} = (k_x^2 + k_y^2) k_B \left(\frac{\nu_{en}}{\Omega_{ce}} T_e - \frac{\nu_{in}}{\Omega_{ci}} T_i \right) + \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \frac{\partial^2 \phi_0}{\partial x^2} + i \left(\frac{\nu_{in}}{\Omega_{ci}} + \frac{\nu_{en}}{\Omega_{ce}} \right) \left(k_x \frac{\partial \phi_0}{\partial x} + B [k_y u_x - k_x u_y] \right) \quad (\text{A.30})$$

$$a_{21} = i \left(- \frac{k_y}{B} \frac{\partial n_0}{\partial x} \right) \quad (\text{A.31})$$

$$a_{22} = D(k_x^2 + k_y^2) + i \left(- \omega + \frac{k_y}{B} \frac{\partial \phi_0}{\partial x} \right). \quad (\text{A.32})$$

The determinant of \mathbf{A} , based on Eq. A.26, is 0, resulting in

$$\det \mathbf{A} = a_{11} a_{22} - a_{12} a_{21} = 0, \quad (\text{A.33})$$

which is used to obtain the non-trivial solution. The dispersion relation is found by solving Eq. A.33 for ω . Expanding Eq. A.33 is quite complicated, but some simplifications can be made. When $I = 1$, Eq. A.33 is quadratic in ω and takes the form

$$a\omega^2 + b\omega + c = 0, \quad (\text{A.34})$$

which is solved using the quadratic formula, resulting in

$$\omega = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \quad (\text{A.35})$$

When $I = 0$, Eq. A.33 is linear in ω and takes the form

$$a\omega + b = 0, \quad (\text{A.36})$$

which, solving for ω , yields

$$\omega = -\frac{b}{a}. \quad (\text{A.37})$$

The terms that go into the coefficients a , b , and c for either case are extensive and difficult to solve for by hand. Therefore, the analytical solution for ω is found using Listing A.1. This is then evaluated numerically using Listing A.2. Presently, this is only benchmarked for the non-inertial case (see Section 3.3.2).

A.5 Code

Listing A.1: Mathematica notebook that analytically solves for ω using Eqs. A.35 and A.37.

```

1 (* This notebook solves the omega based on the dispersion relation using only the
   continuity and current closure equations with the following assumptions
2 k is in the x and y directions
3 n = n0(x) + n1(x,y,t)
4 phi = phi0(x) + phi1(x,y,t)
5 B is in the z direction
6 u is in the x and y directions
7 Is is a factor that turns the inertial terms on or off *)
8 In[2]:= (* Set elements of A matrix for inertial case. *)
9 (* In this case, Is = 1 *)
10 Is = 1;
11 a11 = Is * e (1/0ci+1/0ce)*
12 (kx*dn0dx - kx*ky*dn0dx*dphi0dx/B)-
13 (nuin/0ci+nuen/0ce)*n0*(kx^2+ky^2)+
14 I*( Is*e*(1/0ci+1/0ce)* (
15 \[Omega]*(kx^2+ky^2)*n0-
16 ky*(n0*d3phi0dx3 + dn0dx*d2phi0dx2+(kx^2+ky^2)*n0*dphi0dx)/B));
17 a12 = (kx^2+ky^2)*kb*(nuen*Te/0ce - nuin*Ti/0ci)+
18 (nuin/0ci+nuen/0ce)*d2phi0dx2+
19 I*(nuin/0ci+nuen/0ce)*
20 (kx*dphi0dx+B*(ky*ux-kx*uy));
21 a21 = -I*ky*dn0dx/B;
22 a22= Diff*(kx^2+ky^2)+I*(-\[Omega]+ky*dphi0dx/B);
23 In[7]:= (* Take determinant of A *)
24 detAIinertial = a11*a22 - a12*a21;
25 In[8]:= (* Get the coefficients of omega
26 The inertial case gives a quadratic polynomial in omega *)
27 coeffsInertial = Simplify[CoefficientList[detAIinertial,\[Omega]]];
28 In[9]:= (* Rewrite the coefficients in a simpler manner *)
29 cInertial = coeffsInertial[[1]];
30 bInertial = coeffsInertial[[2]];
31 aInertial = coeffsInertial[[3]];
32 In[12]:= (* Use the quadratic formula to get the solutions for omega *)
33 wpInertial =Simplify[( -bInertial + (bInertial^2 - 4 * aInertial * cInertial)^.5 ) /
   (2*aInertial)];
34 wmInertial = Simplify[( -bInertial - (bInertial^2 - 4 * aInertial * cInertial)^.5 ) /
   (2*aInertial)];
35 In[14]:= (* Now consider the noninertial case.*)
36 (* In this case, Is = 0 *)
37 (* Note that this is just copy pasted from the above but just changing Is *)
38 Is = 0;
39 a11 = Is * e (1/0ci+1/0ce)*
40 (kx*dn0dx - kx*ky*dn0dx*dphi0dx/B)-
41 (nuin/0ci+nuen/0ce)*n0*(kx^2+ky^2)+
42 I*( Is*e*(1/0ci+1/0ce)* (
43 \[Omega]*(kx^2+ky^2)*n0-

```

```

44 ky*(n0*d3phi0dx3 + dn0dx*d2phi0dx2+(kx^2+ky^2)*n0*dphi0dx)/B));
45 a12 = (kx^2+ky^2)*kb*(nuen*Te/Oce - nuin*Ti/Oci)+
46 (nuin/Oci+nuen/Oce)*d2phi0dx2+
47 I*(nuin/Oci+nuen/Oce)*
48 (kx*dphi0dx+B*(ky*ux-kx*uy));
49 a21 = -I*ky*dn0dx/B;
50 a22= Diff*(kx^2+ky^2)+I*(-\[Omega]+ky*dphi0dx/B);
51 In[19]:= (* Take determinant of A *)
52 detANoninertial = a11*a22 - a12*a21;
53 In[20]:= (* Get the coefficients of omega
54 The non-inertial case is linear in omega *)
55 coeffsNoninertial = Simplify[CoefficientList[detANoninertial,\[Omega]]];
56 In[21]:= bNoninertial = coeffsNoninertial[[1]];
57 aNoninertial = coeffsNoninertial[[2]];
58 In[23]:= (* Get solution for omega. *)
59 wNoninertial =Simplify[ -bNoninertial / aNoninertial];
60 (* Use the ToMatlab package ( found here
61     https://library.wolfram.com/infocenter/MathSource/577/ ) to convert this to Matlab
62     code. This is what is used in
63     cont_divJ_dispersionSolver.m *)
64 <<ToMatlab.m
65 In[34]:= PrintMatlab[wpInertial, "omega(1)"]
66 In[35]:= PrintMatlab[wmInertial, "omega(2)"]
67 In[32]:= PrintMatlab[wNoninertial,"omega"]

```

Listing A.2: Matlab function that calculates the wave frequency and growth rate based on the theory derived in Appendix A. The functional forms are found using Listing A.1.

```

1 function [omegaR, gamma] = cont_divJ_dispersionSolver(I, e, kx, ky, n0, dn0dx, ...
2     Oci, Oce, dphi0dx, d2phi0dx2, d3phi0dx3, B, ux, uy, nuin, nuen, kb, Diff, ...
3     Ti, Te)
4
5 % Note that this function assumes that all of the inputs are scalars
6 % If you want to see what happens when you change certain parameters, you will
7 % need to call this function in some sort of loop.
8
9 % This function solves for the real frequency and the growth rate
10 % from the dispersion relation using the following assumptions:
11 % Only the continuity and current closure equations are important.
12 % k is in the x and y directions
13 % n = n0(x) + n1(x,y,t)
14 % phi = phi0(x) + phi1(x,y,t)
15 % B is only in z
16 % The neutral wind is in the x and y directions
17
18 % For more information on how this is all derived,
19 % Please see Appendix A from Chirag Rathod's dissertation.
20
21 % omega is already solved for in the Mathematica notebook cont_divJ_dispersionSolver.nb
22

```



```

23 % If I is 1, then this is the inertial case.
24 % The dispersion relation is quadratic in omega and therefore there are 2 solutions
25 if I == 1
26     omega(1)=(1/2).*e.^(-1).*(kx.^2+ky.^2).^(-1).*n0.^(-1).*Oce.*Oci.*( ...
27         Oce+Oci).^(-1).*((sqrt(-1)*(-1)).*B.^(-1).*Oce.^(-1).*Oci.^( ...
28         -1).*((-1).*B.*dn0dx.*e.*kx.*(Oce+Oci)+sqrt(-1).*e.*ky.*( ...
29         d2phi0dx2.*dn0dx+(sqrt(-1)*(-1)).*dn0dx.*dphi0dx.*kx+( ...
30         d3phi0dx3+2.*dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+B.*( ...
31         kx.^2+ky.^2).*n0.*(nuin.*Oce+nuen.*Oci+Diff.*e.*(kx.^2+ ...
32         ky.^2).*(Oce+Oci)))+( (-1).*B.^(-2).*Oce.^(-2).*Oci.^(-2).*(( ...
33         (-1).*B.*dn0dx.*e.*kx.*(Oce+Oci)+sqrt(-1).*e.*ky.*( ...
34         d2phi0dx2.*dn0dx+(sqrt(-1)*(-1)).*dn0dx.*dphi0dx.*kx+( ...
35         d3phi0dx3+2.*dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+B.*( ...
36         kx.^2+ky.^2).*n0.*(nuin.*Oce+nuen.*Oci+Diff.*e.*(kx.^2+ ...
37         ky.^2).*(Oce+Oci)).^2+4.*e.*(kx.^2+ky.^2).*n0.*(Oce+Oci).*( ...
38         sqrt(-1).*dn0dx.*dphi0dx.*e.*kx.*ky.*(B+(-1).*dphi0dx.*ky).* ...
39         (Oce+Oci)+B.*Diff.*dn0dx.*e.*kx.*(B+(-1).*dphi0dx.*ky).*( ...
40         kx.^2+ky.^2).*(Oce+Oci)+dphi0dx.*e.*ky.^2.*(d2phi0dx2.* ...
41         dn0dx+(d3phi0dx3+dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+( ...
42         sqrt(-1)*(-1)).*B.*Diff.*e.*ky.*(kx.^2+ky.^2).*(d2phi0dx2.* ...
43         dn0dx+(d3phi0dx3+dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+( ...
44         sqrt(-1)*(-1)).*B.*dphi0dx.*ky.*(kx.^2+ky.^2).*n0.*(nuin.* ...
45         Oce+nuen.*Oci)+( (-1).*B.^2.*Diff.*(kx.^2+ky.^2).^2.*n0.*( ...
46         nuin.*Oce+nuen.*Oci)+sqrt(-1).*B.*dn0dx.*ky.*(d2phi0dx2.*( ...
47         nuin.*Oce+nuen.*Oci)+kb.*(kx.^2+ky.^2).*(nuen.*Oci.*Te+(-1) ...
48         .*nuin.*Oce.*Ti)+sqrt(-1).*(nuin.*Oce+nuen.*Oci).*(dphi0dx.* ...
49         kx+B.*ky.*ux+(-1).*B.*kx.*uy))))).^0.5E0);
50
51     omega(2)=(1/2).*e.^(-1).*(kx.^2+ky.^2).^(-1).*n0.^(-1).*Oce.*Oci.*( ...
52         Oce+Oci).^(-1).*((sqrt(-1)*(-1)).*B.^(-1).*Oce.^(-1).*Oci.^( ...
53         -1).*((-1).*B.*dn0dx.*e.*kx.*(Oce+Oci)+sqrt(-1).*e.*ky.*( ...
54         d2phi0dx2.*dn0dx+(sqrt(-1)*(-1)).*dn0dx.*dphi0dx.*kx+( ...
55         d3phi0dx3+2.*dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+B.*( ...
56         kx.^2+ky.^2).*n0.*(nuin.*Oce+nuen.*Oci+Diff.*e.*(kx.^2+ ...
57         ky.^2).*(Oce+Oci)))+( (-1).*((-1).*B.^(-2).*Oce.^(-2).*Oci.^( ...
58         -2).*((-1).*B.*dn0dx.*e.*kx.*(Oce+Oci)+sqrt(-1).*e.*ky.*( ...
59         d2phi0dx2.*dn0dx+(sqrt(-1)*(-1)).*dn0dx.*dphi0dx.*kx+( ...
60         d3phi0dx3+2.*dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+B.*( ...
61         kx.^2+ky.^2).*n0.*(nuin.*Oce+nuen.*Oci+Diff.*e.*(kx.^2+ ...
62         ky.^2).*(Oce+Oci)).^2+4.*e.*(kx.^2+ky.^2).*n0.*(Oce+Oci).*( ...
63         sqrt(-1).*dn0dx.*dphi0dx.*e.*kx.*ky.*(B+(-1).*dphi0dx.*ky).* ...
64         (Oce+Oci)+B.*Diff.*dn0dx.*e.*kx.*(B+(-1).*dphi0dx.*ky).*( ...
65         kx.^2+ky.^2).*(Oce+Oci)+dphi0dx.*e.*ky.^2.*(d2phi0dx2.* ...
66         dn0dx+(d3phi0dx3+dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+( ...
67         sqrt(-1)*(-1)).*B.*Diff.*e.*ky.*(kx.^2+ky.^2).*(d2phi0dx2.* ...
68         dn0dx+(d3phi0dx3+dphi0dx.*(kx.^2+ky.^2)).*n0).*(Oce+Oci)+( ...
69         sqrt(-1)*(-1)).*B.*dphi0dx.*ky.*(kx.^2+ky.^2).*n0.*(nuin.* ...
70         Oce+nuen.*Oci)+( (-1).*B.^2.*Diff.*(kx.^2+ky.^2).^2.*n0.*( ...
71         nuin.*Oce+nuen.*Oci)+sqrt(-1).*B.*dn0dx.*ky.*(d2phi0dx2.*( ...
72         nuin.*Oce+nuen.*Oci)+kb.*(kx.^2+ky.^2).*(nuen.*Oci.*Te+(-1) ...
73         .*nuin.*Oce.*Ti)+sqrt(-1).*(nuin.*Oce+nuen.*Oci).*(dphi0dx.* ...

```

```

74         kx+B.*ky.*ux+(-1).*B.*kx.*uy))))).^0.5E0);
75
76 elseif I == 0 % This is noninertial case. Linear in omega so only one solution
77     omega=(sqrt(-1)*(-1)).*B.^(-1).*(kx.^2+ky.^2).^(-1).*n0.^(-1).*( ...
78         nuin.*Oce+nuen.*Oci).^(-1).*((sqrt(-1)*(-1)).*d2phi0dx2.* ...
79         dn0dx.*ky.*(nuin.*Oce+nuen.*Oci)+(kx.^2+ky.^2).*(sqrt(-1).* ...
80         dphi0dx.*ky+B.*Diff.*(kx.^2+ky.^2)).*n0.*(nuin.*Oce+nuen.* ...
81         Oci)+dn0dx.*ky.*(dphi0dx.*kx.*(nuin.*Oce+nuen.*Oci)+(sqrt( ...
82         -1)*(-1)).*kb.*(kx.^2+ky.^2).*(nuen.*Oci.*Te+(-1).*nuin.* ...
83         Oce.*Ti)+B.*(nuin.*Oce+nuen.*Oci).*(ky.*ux+(-1).*kx.*uy));
84 end
85
86 % Split omega into its real and imaginary components
87 omegaR = real(omega);
88 gamma = imag(omega);

```

Appendix B

Code Verification Studies

B.1 Differentiation

This section lays out in greater detail the functions used to understand the discretization error in differentiation using the pseudo-spectral method with a Fourier basis, as shown in Section 3.5.2. The functions F_1 through F_7 , Eqs. 3.171-3.177, rewritten here for convenience, are

$$F_1 = a_x \sin\left(\frac{2\pi n_{x1}x}{L_x}\right) + a_y \cos\left(\frac{2\pi n_{y1}y}{L_y}\right) + a_{xy} \cos\left(\frac{2\pi n_{xy1}x}{L_x}\right) \sin\left(\frac{2\pi n_{xy2}y}{L_y}\right) \quad (\text{B.1})$$

$$F_2 = b_x \left[\tanh\left(\frac{x - x_{a1}}{c_x}\right) - \tanh\left(\frac{x - x_{a2}}{c_x}\right) \right] + b_y \left[\tanh\left(\frac{y - y_{a1}}{c_y}\right) - \tanh\left(\frac{y - y_{a2}}{c_y}\right) \right] \quad (\text{B.2})$$

$$F_3 = \sum_{i=1}^4 d_i \left[e_x \ln\left(\cosh\left[\frac{x - x_{b_i}}{g_x}\right]\right) + e_y \ln\left(\cosh\left[\frac{y - y_{b_i}}{g_y}\right]\right) \right] \quad (\text{B.3})$$

$$F_4 = l_x \exp\left[-\frac{(x - x_c)^2}{h_x}\right] + l_y \exp\left[-\frac{(y - y_c)^2}{h_y}\right] \quad (\text{B.4})$$

$$F_5 = m_x \exp\left[-\frac{(x - x_d)^2}{o_x} - \frac{(y - y_d)^2}{o_y}\right] \quad (\text{B.5})$$

$$F_6 = p_x \exp\left[-\frac{(x - x_e)^2}{q_x}\right] \sin\left(\frac{2\pi n_{y2}y}{L_y}\right) + p_y \exp\left[-\frac{(y - y_e)^2}{q_y}\right] \cos\left(\frac{2\pi n_{x2}x}{L_x}\right) \quad (\text{B.6})$$

$$F_7 = \sum_{i=1}^6 F_i. \quad (\text{B.7})$$

The derivatives of Eqs. B.1-B.7 with respect to x are

$$\frac{\partial F_1}{\partial x} = \frac{2\pi a_x n_{x1}}{L_x} \cos\left(\frac{2\pi n_{x1} x}{L_x}\right) - \frac{2\pi a_{xy} n_{xy1}}{L_x} \sin\left(\frac{2\pi n_{xy1} x}{L_x}\right) \sin\left(\frac{2\pi n_{xy2} y}{L_y}\right) \quad (\text{B.8})$$

$$\frac{\partial F_2}{\partial x} = \frac{b_x}{c_x} \left[\operatorname{sech}^2\left(\frac{x - x_{a1}}{c_x}\right) - \operatorname{sech}^2\left(\frac{x - x_{a2}}{c_x}\right) \right] \quad (\text{B.9})$$

$$\frac{\partial F_3}{\partial x} = \frac{e_x}{g_x} \sum_{i=1}^4 d_i \tanh\left(\frac{x - x_{b_i}}{g_x}\right) \quad (\text{B.10})$$

$$\frac{\partial F_4}{\partial x} = \frac{2l_x(x_c - x)}{h_x} \exp\left[-\frac{(x - x_c)^2}{h_x}\right] \quad (\text{B.11})$$

$$\frac{\partial F_5}{\partial x} = \frac{2m_x(x_d - x)}{o_x} \exp\left[-\frac{(x - x_d)^2}{o_x} - \frac{(y - y_d)^2}{o_y}\right] \quad (\text{B.12})$$

$$\begin{aligned} \frac{\partial F_6}{\partial x} = & \frac{2p_x(x_e - x)}{q_x} \exp\left[-\frac{(x - x_e)^2}{q_x}\right] \sin\left(\frac{2\pi n_{y2} y}{L_y}\right) \\ & - \frac{2\pi p_y n_{x2}}{L_x} \exp\left[-\frac{(y - y_e)^2}{q_y}\right] \sin\left(\frac{2\pi n_{x2} x}{L_x}\right) \end{aligned} \quad (\text{B.13})$$

$$\frac{\partial F_7}{\partial x} = \sum_{i=1}^6 \frac{\partial F_i}{\partial x}. \quad (\text{B.14})$$

The derivatives of Eqs. B.1-B.7 with respect to y are

$$\frac{\partial F_1}{\partial y} = -\frac{2\pi a_y n_{y1}}{L_y} \sin\left(\frac{2\pi n_{y1} y}{L_y}\right) + \frac{2\pi a_{xy} n_{xy2}}{L_y} \cos\left(\frac{2\pi n_{xy1} x}{L_x}\right) \cos\left(\frac{2\pi n_{xy2} y}{L_y}\right) \quad (\text{B.15})$$

$$\frac{\partial F_2}{\partial y} = \frac{b_y}{c_y} \left[\operatorname{sech}^2\left(\frac{y - y_{a1}}{c_y}\right) - \operatorname{sech}^2\left(\frac{y - y_{a2}}{c_y}\right) \right] \quad (\text{B.16})$$

$$\frac{\partial F_3}{\partial y} = \frac{e_y}{g_y} \sum_{i=1}^4 d_i \tanh\left(\frac{y - y_{b_i}}{g_y}\right) \quad (\text{B.17})$$

$$\frac{\partial F_4}{\partial y} = \frac{2l_y(y_c - y)}{h_y} \exp\left[-\frac{(y - y_c)^2}{h_y}\right] \quad (\text{B.18})$$

$$\frac{\partial F_5}{\partial y} = \frac{2m_x(y_d - y)}{o_y} \exp\left[-\frac{(x - x_d)^2}{o_x} - \frac{(y - y_d)^2}{o_y}\right] \quad (\text{B.19})$$

$$\begin{aligned} \frac{\partial F_6}{\partial y} = & \frac{2\pi p_x n_{y2}}{L_y} \exp\left[-\frac{(x - x_e)^2}{q_x}\right] \cos\left(\frac{2\pi n_{y2} y}{L_y}\right) \\ & + \frac{2p_y(y_e - y)}{q_y} \exp\left[-\frac{(y - y_e)^2}{q_y}\right] \cos\left(\frac{2\pi n_{x2} x}{L_x}\right) \end{aligned} \quad (\text{B.20})$$

$$\frac{\partial F_7}{\partial y} = \sum_{i=1}^6 \frac{\partial F_i}{\partial y}. \quad (\text{B.21})$$

The Laplacians of Eqs. B.1-B.7 are

$$\begin{aligned} \nabla^2 F_1 = & -\frac{4\pi^2}{L_x^2 L_y^2} \left[a_x L_y^2 n_{x_1}^2 \sin\left(\frac{2\pi n_{x_1} x}{L_x}\right) + a_y L_x^2 n_{y_1}^2 \cos\left(\frac{2\pi n_{y_1} y}{L_y}\right) \right. \\ & \left. a_{xy} \left(L_x^2 n_{xy_2}^2 + L_y^2 n_{xy_1}^2 \right) \cos\left(\frac{2\pi n_{xy_1} x}{L_x}\right) \sin\left(\frac{2\pi n_{xy_2} y}{L_y}\right) \right] \end{aligned} \quad (\text{B.22})$$

$$\begin{aligned} \nabla^2 F_2 = & \frac{2b_x}{c_x^2} \left[-\operatorname{sech}^2\left(\frac{x-x_{a_1}}{c_x}\right) \tanh\left(\frac{x-x_{a_1}}{c_x}\right) + \operatorname{sech}^2\left(\frac{x-x_{a_2}}{c_x}\right) \tanh\left(\frac{x-x_{a_2}}{c_x}\right) \right] \\ & + \frac{2b_y}{c_y^2} \left[-\operatorname{sech}^2\left(\frac{y-y_{a_1}}{c_y}\right) \tanh\left(\frac{y-y_{a_1}}{c_y}\right) + \operatorname{sech}^2\left(\frac{y-y_{a_2}}{c_y}\right) \tanh\left(\frac{y-y_{a_2}}{c_y}\right) \right] \end{aligned} \quad (\text{B.23})$$

$$\nabla^2 F_3 = \sum_{i=1}^4 d_i \left[\frac{e_x}{g_x^2} \operatorname{sech}^2\left(\frac{x-x_{b_i}}{g_x}\right) + \frac{e_y}{g_y^2} \operatorname{sech}^2\left(\frac{y-y_{b_i}}{g_y}\right) \right] \quad (\text{B.24})$$

$$\begin{aligned} \nabla^2 F_4 = & \frac{2l_x}{h_x} \left[\frac{2(x-x_c)^2}{h_x} \exp\left(-\frac{[x-x_c]^2}{h_x}\right) - \exp\left(-\frac{[x-x_c]^2}{h_x}\right) \right] \\ & \frac{2l_y}{h_y} \left[\frac{2(y-y_c)^2}{h_y} \exp\left(-\frac{[y-y_c]^2}{h_y}\right) - \exp\left(-\frac{[y-y_c]^2}{h_y}\right) \right] \end{aligned} \quad (\text{B.25})$$

$$\nabla^2 F_5 = -\frac{2m_x [o_x o_y^2 - 2o_y^2(x-x_d)^2 + o_x^2(o_y - 2[y-y_d]^2)]}{o_x^2 o_y^2} \exp\left[-\frac{(x-x_d)^2}{o_x} - \frac{(y-y_d)^2}{o_y}\right] \quad (\text{B.26})$$

$$\begin{aligned} \nabla^2 F_6 = & -2p_x \left[\frac{2\pi^2 n_{y_2}^2}{L_y^2} + \frac{q_x - 2(x-x_e)^2}{q_x^2} \right] \exp\left[-\frac{(x-x_e)^2}{q_x}\right] \sin\left(\frac{2\pi n_{y_2} y}{L_y}\right) \\ & + -2p_y \left[\frac{2\pi^2 n_{x_2}^2}{L_x^2} + \frac{q_y - 2(y-y_e)^2}{q_y^2} \right] \exp\left[-\frac{(y-y_e)^2}{q_y}\right] \sin\left(\frac{2\pi n_{x_2} x}{L_x}\right) \end{aligned} \quad (\text{B.27})$$

$$\nabla^2 F_7 = \sum_{i=1}^6 \nabla^2 F_i. \quad (\text{B.28})$$

The coefficients for all of these functions are found in Table B.1. L_x and L_y define the domain size. The table is organized such that a horizontal line indicates the coefficients of the next function.

Figures B.1 through B.4 show the exact values for the original functions (Eqs. B.1-B.7), their x derivatives (Eqs. B.8-B.14), their y derivatives (Eqs. B.15-B.21), and their Laplacians (Eqs. B.22-B.28), respectively.

Table B.1: A table of the coefficients used in Eqs. B.1-B.7. The first two define the domain size. The separation by horizontal lines indicates that the coefficients correspond to a different function.

Coefficient	Value
L_x	1×10^4
L_y	1×10^4
a_x	1
a_y	2
a_{xy}	0.7
n_{x_1}	1
n_{y_1}	100
n_{xy_1}	3
n_{xy_2}	10
b_x	1
b_y	2
c_x	300
c_y	400
$x_{a_{1,2}}$	$[0.3, 0.7]L_x$
$y_{a_{1,2}}$	$[0.35, 0.65]L_y$
$d_{1,2,3,4}$	$[1, -1, -1, 1]$
g_x	200
g_y	250
e_x	-0.5
e_y	0.25
$x_{b_{1,2,3,4}}$	$[0.2, 0.3, 0.7, 0.8]L_x$
$y_{b_{1,2,3,4}}$	$[0.2, 0.3, 0.7, 0.8]L_y$
l_x	3
l_y	-4
h_x	4×10^4
h_y	8×10^4
x_c	$0.4L_x$
y_c	$0.7L_y$
m_x	-6
x_d	$0.55L_x$
y_d	$0.5L_y$
o_x	5×10^5
o_y	4×10^5
p_x	2
p_y	3
q_x	2×10^5
q_y	3×10^5
n_{x_2}	50
n_{y_2}	8
x_e	$0.7L_x$
y_e	$0.4L_y$

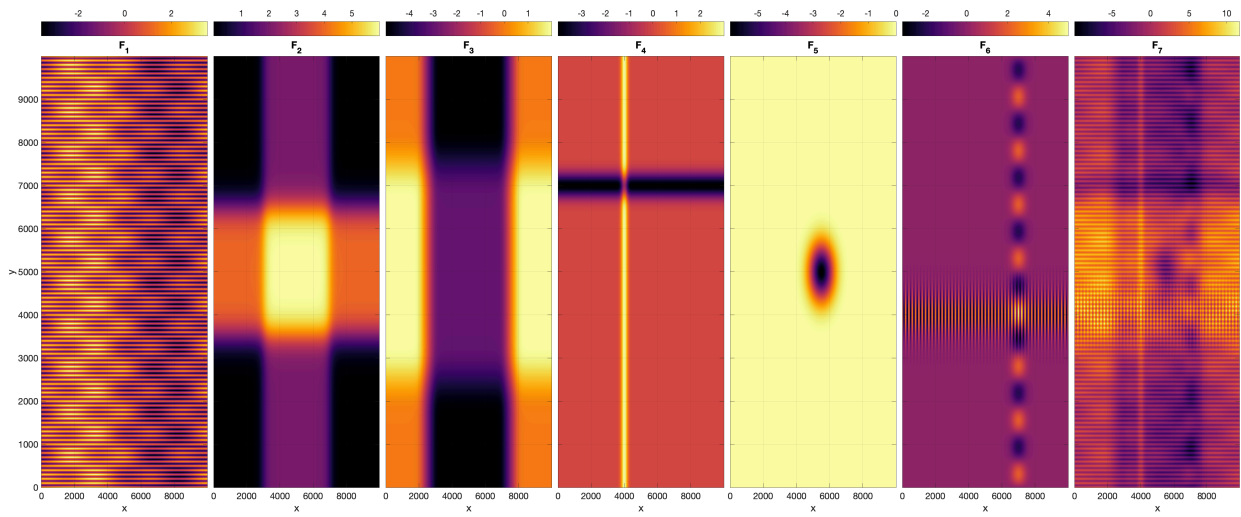


Figure B.1: Plots of F_1 to F_7 (Eqs. B.1-B.7)

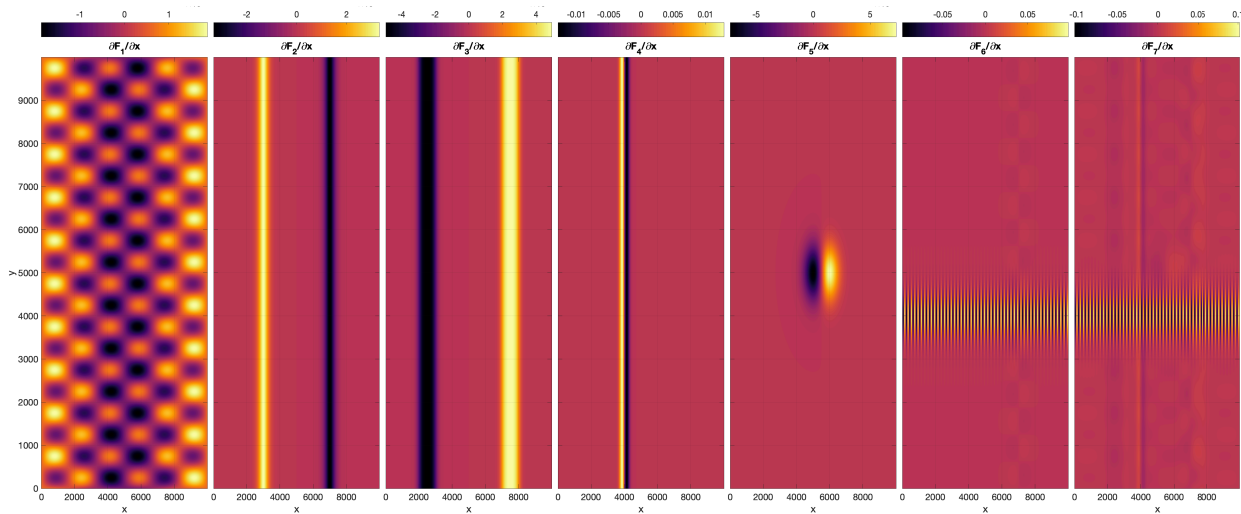


Figure B.2: Plots of exact x derivatives of F_1 to F_7 (Eqs. B.8-B.14)

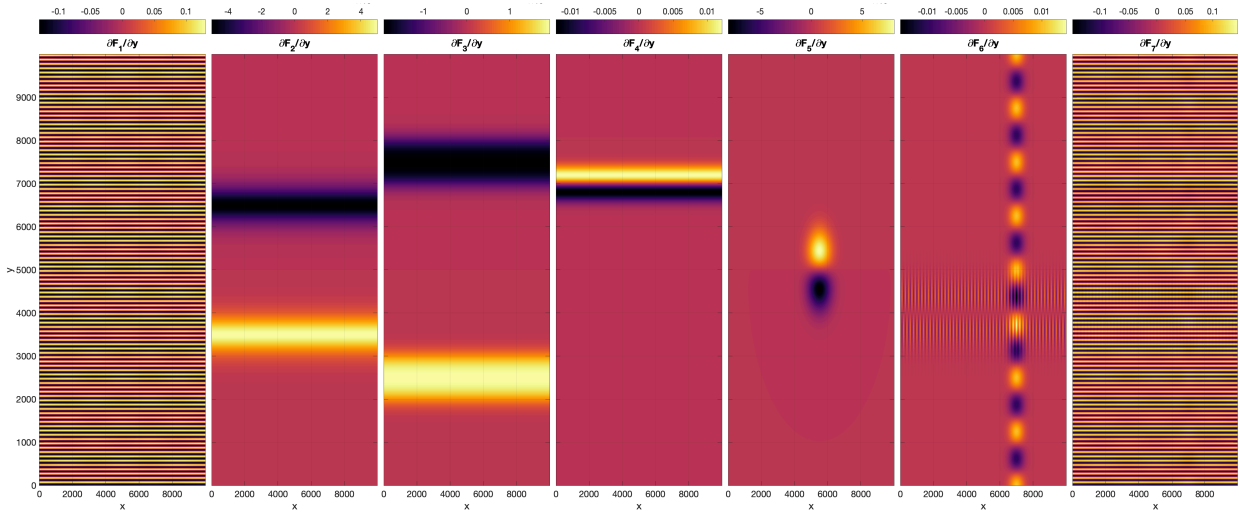


Figure B.3: Plots of exact y derivatives of F_1 to F_7 (Eqs. B.15-B.21)

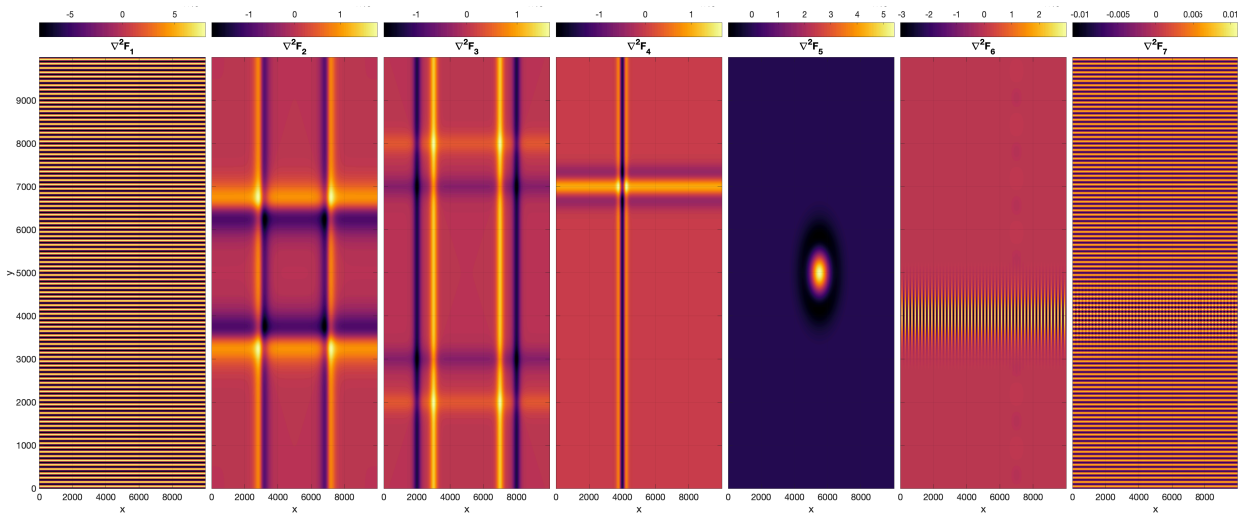


Figure B.4: Plots of exact Laplacians of F_1 to F_7 (Eqs. B.22-B.28)

B.2 Iterative Solver

This section lays out in greater detail the functions used to understand the discretization error when using the iterative solver. The iterative solver solves Eq. 3.136 for $\partial\phi/\partial t$. The results are described in Section 3.5.3. The manufactured solutions for n_{bg} , n_p , ϕ_{bg} , ϕ_p , and $\partial\phi_p/\partial t$ from Eq. 3.178, rewritten here for convenience, are all of the form

$$f(x, y) = \left[a_x \sin\left(\frac{2\pi n_x}{L_x} x\right) a_y \cos\left(\frac{2\phi n_y}{L_y} y\right) + a_{xy} \cos\left(\frac{2\phi n_{xy1}}{L_x} x\right) \sin\left(\frac{2\pi n_{xy2}}{L_y} y\right) + a_0 \right] v_0, \quad (\text{B.29})$$

where a_0 and v_0 are constant input coefficients. Table B.2 shows the coefficients used for this test with the remaining parameters defined in Table B.3. Figure B.5 shows the exact solution of $\partial\phi_p/\partial t$ (left) and the source term (right) generated from these solutions.

Table B.2: A table of the coefficients for different manufactured solutions of the form of Eq. B.29.

Variable	a_x	a_y	a_{xy}	n_x	n_y	n_{xy1}	n_{xy2}	a_0	v_0
n_{bg}	2	4	1	5	6	2	4	10	$3 \times 10^{11} \text{ m}^{-3}$
n_p	1	2	0.7	1	10	3	10	8	10^{11} m^{-3}
ϕ_{bg}	2.7	0.1	1	1	12	6	7	0	2000B V
ϕ_p	2	1	0.8	2	10	2	8	0	1000B V
$\partial\phi_p/\partial t$	1	3	2	3	13	1	5	0	2000BV/s

Table B.3: A table of the auxiliary parameters for testing the iterative solver.

Variable	Value
$[L_x, L_y]$	$[10^4, 10^4] \text{ m}$
B	$5 \times 10^{-5} \text{ T}$
\mathbf{u}	$[-500, 300, 0] \text{ m/s}$
$r_n = r_i$	152 pm
m_i	16 amu
n_n	10^{14} m^{-3}
aliasType	1: Zero-Padding

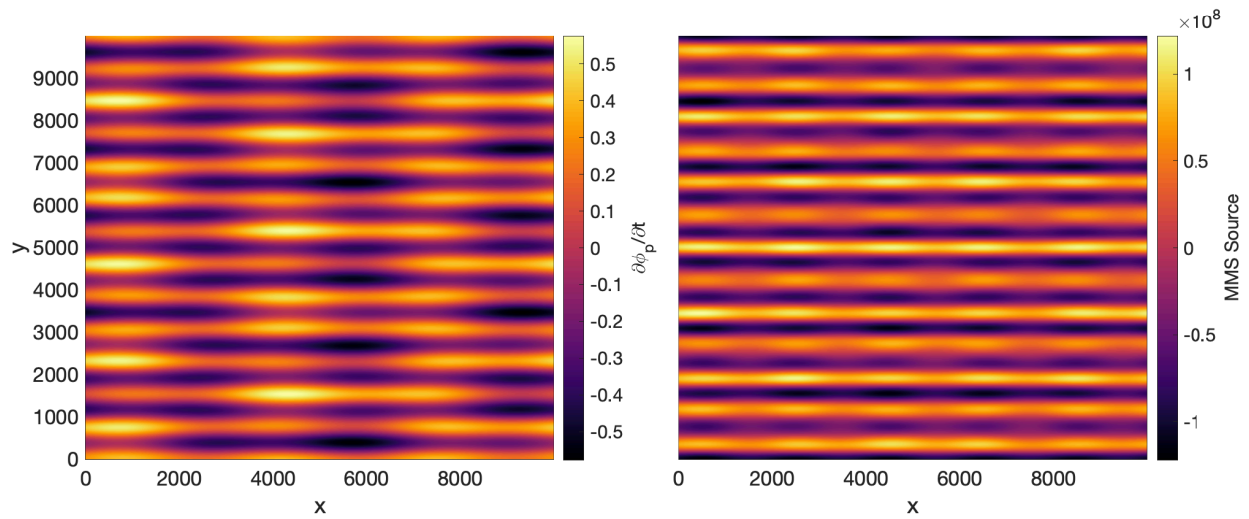


Figure B.5: The left panel shows the exact solution for $\partial\phi_p/\partial t$ and the right panel shows the method of manufactured solutions source term. The source is found by substituting Eq. B.29 into Eq. 3.136 with the appropriate parameters from Table B.2.

Appendix C

List of Input Files

This section discusses the different input files that are used for all of the simulations as well as an in depth understanding of the initial conditions and background profiles used.

C.1 GDI_tanh2N_noV

This section discusses a GDI simulation with no background velocity. The ion and electron temperatures are set as a constant uniform background. The density is initialized using 2 hyperbolic tangent functions.

C.1.1 Initial Conditions

The density is initialized using a set of hyperbolic tangent functions in x . This is a useful set of functions due to their ability to set regions of different density with a continuous gradient between them to limit numerical error. The parameters that define the function are the outer bound (o), the middle bound (m), the length scaling factors in meters ($L_g^{L,R}$), the gradient locations in meters ($x_g^{L,R}$), and a reference density in m^{-3} (n_0). The functional form of the set of hyperbolic tangents is

$$n_{bg}(x) = n_0 \left[a^L \tanh \left(\frac{x - x_g^L}{L_g^L} \right) + a^R \tanh \left(\frac{x - x_g^R}{L_g^R} \right) + d \right], \quad (\text{C.1})$$

where a^L , a^R , and d are constants that are defined by o and m as

$$a^L = \frac{m - o}{2} \quad (\text{C.2})$$

$$a^R = \frac{o - m}{2} \quad (\text{C.3})$$

$$d = o. \quad (\text{C.4})$$

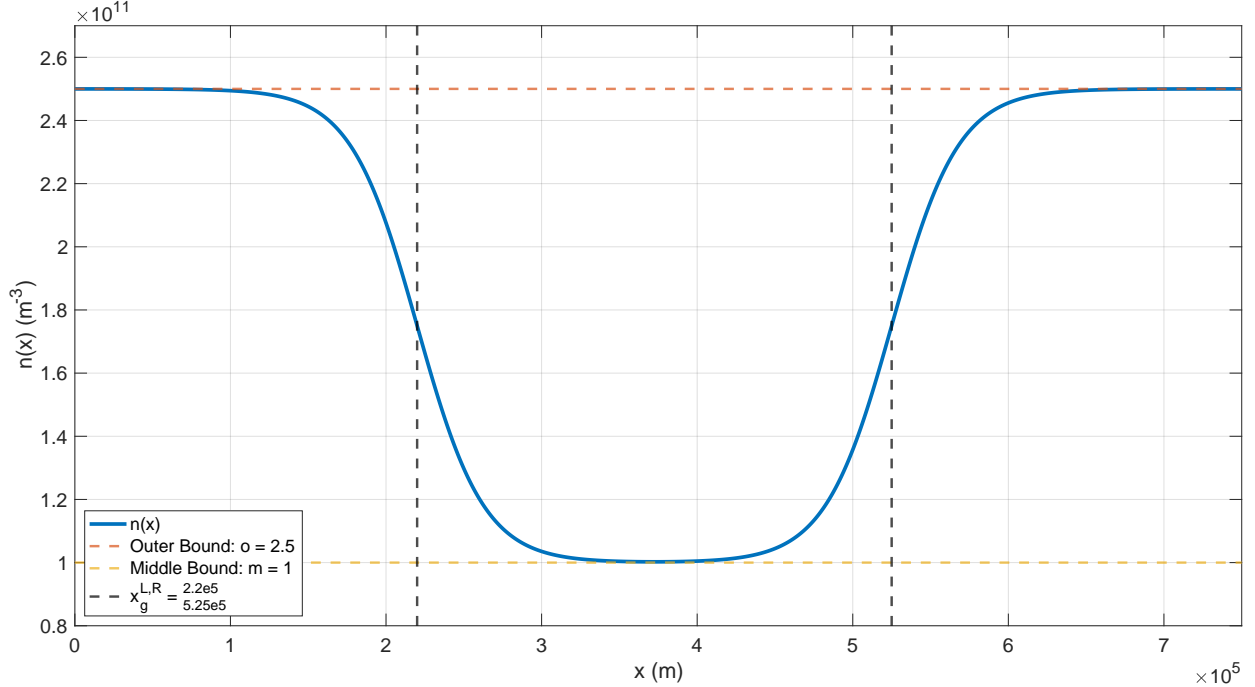


Figure C.1: An example of the background density profile as a function of x . The blue curve represents Eq. C.1 with $o = 2.5$, $m = 1$, $x_g^L = 2.2 \times 10^5$ m, $x_g^R = 5.25 \times 10^5$ m, $L_g^L = L_g^R = 4.3 \times 10^4$ m and $n_0 = 10^{11}$ m $^{-3}$. The red dashed line indicates the user-defined outer bound, o . The green dashed line indicates the user-defined middle bound, m . The black dashed lines indicate the locations of the density gradients, $x_g^{L,R}$. The width of the gradient regions is defined by $L_g^{L,R}$.

The outer and middle bounds must be any number greater than 0 to prevent negative density issues. If one of these is defined too close to 0, it is possible that the density can fall below 0 during the simulation, resulting in a crash. The condition $x_g^L < x_g^R$ must be satisfied to ensure the initial condition behaves as intended. The width of the gradient region, i.e., the strength of the derivative, is defined by $L_g^{L,R}$. A larger $L_g^{L,R}$ increases the gradient region (decreases the strength of the derivative) and a smaller $L_g^{L,R}$ decreases the gradient region (increases the strength of the derivative). Figure C.1 shows an example plot of Eq. C.1 as a function of x with $o = 2.5$, $m = 1$, $x_g^L = 2.2 \times 10^5$ m, $x_g^R = 5.25 \times 10^5$ m, $L_g^L = L_g^R = 4.3 \times 10^4$ m, and $n_0 = 10^{11}$ m $^{-3}$, with dashed lines annotating the effect of different parameters.

C.1.2 Input File

Listing C.1: Input file for a GDI simulation that approximates the density with 2 hyperbolic tangent functions and has no background velocity.

```

1  % This script is for a GDI simulation with 2 tanh functions to initialize the density
2  % and with no initial velocity
3
4  % Note: you will want to run this script in the directory in which you want your data to
   % be saved
5
6  % Close and clear everything;
7  close all; clearvars; clc;
8
9  % Set up the path here. This needs to be changed depending on where you have the code
   % saved
10 addpath('location/to/spacesciencescodes/matlabCodes/keskinen/');
11
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 % Everything in this section is for setting up the simulation.
14 % These parameters are what you want to change for running different simulations.
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % Save data every saveFrequency number of time steps.
18 % For more frames, decrease this number.
19 % For fewer frames, increase this number.
20 saveFrequency = 20;
21
22 % This is the maximum allowable time step in s.
23 % This makes sure that the time step does not skip over important physics
24 % This is set with some a priori understanding of the simulation
25 % It might need to be iterated upon and refined to get an idea of what's going on.
26 dt_max = 20;
27
28 % This is the minimum allowable time step in s.
29 % It is possible that due to the electric potential diverging, the velocity goes to
   % infinity.
30 % This results in the time step decreasing to 0.
31 % This can result in the simulation taking a long time despite nothing substantial
   % happening
32 % This is used as a check to cancel the simulation since it means the potential is
   % blowing up.
33 dt_min = 1e-6;
34
35 % This is the end time in s.
36 % Set this based on when you think you will have enough temporal evolution to your liking
37 tend = 4000;
38
39 % This is the maximum allowable error for the iterative method used to calculate the
   % electric potential

```

```
40 % Increasing this will yield better results but at the cost of additional computational
    effort
41 % It has not been tested yet but setting this too low will result in the required error
    to be below machine precision.
42 % That would result in the potential never being able to be solved (since the error can
    only go as low as machine precision)
43 err_max = 1e-8;
44
45 % Set the Courant-Friedrichs-Lewy condition for numerical stability
46 % This is dependent on the time integration scheme used
47 % Typically, this is left at 0.9 or 1.
48 % Note that there is additional physics that exist in this system of equations
49 % When calculating the time step, the CFL might be met but it might not consider some
    other important physical time scale
50 CFL = 1;
51
52 % Set the domain length.
53 % First element is the domain length in x direction
54 % Second element is the domain length in y direction
55 L = [3e5 2.5e5];
56
57 % Set the number of grid points in each direction
58 % First element is the number of grid points in the x direction
59 % Second element is the number of grid points in the y direction
60 n = [256 512];
61
62 % Set the artificial diffusion constant for the density source term
63 % This is needed to dissipate numerical error
64 % Note that as this number is increased, everything will generally take longer to run
65 % If this is increased too high, from experience, the simulation might crash.
66 % The solution is to decrease dt_max (however, this results in the code taking very long
    to run).
67 D = 1e3;
68
69 % This is a flag to choose which type of Fourier mesh to use.
70 % 0 means using the exact definition of k
71 % 1 means using the approximation of k and k^2
72 % See the makeFourierMesh2D function for more information on this.
73 % Depending on the initial conditions chosen, mesh type 1 might need to be chosen to get
    far enough in time
74 FourierMeshType = 1;
75
76 % This is a flag to determine the type of de-aliasing algorithm to use
77 % 0 means using no de-aliasing algorithm
78 % 1 means using the zero pad method (more computationally expensive)
79 % See the convolve2D function for more information on this
80 aliasType = 0;
81
82 % Set the magnetic field in T
83 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
```

```

84 % This code really only works for B purely in the z direction (third element) at the
      moment
85 % So, only initialize the third element and leave the first two 0.
86 B = [0 0 5e-5];
87
88 % Set the neutral wind in m/s.
89 % It is defined as a vector but only parts of the code are general enough to deal with
      all elements
90 % This code really only works for u purely in the xy plane (first and second elements)
      at the moment
91 % Soe only initialize the first two elements (x and y respectively) and leave the third
      one 0.
92 u = [-500 0 0];
93
94 % The background potential is set to be 0 for this simulation
95 phi0 = 0;
96
97 % The density is initialized using a set of 2 hyperbolic tangent functions
98 % If we consider the normalized density, this variable determines the value to set in
      the middle region
99 % Typically, this is kept as one but can be any arbitrary number that is greater than 0
100 middle = 1;
101
102 % If we consider the normalized density, this variable detemrines the value to set in
      the outer regions
103 % This can be any arbitrary number that is greater than 0
104 outer = 2.5;
105
106 % Set the location for the left density gradient in meters
107 % Note that this location has to be further left in x than xgN_right
108 xgN_left = .75e5;
109
110 % Set the location for the right density gradient in meters
111 % Note that this location has to be further right in x than xgN_lefttt
112 xgN_right = 2.2e5;
113
114 % Set the denominator term for the left density gradient in meters
115 % Note if this is too small, this can result in fast accumulation of numerical error
116 % If this is too big, the density might not be periodic enough within the domain and the
      code will crash
117 LgN_left = 3e4;
118
119 % Set the denominator term for the right density gradient in meters
120 % Note if this is too small, this can result in fast accumulation of numerical error
121 % If this is too big, the density might not be periodic enough within the domain and the
      code will crash
122 % Note that this is not required to be the same as LgN_left (it just is in this case).
123 LgN_right = 3e4;
124
125 % Set the level of the background density.
126 % The normalized set of hyperbolic tangents are multiplied by n0 to give the density

```

```

127 n0 = 1e11;
128
129 % Depending on how xg_left, xg_right, L_left, and L_right are set, you might have a
      function that
130 % does not quite look like a hyperbolic tangent.
131 % Play around with the different parameters to determine what works best for your
      simulation
132
133 % Set a constant ion temperature in K
134 % The ion temperature is assumed to be constant in the entire domain
135 Ti0 = 1000;
136
137 % Set a constant electron temperature in K
138 % The electron temperature is assumed to be constant in the entire domain
139 Te0 = 1000;
140
141 % Make a vector of the background constant multiplicative values (all the 0 variables)
      for each primitive variable
142 % This is used when calculating the perturbation
143 var_0 = [phi0, n0, Ti0, Te0];
144
145 % Set the neutral number density in 1/m^3
146 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
147 % See the calcCollFreqk function for more information.
148 % Note that the model assumes F region ionosphere, therefore if nn is too large, the
      code will crash
149 % The assumption that nu/Omega is small will break.
150 nn = 1e14;
151
152 % Set the radius of the neutral particles in m
153 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
154 % See the calcCollFreqk function for more information.
155 rn = 152e-12; % This specific value assumed O is dominant neutral species
156
157 % Set the radius of the ions in m
158 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
159 % See the calcCollFreqk function for more information.
160 ri = 152e-12; % This specific value assumed O+ is dominant ion species
161
162 % Set the mass of the neutrals in kg
163 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
164 % See the calcCollFreqk function for more information.
165 mi = 16 * 1.66053906660e-27; % This specific value assumed O is dominant neutral species
166
167 % Set the mass of the ions in kg
168 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
169 % See the calcCollFreqk function for more information.
170 mn = 16 * 1.66053906660e-27; % This specific value assumed O+ is dominant ion species
171
172
173 % This is what is used to setup the initial perturbation for the instability growth

```



```

174 % See the pert function for more information
175 % For now, only have pertType be 0, 1, or 4. The other types aren't properly implemented
    yet
176 % Each pertType corresponds to a different type of perturbation to apply
177 % 0 is single mode
178 % 1 is multi mode
179 % 2 is integral of single mode (Note fully tested or implemented, DON'T USE!)
180 % 3 is integral of multimode (Note fully tested or implemented, DON'T USE!)
181 % 4 is random noise
182 % 5 is multimode times multimode (Note fully tested or implemented, DON'T USE!)
183 % 6 is integral of multimode times multimode (Note fully tested or implemented, DON'T
    USE!)
184 pertType = 4;
185
186 % Define pertParam depending on which perturbation is chosen
187 % The definition of what goes in pertParam is different for each pertType
188 switch pertType
189     case {0,2} % Single mode and integral of single mode
190         pertParam(1) = 1e-6; % Amplitude of x direction wave
191         pertParam(2) = 1e-6; % Amplitude of y direction wave
192         pertParam(3) = 1; % Mode to be perturbed in x
193         pertParam(4) = 1; % Mode to be perturbed in y
194
195     case {1,3,5,6} % Multi mode, integral of multimode, multimode*multimode, integral of
        multimode*multimode
196         pertParam(1) = 1.e-6; % Minimum random amplitude
197         pertParam(2) = 1.e-5; % Maximum random amplitude
198         pertParam(3) = 0; % Minimum random phase shift
199         pertParam(4) = 2*pi; % Maximum random phase shift
200         pertParam(5) = 64; % Number of modes to excite in x
201         pertParam(6) = 128; % Number of modes to excite in y
202
203     case 4 % Random noise
204         pertParam(1) = -1.e-3; % Minimum random noise
205         pertParam(2) = 1.e-3; % Maximum random noise
206         pertParam(3) = 0; % If 1, only apply random noise to left half of domain
207 end
208
209 % This determines which variable to add the perturbation to (phi, n, Ti, or Te) (1, 2,
    3, or 4)
210 % For this type of simulation (GDI with 2 tanh for density and no velocity), always
    perturb density
211 % Note that pertVar can be a vector allowing multiple variables to be perturbed
212 pertVar = [2];
213
214 % Depending on the perturbation, we may want to multiply it by a Gaussian function so
    that it acts only at a particular location
215 % The following are the parameters for the Gaussian function
216
217 % This sets the width of the left Gaussian (if applicable) in 1/m
218 % This is presently defined based on LgN_left but it does not have to be

```

```

219 gaussParam(1) = 2 / LgN_left;
220
221 % This sets the width of the right Gaussian (if applicable) in 1/m
222 % This is presently defined based on LgN_right but it does not have to be
223 gaussParam(2) = 2 / LgN_right;
224
225 % This is a switch for the left Gaussian.
226 % 0 is off
227 % 1 is on
228 gaussParam(3) = 0;
229
230 % This is a switch for the right Gaussian.
231 % 0 is off
232 % 1 is on
233 gaussParam(4) = 1;
234
235 % Note that if both the left and the right are switched off, then the code assumes you
    want
236 % the perturbation to exist in the entire domain.
237 % Therefore, it multiplies the perturbation by 1 instead of a set of Gaussians
238
239 % This sets the location of the left Gaussian in m
240 % It is presently based on xgN_left but it does not have to be
241 gaussParam(5) = xgN_left;
242
243 % This sets the location of the right Gaussian in m
244 % It is presently based on xgN_right but it does not have to be
245 gaussParam(6) = xgN_right;
246
247 % This sets the direction we want to apply the perturbation
248 % i.e. is the perturbation a function of x or y?
249 % First index is x, second is y.
250 % Both can be set to 0 or 1 (off or on).
251 pertDir = [0 1];
252
253 % The code calculates the collision frequencies self-consistently
254 % There may be situations in which you want to specify a constant set of frequencies
255 % constCollSwitch determines if you want to use the self-consistent or pre-defined
    frequencies
256 % 0 is off which means that you are using the self-consistent collision frequencies
257 % 1 is on which means that you are using the pre-defined constant collision frequencies
258 constCollSwitch = 0;
259
260 % If constCollSwitch is 0, then neglect the next few lines
261 % If constCollSwitch is 1, then set the collision frequencies
262 nuin = 0; % ion-neutral
263 nuii = 0; % ion-ion
264 nuie = 0; % ion-electron
265 nuen = 0; % electron-neutral
266 nuei = 0; % electron-ion
267 nuee = 0; % electron-electron

```

```

268
269 % The calcCollFreqk function takes all of the above set of a parameters in as a single
      vector
270 constColls = [constCollSwitch , nuin , nuui , nuie , nuen , nuei , nuee];
271
272 % This is a flag to determine if the ion-neutral frictional heating source terms will be
      included
273 % 0 means off
274 % 1 means on
275 inHeatingSwitch = 1;
276
277 % Set the neutral temperature in K
278 % This is only needed if inHeatingSwitch == 1
279 % Otherwise, this is not used in the code
280 % Note that for the code to work properly, this needs to be a 2D matrix
281 % This can allow for non-uniform neutral temperatures
282 Tn = 1000 * ones(n);
283
284 % This is a flag to determine if the ion-electron frictional heating source terms will
      be included
285 % 0 means off
286 % 1 means on
287 ieHeatingSwitch = 0;
288
289 % This is a flag to determine if the thermal conduction source terms will be included
290 % 0 means off
291 % 1 means on
292 condSwitch = 0;
293
294 % Choose frame to restart.
295 % 0 means that the simulation will start from the beginning and make the initial
      conditions, then run.
296 % For any other integer (> 0), it will go to that frame and rerun the simulation from
      that start point
297 restartFrame = 0;
298
299 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
300 % Everything past here does not need to be changed to run this simulation.
301 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
302
303 % Set the counter to the frame to restart so that new frames are saved accordingly and
      not overwritng old ones
304 counter = restartFrame;
305
306 % Setup max time iterations so that the code has some finite end iteration.
307 iter_max = 100000;
308
309
310 % Setup max iterations for potential equation.
311 % If the iterations go over this, the code will stop because the potential is not
      converging.

```

```

312 phi_iter_max = 500;
313
314 % Set fundamental physical constants
315 me = 9.1093837015e-31; % kg
316 e = 1.602176634e-19; % C
317 kb = 1.380649e-23; % J/K
318 eps0 = 8.8541878128e-12; % F/m
319
320 % Get computational time
321 tic;
322
323 % Calculate the magnitude of B and its square
324 Bmag = norm(B);
325 B2 = Bmag^2;
326
327 % Calculate gyrofrequencies
328 Oce = e * Bmag / me;
329 Oci = e * Bmag / mi;
330
331 % Calculate Cm which is a constant that shows up frequently in the potential source term
332 Cm = (1 / Oci + 1/Oce)^-1;
333
334 % Make spatial mesh
335 [XX, YY, kmax] = makeSpatialMesh2D(L, n);
336
337 % Get a vector of just x
338 xVec = XX(:,1);
339
340 % Get a vector of just y
341 yVec = YY(1,:)';
342
343 % Set the indices of the two regions from which to get the E field energy spectrum
344 domain_inds{1} = 1:n(1)/4;
345 domain_inds{2} = ( n(1)/4+1 ) : n(1)/2;
346
347 % Set fileIDs and names of energy spectrum data
348 fileIDs(1) = fopen('Eenergy_left.txt','w');
349 fileIDs(2) = fopen('Eenergy_right.txt','w');
350
351 % Make a Fourier mesh
352 [k, ksqu, ksqu_4inv] = makeFourierMesh2D(L, n, FourierMeshType);
353
354
355 % Set fileIDs for time vector
356 timeFileID = fopen('tVec.txt','w');
357
358 % Note that the definition of prim_var is
359 % 1: phi
360 % 2: ne
361 % 3: Ti
362 % 4: Te

```

```

363 % 5: Pi
364 % 6: Pe
365
366 % Make the names for each primitive variable
367 varNames = {'phi' , 'ne' , 'Ti' , 'Te' , 'Pi' , 'Pe'};
368
369 % Check if we are restarting or starting a new simulations
370 if restartFrame == 0 % This means we are starting a new simulation
371
372     % Set the initial conditions for the background
373
374     % Preallocate the unperturbed variables as all zeros
375     prim_var_unpert = zeros([n 6]);
376
377     % The electric potential is already initialized to be 0
378
379     % Set the background density
380     % They are based on a set of 2 hyperbolic tangent functions
381     % See the tanh function for more information
382     prim_var_unpert(:,:,2) = tanhIC(XX, outer, middle, [xgN_left, xgN_right], ...
383         [LgN_left, LgN_right], n0);
384
385     % Set the background ion temperature
386     prim_var_unpert(:,:,3) = Ti0;
387
388     % Set the background electron temperature
389     prim_var_unpert(:,:,4) = Te0;
390
391
392     % Preallocate all perturbations as all zeros
393     prim_var_pert = zeros([size(XX) 6]);
394
395     % Iterate through all of the variables to be perturbed
396     for j = 1:length(pertVar)
397         % Initialize some perturbation
398         prim_var_pert(:,:,pertVar(j)) = pert(XX, YY, L, pertParam, ...
399             pertType, gaussParam, pertDir, var_0(pertVar(j)) );
400     end
401
402     % Get total variable by adding background and perturbation
403     prim_var(:,:,1:4) = prim_var_unpert(:,:,1:4) + prim_var_pert(:,:,1:4);
404
405     % Calculate pressures
406     % Unperturbed
407     prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
408     prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
409
410     % Total
411     prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
412     prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
413

```

```

414 % Perturbed
415 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
416 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
417
418 % Save data
419 save3DMat(prim_var_unpert(:,:,1:4), varNames, '_unpert');
420 save3DMat(prim_var_pert(:,:,1:4), varNames, '_0');
421
422 % Initialize a time vector
423 t = 0;
424 iter_start = 1;
425
426 save('t_0.txt','t','-ascii');
427
428 % Save mesh
429 save('X.txt','XX','-ascii');
430 save('Y.txt','YY','-ascii');
431
432 % Save u and B
433 save('u.txt','u','-ascii');
434 save('B.txt','B','-ascii');
435
436 % Save ion mass
437 save('mi.txt','mi','-ascii');
438
439 % Save Fourier mesh type
440 save('FourierMeshType.txt','FourierMeshType','-ascii');
441
442 % If using constant collision frequencies, save them here
443 if constCollSwitch == 1
444     collParam = [nuin , nuie , nuii , nuen , nuee , nuei];
445     save('collisions.txt','collParam','-ascii');
446 end
447
448 % Save other parameters
449 savedParam = [mi mn FourierMeshType nn ri rn];
450 save('param.txt','savedParam','-ascii');
451
452 % Save diffusion constants
453 save('D.txt','D','-ascii');
454
455 % Make and write to a log file
456 fid = fopen('log.txt','w');
457 fprintf(fid, 'L = [%.2e %.2e]\n', L(1), L(2));
458 fprintf(fid, 'n = [%d %d]\n', n(1), n(2));
459 fprintf(fid, 'B = [%.2e %.2e %.2e]\n', B(1), B(2), B(3));
460 fprintf(fid, 'u = [%.2e %.2e %.2e]\n', u(1), u(2), u(3));
461 fprintf(fid, 'FourierMeshType = %d\n', FourierMeshType);
462 fprintf(fid, 'Dn = %.2e\n', D);
463 fprintf(fid, 'mi = %.2e\n', mi);
464 if constColls(1) == 1

```

```

465     fprintf(fid, 'nuin = %.2e\n', nuin);
466     fprintf(fid, 'nuie = %.2e\n', nuie);
467     fprintf(fid, 'nuen = %.2e\n', nuen);
468     fprintf(fid, 'nuei = %.2e\n', nuei);
469     fprintf(fid, 'nuui = %.2e\n', nuei);
470     fprintf(fid, 'nuee = %.2e\n', nuei);
471 else
472     fprintf(fid, 'mn = %.2e\n', mn);
473     fprintf(fid, 'nn = %.2e\n', nn);
474     fprintf(fid, 'ri = %.2e\n', ri);
475     fprintf(fid, 'rn = %.2e\n', rn);
476 end
477
478 if aliasType == 0
479     fprintf(fid, '\nUsing no dealiasing algorithm\n\n');
480 elseif aliasType == 1
481     fprintf(fid, '\nUsing zero-padding dealiasing\n\n');
482 end
483 fclose(fid);
484
485 else % If restarting from a dataset
486     % Load variables
487     for j = 1:4
488         prim_var_pert(:, :, j) = load([varNames{j} '_' num2str(counter) '.txt']);
489         prim_var_unpert(:, :, j) = load([ varNames{j} '_unpert.txt']);
490     end
491
492     % Get total variables
493     prim_var = prim_var_pert + prim_var_unpert;
494
495     % Calculate pressures
496     % Unperturbed
497     prim_var_unpert(:, :, 5) = kb * prim_var_unpert(:, :, 2) .* prim_var_unpert(:, :, 3);
498     prim_var_unpert(:, :, 6) = kb * prim_var_unpert(:, :, 2) .* prim_var_unpert(:, :, 4);
499
500     % Total
501     prim_var(:, :, 5) = kb * prim_var(:, :, 2) .* prim_var(:, :, 3);
502     prim_var(:, :, 6) = kb * prim_var(:, :, 2) .* prim_var(:, :, 4);
503
504     % Perturbed
505     prim_var_pert(:, :, 5) = prim_var(:, :, 5) - prim_var_unpert(:, :, 5);
506     prim_var_pert(:, :, 6) = prim_var(:, :, 6) - prim_var_unpert(:, :, 6);
507
508     % Loading time data
509     t = load(['t_' num2str(restartFrame) '.txt']);
510
511     timeFileID = fopen('tVec.txt', 'a');
512
513     % Open electric field energy data
514     fileIDs(1) = fopen('Eenergy_left.txt', 'a');
515     fileIDs(2) = fopen('Eenergy_right.txt', 'a');

```

```

516
517     % Set the start of the iterations
518     iter_start = 1;
519 end
520
521 % Run simulation
522 runSimulation_inertia_RK4(prim_var_unpert, prim_var_pert, k, ksqu, ksqu_4inv, kmax, t,
    CFL, dt_min, ...
523     dt_max, tend, u, B, mi, ri, mn, nn, rn, D, iter_start, iter_max, err_max, ...
524     phi_iter_max, saveFrequency, counter, varNames, inHeatingSwitch, ieHeatingSwitch, ...
525     constColls, condSwitch, aliasType, xVec, yVec, domain_inds, fileIDs, timeFileID,
    pertType);
526
527 % Close file IDs
528 fclose('all');
529
530 % Get computational time
531 elapsed_time = toc;
532
533 % Finish up some logging
534 fid = fopen('log.txt','a');
535 fprintf('Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
536 fprintf(fid,'Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
537 fprintf('\n\n\n\n\n\n\n\n');
538 fprintf('Done!\n\n');

```

C.2 GDI_tanh4N_sech2V

This section discusses a GDI simulation that uses a set of 4 hyperbolic tangent functions to model the background density and 2 hyperbolic secant squared functions to model the background velocity.

C.2.1 Initial Conditions

Eq. C.1 has the feature that the outer bounds on the left and the right are the same. For some simulations, however, this may not be ideal, and having different bounds on the left and right might be the correct representation of the physical phenomenon being simulated. However, due to the choice of Fourier basis functions, the initial conditions are required to be periodic within the domain. One method to resolve this is by using a set of 4 hyperbolic tangent functions to maintain total domain periodicity while only considering the results in the desired half of the domain. The parameters that define this function are the left level (l_1), the middle level (l_2), the right level (l_3), the length scaling factors in meters ($L_{gN}^{L,R}$), the gradient locations in meters ($x_{gN}^{L,R}$), and a reference density in m^{-3} (n_0). The functional

form of this set of hyperbolic tangents is

$$n_{bg}(x) = n_0 \left[a^L \tanh \left(\frac{x - x_{gN}^L}{L_{gN}^L} \right) + a^R \tanh \left(\frac{x - x_{gN}^R}{L_{gN}^R} \right) - a^R \tanh \left(\frac{x - L_x + x_{gN}^R}{L_{gN}^R} \right) - a^L \tanh \left(\frac{x - L_x + x_{gN}^L}{L_{gN}^L} \right) + d \right], \quad (\text{C.5})$$

where L_x is the domain length in the x direction and a^L , a^R , and d are defined by l_1 , l_2 , and l_3 as

$$a^L = \frac{l_2 - l_1}{2} \quad (\text{C.6})$$

$$a^R = \frac{l_3 - l_2}{2} \quad (\text{C.7})$$

$$d = l_1. \quad (\text{C.8})$$

The $\hat{\mathbf{y}}$ direction velocity is modeled as a hyperbolic secant squared function in x . However, to maintain periodicity in the electric potential, the velocity needs to be modeled by two hyperbolic secant squared functions with different signs. The functional form of this is

$$V_{y_{bg}}(x) = V_0 \left[\text{sech}^2 \left(\frac{x - x_{gV}^L}{L_{gV}} \right) - \text{sech}^2 \left(\frac{x - x_{gV}^R}{L_{gV}} \right) \right], \quad (\text{C.9})$$

where $x_{gV}^{L,R}$ are the profile locations in meters, L_{gV} is the length scaling factor in meters, and V_0 is a reference velocity in m/s.

In the computational model, it is the electric potential that needs to be initialized as opposed to the velocity. The velocity from Eq. C.9 is assumed to be purely $\mathbf{E} \times \mathbf{B}$ driven. The 2D $\mathbf{E} \times \mathbf{B}$ drift is

$$\mathbf{V}_{\mathbf{E} \times \mathbf{B}} = -\frac{\nabla \phi \times \mathbf{B}}{B^2} = \frac{1}{B} \begin{bmatrix} -\frac{\partial \phi}{\partial y} \\ \frac{\partial \phi}{\partial x} \\ 0 \end{bmatrix} = \begin{bmatrix} V_x \\ V_y \\ 0 \end{bmatrix}, \quad (\text{C.10})$$

where $V_x = 0$. Therefore $V_y = B^{-1} \partial \phi / \partial x$. Thus, the electric potential is

$$\phi = B \int V_y dx. \quad (\text{C.11})$$

The electric potential initial condition is found by taking the integral of Eq. C.9, resulting in

$$\phi_{bg}(x) = BV_0 L_{gV} \left[\tanh \left(\frac{x - x_{gV}^L}{L_{gV}} \right) - \tanh \left(\frac{x - x_{gV}^R}{L_{gV}} \right) \right]. \quad (\text{C.12})$$

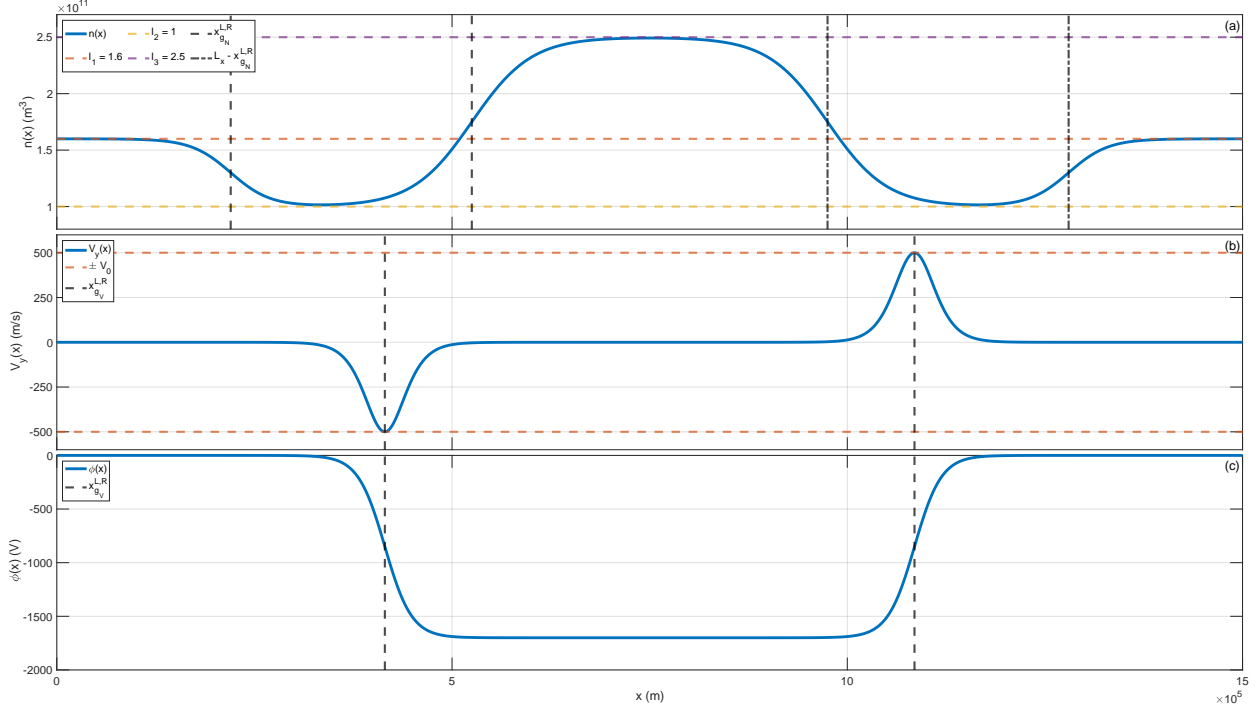


Figure C.2: An example of the background profiles as functions of x . The domain length, L_x , is 1.5×10^6 m. Panel (a) shows the background density profile (blue curve) using Eq. C.5, with $l_1 = 1.6$, $l_2 = 1$, $l_3 = 2.5$, $x_{g_N}^L = 2.2 \times 10^5$ m, $x_{g_N}^R = 5.25 \times 10^5$ m, $L_{g_N}^L = 5 \times 10^4$ m, $L_{g_N}^R = 7.5 \times 10^4$ m, and $n_0 = 10^{11}$ m $^{-3}$. The purple, red, and yellow dashed lines indicate the user-defined levels of the set of hyperbolic tangent functions. The black lines indicate the locations of the density gradients. Panel (b) shows the background velocity profile (blue curve) using Eq. C.9, with $x_{g_V}^L = 4.15 \times 10^5$ m, $x_{g_V}^R = 1.085 \times 10^6$ m, $L_{g_V} = 3.4 \times 10^4$ m, and $V_0 = -500$ m/s. The red dashed lines indicate the user-defined maximum velocity, V_0 . The black dashed lines indicate the locations of the velocity profiles. Panel (c) shows the background electric potential profile (blue curve) using Eq. C.12, with $B = 5 \times 10^{-5}$ T. The black dashed lines indicate the locations of the potential gradients.

All of the levels (l_1 , l_2 , and l_3) need to be greater than 0 to avoid negative density issues. Care should be taken to make sure that they are not too close to 0 such that, during simulation run time, the density becomes lower than 0. For both the density and velocity, the condition $x_g^L < x_g^R$ must be satisfied. The parameter L_{g_N} determines the width of the density gradients. The parameter L_{g_V} determines the width of both the velocity channel and the potential gradient. Figure C.2(a) shows an example plot of the density initial condition using Eq. C.5, with $l_1 = 1.6$, $l_2 = 1$, $l_3 = 2.5$, $x_{g_N}^L = 2.2 \times 10^5$ m, $x_{g_N}^R = 5.25 \times 10^5$ m, $L_{g_N}^L = 5 \times 10^4$ m, $L_{g_N}^R = 7.5 \times 10^4$ m, and $n_0 = 10^{11}$ m $^{-3}$. Figure C.2(b) shows an example plot of the velocity initial condition using Eq. C.9, with $x_{g_V}^L = 4.15 \times 10^5$ m, $x_{g_V}^R = 1.085 \times 10^6$ m, $L_{g_V} = 3.4 \times 10^4$ m, and $V_0 = -500$ m/s. Figure C.2(c) shows an example plot of the electric

potential initial condition using Eq. C.12, with $B = 5 \times 10^{-5}$ T. Each plot is annotated with the dashed lines to show the effects of some parameters.

C.2.2 Input File

Listing C.2: Input file for a GDI simulation that approximates the density with 4 hyperbolic tangent functions and approximates the velocity with 2 hyperbolic secant squared functions.

```

1  % This script is for a GDI simulation with 4 tanh functions to initialize the density
2  % and with a sech^2 function to initialize the velocity
3
4  % Note: you will want to run this script in the directory in which you want your data to
   % be saved
5
6  % Close and clear everything;
7  close all; clearvars; clc;
8
9  % Set up the path here. This needs to be changed depending on where you have the code
   % saved
10 addpath('location/to/spacesciencescodes/matlabCodes/keskinen/');
11
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 % Everything in this section is for setting up the simulation.
14 % These parameters are what you want to change for running different simulations.
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % Save data every saveFrequency number of time steps.
18 % For more frames, decrease this number.
19 % For fewer frames, increase this number.
20 saveFrequency = 100;
21
22 % This is the maximum allowable time step in s.
23 % This makes sure that the time step does not skip over important physics
24 % This is set with some a priori understanding of the simulation
25 % It might need to be iterated upon and refined to get an idea of what's going on.
26 dt_max = 20;
27
28 % This is the minimum allowable time step in s.
29 % It is possible that due to the electric potential diverging, the velocity goes to
   % infinity.
30 % This results in the time step decreasing to 0.
31 % This can result in the simulation taking a long time despite nothing substantial
   % happening
32 % This is used as a check to cancel the simulation since it means the potential is
   % blowing up.
33 dt_min = 1e-6;
34
35 % This is the end time in s.
36 % Set this based on when you think you will have enough temporal evolution to your liking

```

```
37 tend = 7500*2;
38
39 % This is the maximum allowable error for the iterative method used to calculate the
    electric potential
40 % Increasing this will yield better results but at the cost of additional computational
    effort
41 % It has not been tested yet but setting this too low will result in the required error
    to be below machine precision.
42 % That would result in the potential never being able to be solved (since the error can
    only go as low as machine precision)
43 err_max = 1e-8;
44
45 % Set the Courant-Friedrichs-Lewy condition for numerical stability
46 % This is dependent on the time integration scheme used
47 % Typically, this is left at 0.9 or 1.
48 % Note that there is additional physics that exist in this system of equations
49 % When calculating the time step, the CFL might be met but it might not consider some
    other important physical time scale
50 CFL = 1;
51
52 % Set the domain length.
53 % First element is the domain length in x direction
54 % Second element is the domain length in y direction
55 L = [1.5e6 1e6];
56
57 % Set the number of grid points in each direction
58 % First element is the number of grid points in the x direction
59 % Second element is the number of grid points in the y direction
60 n = [1024 256];
61
62 % Set the artificial diffusion constant for the density source term
63 % This is needed to dissipate numerical error
64 % Note that as this number is increased, everything will generally take longer to run
65 % If this is increased too high, from experience, the simulation might crash.
66 % The solution is to decrease dt_max (however, this results in the code taking very long
    to run).
67 D = 1e3;
68
69 % This is a flag to choose which type of Fourier mesh to use.
70 % 0 means using the exact definition of k
71 % 1 means using the approximation of k and k^2
72 % See the makeFourierMesh2D function for more information on this.
73 % Depending on the initial conditions chosen, mesh type 1 might need to be chosen to get
    far enough in time
74 FourierMeshType = 1;
75
76 % This is a flag to determine the type of de-aliasing algorithm to use
77 % 0 means using no de-aliasing algorithm
78 % 1 means using the zero pad method (more computationally expensive)
79 % See the convolve2D function for more information on this
80 aliasType = 0;
```

```

81
82 % Set the magnetic field in T
83 % It is defined as a vector but only parts of the code are general enough to deal with
      all elements
84 % This code really only works for B purely in the z direction (third element) at the
      moment
85 % So, only initialize the third element and leave the first two 0.
86 B = [0 0 5e-5];
87
88 % Set the neutral wind in m/s.
89 % It is defined as a vector but only parts of the code are general enough to deal with
      all elements
90 % This code really only works for u purely in the xy plane (first and second elements)
      at the moment
91 % Soe only initialize the first two elements (x and y respectively) and leave the third
      one 0.
92 u = [-500 0 0];
93
94 % The velocity is initialized using 2 sech^2 functions
95 % Set the left location for the sech^2 function in m
96 % Note that this has to be further left in x than xgPhi_right
97 xgPhi_left = 4.15e5;
98
99 % Set the right location for the sech^2 function in m
100 % Note that this has to be further right in x than xgPhi_left
101 % This case is defined based on the domain length but it does not have to be.
102 xgPhi_right = L(1) - 4.15e5;
103
104 % Set the denominator term for the potential profile in m
105 % Note that the left and right sech^2 need to have the same denominator
106 % This is because the electric potential also needs to be periodic and is the integral
      of the velocity
107 LgPhi = 3.4e4;
108
109 % Set the level of the background velocity in m/s
110 V0 = -500;
111
112 % Calculate the equivalent potential such that ExB = V0
113 % Note that this assumes a purely z direction B field
114 phi0 = V0 * B(3);
115
116 % The density is initialized using a set of 4 hyperbolic tangent functions
117 % If we consider the normalized density, this variable determines the left level
118 % This can be any arbitrary number that is greater than 0
119 levels(1) = 1.6;
120
121 % If we consider the normalized density, this variable determines the middle level
122 % This is typically 1, but it can be any arbitrary number that is greater than 0
123 levels(2) = 1;
124
125 % If we consider the normalized density, this variable determines the right level

```

```

126 % This can be any arbitrary number that is greater than 0
127 levels(3) = 2.5;
128
129 % Set the location for the left density gradient in meters
130 % Note that this location has to be further left in x than xgN_right
131 xgN_left = 2.2e5;
132
133 % Set the location for the right density gradient in meters
134 % Note that this location has to be further right in x than xgN_leftt
135 xgN_right = 5.25e5;
136
137 % Set the denominator term for the left density gradient in meters
138 % Note if this is too small, this can result in fast accumulation of numerical error
139 % If this is too big, the density might not be periodic enough within the domain and the
    code will crash
140 LgN_left = 5e4;
141
142 % Set the denominator term for the right density gradient in meters
143 % Note if this is too small, this can result in fast accumulation of numerical error
144 % If this is too big, the density might not be periodic enough within the domain and the
    code will crash
145 % Note that this is not required to be the same as LgN_left (it just is in this case).
146 LgN_right = 7.5e4;
147
148 % Set the level of the background density.
149 % The normalized set of hyperbolic tangents are multiplied by n0 to give the density
150 n0 = 1e11;
151
152 % Depending on how xg_left, xg_right, L_left, and L_right are set, you might have a
    function that
153 % does not quite look like a hyperbolic tangent.
154 % Play around with the different parameters to determine what works best for your
    simulation
155
156 % Set a constant ion temperature in K
157 % The ion temperature is assumed to be constant in the entire domain
158 Ti0 = 1000;
159
160 % Set a constant electron temperature in K
161 % The electron temperature is assumed to be constant in the entire domain
162 Te0 = 1000;
163
164 % Make a vector of the background constant multiplicative values (all the 0 variables)
    for each primitive variable
165 % This is used when calculating the perturbation
166 var_0 = [phi0, n0, Ti0, Te0];
167
168 % Set the neutral number density in 1/m^3
169 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
170 % See the calcCollFreqk function for more information.

```

```

171 % Note that the model assumes F region ionosphere, therefore if nn is too large, the
      code will crash
172 % The assumption that nu/Omega is small will break.
173 nn = 1e14;
174
175 % Set the radius of the neutral particles in m
176 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
177 % See the calcCollFreqk function for more information.
178 rn = 152e-12; % This specific value assumed O is dominant neutral species
179
180 % Set the radius of the ions in m
181 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
182 % See the calcCollFreqk function for more information.
183 ri = 152e-12; % This specific value assumed O+ is dominant ion species
184
185 % Set the mass of the neutrals in kg
186 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
187 % See the calcCollFreqk function for more information.
188 mi = 16 * 1.66053906660e-27; % This specific value assumed O is dominant neutral species
189
190 % Set the mass of the ions in kg
191 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
192 % See the calcCollFreqk function for more information.
193 mn = 16 * 1.66053906660e-27; % This specific value assumed O+ is dominant ion species
194
195
196 % This is what is used to setup the initial perturbation for the instability growth
197 % See the pert function for more information
198 % For now, only have pertType be 0, 1, or 4. The other types aren't properly implemented
      yet
199 % Each pertType corresponds to a different type of perturbation to apply
200 % 0 is single mode
201 % 1 is multi mode
202 % 2 is integral of single mode (Note fully tested or implemented, DON'T USE!)
203 % 3 is integral of multimode (Note fully tested or implemented, DON'T USE!)
204 % 4 is random noise
205 % 5 is multimode times multimode (Note fully tested or implemented, DON'T USE!)
206 % 6 is integral of multimode times multimode (Note fully tested or implemented, DON'T
      USE!)
207 pertType = 4;
208
209 % Define pertParam depending on which perturbation is chosen
210 % The definition of what goes in pertParam is different for each pertType
211 switch pertType
212     case {0,2} % Single mode and integral of single mode
213         pertParam(1) = 1e-6; % Amplitude of x direction wave
214         pertParam(2) = 1e-6; % Amplitude of y direction wave
215         pertParam(3) = 4; % Mode to be perturbed in x
216         pertParam(4) = 6; % Mode to be perturbed in y
217

```

```

218     case {1,3,5,6} % Multi mode, integral of multimode, multimode*multimode, integral of
                multimode*multimode
219         pertParam(1) = 1.e-6; % Minimum random amplitude
220         pertParam(2) = 1.e-6; % Maximum random amplitude
221         pertParam(3) = 0; % Minimum random phase shift
222         pertParam(4) = 2*pi; % Maximum random phase shift
223         pertParam(5) = 64; % Number of modes to excite in x
224         pertParam(6) = 128; % Number of modes to excite in y
225
226     case 4 % Random noise
227         pertParam(1) = -1.e-6; % Minimum random noise
228         pertParam(2) = 1.e-6; % Maximum random noise
229         pertParam(3) = 1; % If 1, only apply random noise to left half of domain.
230 end
231
232 % This determines which variable to add the perturbation to (phi, n, Ti, or Te) (1, 2,
    3, or 4)
233 % For this type of simulation (GDI with 2 tanh for density and no velocity), always
    perturb density
234 % Note that pertVar can be a vector allowing multiple variables to be perturbed
235 pertVar = [2];
236
237 % Depending on the perturbation, we may want to multiply it by a Gaussian function so
    that it acts only at a particular location
238 % The following are the parameters for the Gaussian function
239
240 % This sets the width of the left Gaussian (if applicable) in 1/m
241 % This is presently defined based on LgN_left but it does not have to be
242 gaussParam(1) = 2 / LgN_left;
243
244 % This sets the width of the right Gaussian (if applicable) in 1/m
245 % This is presently defined based on LgN_right but it does not have to be
246 gaussParam(2) = 2 / LgN_right;
247
248 % This is a switch for the left Gaussian.
249 % 0 is off
250 % 1 is on
251 gaussParam(3) = 0;
252
253 % This is a switch for the right Gaussian.
254 % 0 is off
255 % 1 is on
256 gaussParam(4) = 1;
257
258 % Note that if both the left and the right are switched off, then the code assumes you
    want
259 % the perturbation to exist in the entire domain.
260 % Therefore, it multiplies the perturbation by 1 instead of a set of Gaussians
261
262 % This sets the location of the left Gaussian in m
263 % It is presently based on xgN_left but it does not have to be

```



```

264 gaussParam(5) = xgN_left;
265
266 % This sets the location of the right Gaussian in m
267 % It is presently based on xgN_right but it does not have to be
268 gaussParam(6) = xgN_right;
269
270 % This sets the direction we want to apply the perturbation
271 % i.e. is the perturbation a function of x or y?
272 % First index is x, second is y.
273 % Both can be set to 0 or 1 (off or on).
274 pertDir = [0 1];
275
276 % The code calculates the collision frequencies self-consistently
277 % There may be situations in which you want to specify a constant set of frequencies
278 % constCollSwitch determines if you want to use the self-consistent or pre-defined
      frequencies
279 % 0 is off which means that you are using the self-consistent collision frequencies
280 % 1 is on which means that you are using the pre-defined constant collision frequencies
281 constCollSwitch = 0;
282
283 % If constCollSwitch is 0, then neglect the next few lines
284 % If constCollSwitch is 1, then set the collision frequencies
285 nuin = 0; % ion-neutral
286 nuii = 0; % ion-ion
287 nuie = 0; % ion-electron
288 nuen = 0; % electron-neutral
289 nuei = 0; % electron-ion
290 nuee = 0; % electron-electron
291
292 % The calcCollFreqk function takes all of the above set of a parameters in as a single
      vector
293 constColls = [constCollSwitch , nuin , nuii , nuie , nuen , nuei , nuee];
294
295 % This is a flag to determine if the ion-neutral frictional heating source terms will be
      included
296 % 0 means off
297 % 1 means on
298 inHeatingSwitch = 0;
299
300 % Set the neutral temperature in K
301 % This is only needed if inHeatingSwitch == 1
302 % Otherwise, this is not used in the code
303 % Note that for the code to work properly, this needs to be a 2D matrix
304 % This can allow for non-uniform neutral temperatures
305 Tn = 1000 * ones(n);
306
307 % This is a flag to determine if the ion-electron frictional heating source terms will
      be included
308 % 0 means off
309 % 1 means on
310 ieHeatingSwitch = 0;

```

```

311
312 % This is a flag to determine if the thermal conduction source terms will be included
313 % 0 means off
314 % 1 means on
315 condSwitch = 0;
316
317 % Choose frame to restart.
318 % 0 means that the simulation will start from the beginning and make the initial
      conditions, then run.
319 % For any other integer (> 0), it will go to that frame and rerun the simulation from
      that start point
320 restartFrame = 0;
321
322 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
323 % Everything past here does not need to be changed to run this simulation.
324 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
325
326 % Set the counter to the frame to restart so that new frames are saved accordingly and
      not overwritng old ones
327 counter = restartFrame;
328
329 % Setup max time iterations so that the code has some finite end iteration.
330 iter_max = 100000;
331
332
333 % Setup max iterations for potential equation.
334 % If the iterations go over this, the code will stop because the potential is not
      converging.
335 phi_iter_max = 500;
336
337 % Set fundamental physical constants
338 me = 9.1093837015e-31; % kg
339 e = 1.602176634e-19; % C
340 kb = 1.380649e-23; % J/K
341 eps0 = 8.8541878128e-12; % F/m
342
343 % Get computational time
344 tic;
345
346 % Calculate the magnitude of B and its square
347 Bmag = norm(B);
348 B2 = Bmag^2;
349
350 % Calculate gyrofrequencies
351 Oce = e * Bmag / me;
352 Oci = e * Bmag / mi;
353
354 % Calculate Cm which is a constant that shows up frequencly in the potential source term
355 Cm = (1 / Oci + 1/Oce)^-1;
356
357 % Make spatial mesh

```

```

358 [XX, YY, kmax] = makeSpatialMesh2D(L, n);
359
360 % Get a vector of just x
361 xVec = XX(:,1);
362
363 % Get a vector of just y
364 yVec = YY(1,:)';
365
366 % Set the indices of the two regions from which to get the E field energy spectrum
367 domain_inds{1} = 1:n(1)/4;
368 domain_inds{2} = ( n(1)/4+1 ) : n(1)/2;
369
370 % Set fileIDs and names of energy spectrum data
371 fileIDs(1) = fopen('Energy_left.txt','w');
372 fileIDs(2) = fopen('Energy_right.txt','w');
373
374 % Make a Fourier mesh
375 [k, ksqu, ksqu_4inv] = makeFourierMesh2D(L, n, FourierMeshType);
376
377
378 % Set fileIDs for time vector
379 timeFileID = fopen('tVec.txt','w');
380
381 % Note that the definition of prim_var is
382 % 1: phi
383 % 2: ne
384 % 3: Ti
385 % 4: Te
386 % 5: Pi
387 % 6: Pe
388
389 % Make the names for each primitive variable
390 varNames = {'phi' , 'ne' , 'Ti' , 'Te' , 'Pi' , 'Pe'};
391
392 % Check if we are restarting or starting a new simulations
393 if restartFrame == 0 % This means we are starting a new simulation
394
395     % Set the initial conditions for the background
396
397     % Preallocate the unperturbed variables as all zeros
398     prim_var_unpert = zeros([n 6]);
399
400     % Set the background electric potential
401     prim_var_unpert(:,:,1) = tanhfromSech2IC(XX, [xgPhi_left, xgPhi_right], ...
402         LgPhi, phi0);
403
404     % Set the background density
405     % They are based on a set of 2 hyperbolic tangent functions
406     % See the tanh function for more information
407     prim_var_unpert(:,:,2) = tanh4IC(XX, levels, ...
408         [xgN_left, xgN_right], [LgN_left, LgN_right], L(1), n0);

```

```

409
410 % Set the background ion temperature
411 prim_var_unpert(:,:,3) = Ti0;
412
413 % Set the background electron temperature
414 prim_var_unpert(:,:,4) = Te0;
415
416
417 % Preallocate all perturbations as all zeros
418 prim_var_pert = zeros([size(XX) 6]);
419
420 % Iterate through all of the variables to be perturbed
421 for j = 1:length(pertVar)
422     % Initialize some perturbation
423     prim_var_pert(:,:,pertVar(j)) = pert(XX, YY, L, pertParam, ...
424         pertType, gaussParam, pertDir, var_0(pertVar(j)) );
425 end
426
427 % Get total variable by adding background and perturbation
428 prim_var(:,:,1:4) = prim_var_unpert(:,:,1:4) + prim_var_pert(:,:,1:4);
429
430 % Calculate pressures
431 % Unperturbed
432 prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
433 prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
434
435 % Total
436 prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
437 prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
438
439 % Perturbed
440 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
441 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
442
443 % Save data
444 save3DMat(prim_var_unpert(:,:,1:4), varNames, '_unpert');
445 save3DMat(prim_var_pert(:,:,1:4), varNames, '_0');
446
447 % Initialize a time vector
448 t = 0;
449 iter_start = 1;
450
451 save('t_0.txt','t','-ascii');
452
453 % Save mesh
454 save('X.txt','XX','-ascii');
455 save('Y.txt','YY','-ascii');
456
457 % Save u and B
458 save('u.txt','u','-ascii');
459 save('B.txt','B','-ascii');

```

```

460
461 % Save ion mass
462 save('mi.txt','mi','-ascii');
463
464 % Save Fourier mesh type
465 save('FourierMeshType.txt','FourierMeshType','-ascii');
466
467 % If using constant collision frequencies, save them here
468 if constCollSwitch == 1
469     collParam = [nuin , nuie , nuui , nuen , nuee , nuei];
470     save('collisions.txt','collParam','-ascii');
471 end
472
473 % Save other parameters
474 savedParam = [mi mn FourierMeshType nn ri rn];
475 save('param.txt','savedParam','-ascii');
476
477 % Save diffusion constants
478 save('D.txt','D','-ascii');
479
480 % Make and write to a log file
481 fid = fopen('log.txt','w');
482 fprintf(fid, 'L = [%.2e %.2e]\n', L(1), L(2));
483 fprintf(fid, 'n = [%d %d]\n', n(1), n(2));
484 fprintf(fid, 'B = [%.2e %.2e %.2e]\n', B(1), B(2), B(3));
485 fprintf(fid, 'u = [%.2e %.2e %.2e]\n', u(1), u(2), u(3));
486 fprintf(fid, 'FourierMeshType = %d\n', FourierMeshType);
487 fprintf(fid, 'Dn = %.2e\n', D);
488 fprintf(fid, 'mi = %.2e\n', mi);
489 if constColls(1) == 1
490     fprintf(fid, 'nuin = %.2e\n', nuin);
491     fprintf(fid, 'nuie = %.2e\n', nuie);
492     fprintf(fid, 'nuen = %.2e\n', nuen);
493     fprintf(fid, 'nuei = %.2e\n', nuei);
494     fprintf(fid, 'nuui = %.2e\n', nuui);
495     fprintf(fid, 'nuee = %.2e\n', nuee);
496 else
497     fprintf(fid, 'mn = %.2e\n',mn);
498     fprintf(fid, 'nn = %.2e\n',nn);
499     fprintf(fid, 'ri = %.2e\n',ri);
500     fprintf(fid, 'rn = %.2e\n',rn);
501 end
502
503 if aliasType == 0
504     fprintf(fid, '\nUsing no dealiasing algorithm\n\n');
505 elseif aliasType == 1
506     fprintf(fid, '\nUsing zero-padding dealiasing\n\n');
507 end
508 fclose(fid);
509
510 else % If restarting from a dataset

```

```

511 % Load variables
512 for j = 1:4
513     prim_var_pert(:,:,j) = load([varNames{j} '_' num2str(counter) '.txt']);
514     prim_var_unpert(:,:,j) = load([ varNames{j} '_unpert.txt']);
515 end
516
517 % Get total variables
518 prim_var = prim_var_pert + prim_var_unpert;
519
520 % Calculate pressures
521 % Unperturbed
522 prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
523 prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
524
525 % Total
526 prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
527 prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
528
529 % Perturbed
530 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
531 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
532
533 % Loading time data
534 t = load(['t_' num2str(restartFrame) '.txt']);
535
536 timeFileID = fopen('tVec.txt','a');
537
538 % Open electric field energy data
539 fileIDs(1) = fopen('Eenergy_left.txt','a');
540 fileIDs(2) = fopen('Eenergy_right.txt','a');
541
542 % Set the start of the iterations
543 iter_start = 1;
544 end
545
546 % Run simulation
547 runSimulation_inertia_RK4(prim_var_unpert, prim_var_pert, k, ksqu, ksqu_4inv, kmax, t,
    CFL, dt_min, ...
548 dt_max, tend, u, B, mi, ri, mn, nn, rn, D, iter_start, iter_max, err_max, ...
549 phi_iter_max, saveFrequency, counter, varNames, inHeatingSwitch, ieHeatingSwitch, ...
550 constColls, condSwitch, aliasType, xVec, yVec, domain_inds, fileIDs, timeFileID,
    pertType);
551
552 % Close file IDs
553 fclose('all');
554
555 % Get computational time
556 elapsed_time = toc;
557
558 % Finish up some logging
559 fid = fopen('log.txt','a');

```

```

560 fprintf('Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
561 fprintf(fid,'Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
562 fprintf('\n\n\n\n\n\n\n\n\n');
563 fprintf('Done!\n\n');

```

C.3 GDI_KHI_tanh4N_tanhV

This section discusses a GDI and KHI combination simulation that uses a set of 4 hyperbolic tangent functions to model the background density and 4 hyperbolic tangent functions to model the background velocity.

C.3.1 Initial Conditions

The density initial conditions are the same as those of Section C.2 and are completely defined by Eq. C.5. The velocity is modeled using 4 hyperbolic tangent functions of the form

$$V_{y_{bg}}(x) = \frac{V_0}{2} \left[\tanh\left(\frac{x - x_{gv}^L}{L_{gv}^L}\right) - \tanh\left(\frac{x - x_{gv}^R}{L_{gv}^R}\right) - \tanh\left(\frac{x - L_x + x_{gv}^R}{L_{gv}^R}\right) + \tanh\left(\frac{x - L_x + x_{gv}^L}{L_{gv}^L}\right) \right], \quad (\text{C.13})$$

where $x_{gv}^{L,R}$ are the gradient locations in meters, $L_{gv}^{L,R}$ are the length scaling factors in meters, L_x is the domain length in meters, and V_0 is a reference velocity in m/s. The electric potential initial condition is found using Eq. C.11, resulting in

$$\phi_{bg} = \frac{BV_0}{2} \left[L_{gv}^L \ln\left(\cosh\left[\frac{x - x_{gv}^L}{L_{gv}^L}\right]\right) - L_{gv}^R \ln\left(\cosh\left[\frac{x - x_{gv}^R}{L_{gv}^R}\right]\right) - L_{gv}^R \ln\left(\cosh\left[\frac{x - L_x + x_{gv}^R}{L_{gv}^R}\right]\right) + L_{gv}^L \ln\left(\cosh\left[\frac{x - L_x + x_{gv}^L}{L_{gv}^L}\right]\right) \right], \quad (\text{C.14})$$

where B is the magnetic field strength in T.

The conditions on the density initial condition are the same as those in Section C.2. Additionally, for both the density and velocity, the condition $x_g^L < x_g^R$ must be satisfied. The parameter L_{gN} determines the width of the density gradients. The parameter L_{gv} determines the width of both the velocity gradient and the potential gradient. Figure C.3(a) shows an example plot of the density initial condition using Eq. C.5, with $l_1 = 1.6$, $l_2 = 1$, $l_3 = 2.5$, $x_{gN}^L = 2.2 \times 10^5$ m, $x_{gN}^R = 5.25 \times 10^5$ m, $L_{gN}^L = 5 \times 10^4$ m, $L_{gN}^R = 7.5 \times 10^4$ m, and

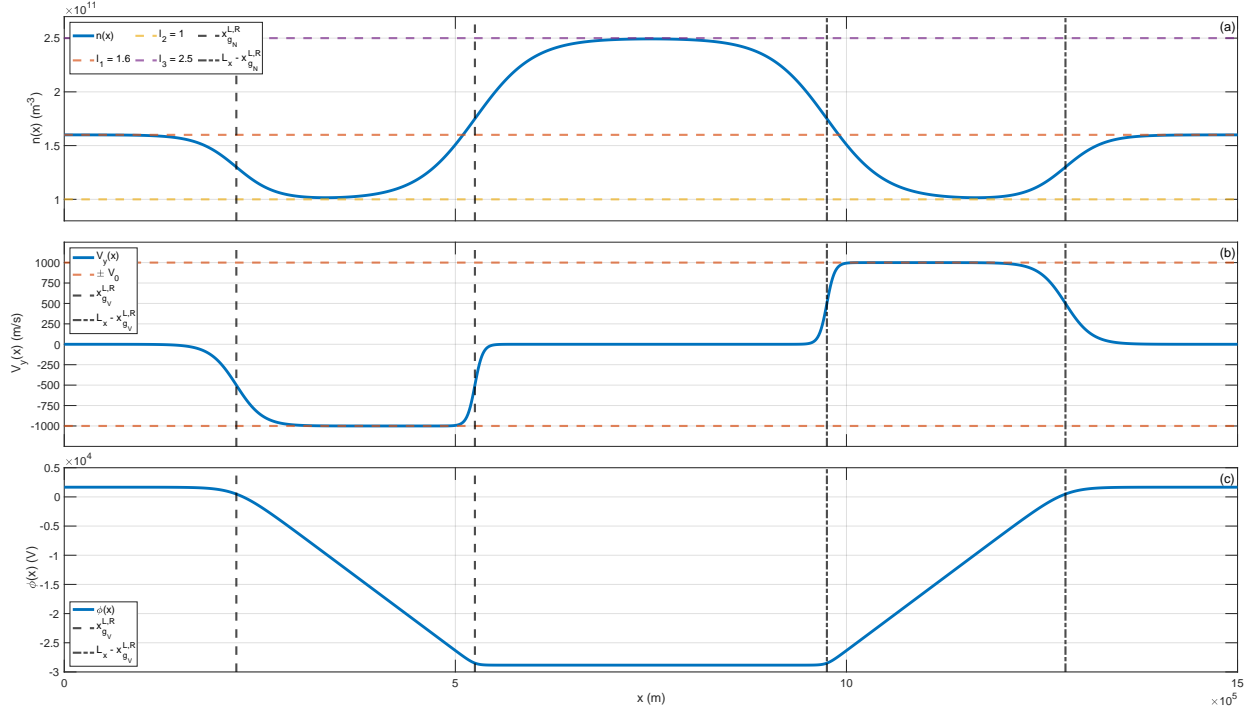


Figure C.3: An example of the background density, velocity, and electric potential profiles as functions of x . The domain length, L_x is 1.5×10^6 m. Panel (a) shows the background density profile (blue curve) using Eq. C.5, with $l_1 = 1.6$, $l_2 = 1$, $l_3 = 2.5$, $x_{g_N}^L = 2.2 \times 10^5$ m, $x_{g_N}^R = 5.25 \times 10^5$ m, $L_{g_N}^L = 5 \times 10^4$ m, $L_{g_N}^R = 7.5 \times 10^4$ m, and $n_0 = 10^{11}$ m $^{-3}$. The purple, red, and yellow dashed lines indicate the user-defined levels of the set of hyperbolic tangent functions. The black lines indicate the locations of the the density gradients. Panel (b) shows the background velocity profile (blue curve) using Eq. C.13, with $x_{g_V}^L = 2.2 \times 10^5$ m, $x_{g_V}^R = 5.25 \times 10^5$ m, $L_{g_V}^L = 3.4 \times 10^4$ m, $L_{g_V}^R = 1 \times 10^4$ m, and $V_0 = -1000$ m/s. The red dashed lines indicate the user-defined maximum velocity, V_0 . The black dashed lines indicate the locations of the velocity profiles. Panel (c) shows the background electric potential profile (blue curve) using Eq. C.14, with $B = 5 \times 10^{-5}$ T. The black dashed lines indicate the locations of the potential gradients.

$n_0 = 10^{11}$ m $^{-3}$. Figure C.3(b) shows an example plot of the velocity initial condition using Eq. C.13, with $x_{g_V}^L = 2.2 \times 10^5$ m, $x_{g_V}^R = 5.25 \times 10^5$ m, $L_{g_V}^L = 3.4 \times 10^4$ m, $L_{g_V}^R = 1 \times 10^4$ m, and $V_0 = -1000$ m/s. Figure C.2(c) shows an example plot of the electric potential initial condition using Eq. C.14, with $B = 5 \times 10^{-5}$ T. Each plot is annotated with the dashed lines to show the effects of some parameters.

C.3.2 Input File

Listing C.3: Input file for a GDI and KHI combination simulation that approximates the density with 4 hyperbolic tangent functions and approximates the velocity with 4 hyperbolic tangent functions.

```

1 % This script is for a GDI simulation with 4 tanh functions to initialize the density
2 % and with a tanh function to initialize the velocity
3
4 % Note: you will want to run this script in the directory in which you want your data to
   be saved
5
6 % Close and clear everything;
7 close all; clearvars; clc;
8
9 % Set up the path here. This needs to be changed depending on where you have the code
   saved
10 addpath('location/to/spacesciencescodes/matlabCodes/keskinen/');
11
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 % Everything in this section is for setting up the simulation.
14 % These parameters are what you want to change for running different simulations.
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % Save data every saveFrequency number of time steps.
18 % For more frames, decrease this number.
19 % For fewer frames, increase this number.
20 saveFrequency = 100;
21
22 % This is the maximum allowable time step in s.
23 % This makes sure that the time step does not skip over important physics
24 % This is set with some a priori understanding of the simulation
25 % It might need to be iterated upon and refined to get an idea of what's going on.
26 dt_max = 20;
27
28 % This is the minimum allowable time step in s.
29 % It is possible that due to the electric potential diverging, the velocity goes to
   infinity.
30 % This results in the time step decreasing to 0.
31 % This can result in the simulation taking a long time despite nothing substantial
   happening
32 % This is used as a check to cancel the simulation since it means the potential is
   blowing up.
33 dt_min = 1e-6;
34
35 % This is the end time in s.
36 % Set this based on when you think you will have enough temporal evolution to your liking
37 tend = 7500*2;
38
39 % This is the maximum allowable error for the iterative method used to calculate the
   electric potential
40 % Increasing this will yield better results but at the cost of additional computational
   effort

```

```

41 % It has not been tested yet but setting this too low will result in the required error
    to be below machine precision.
42 % That would result in the potential never being able to be solved (since the error can
    only go as low as machine precision)
43 err_max = 1e-8;
44
45 % Set the Courant-Friedrichs-Lewy condition for numerical stability
46 % This is dependent on the time integration scheme used
47 % Typically, this is left at 0.9 or 1.
48 % Note that there is additional physics that exist in this system of equations
49 % When calculating the time step, the CFL might be met but it might not consider some
    other important physical time scale
50 CFL = 1;
51
52 % Set the domain length.
53 % First element is the domain length in x direction
54 % Second element is the domain length in y direction
55 L = [1.5e6 1e6];
56
57 % Set the number of grid points in each direction
58 % First element is the number of grid points in the x direction
59 % Second element is the number of grid points in the y direction
60 n = [1024 256];
61
62 % Set the artificial diffusion constant for the density source term
63 % This is needed to dissipate numerical error
64 % Note that as this number is increased, everything will generally take longer to run
65 % If this is increased too high, from experience, the simulation might crash.
66 % The solution is to decrease dt_max (however, this results in the code taking very long
    to run).
67 D = 1e3;
68
69 % This is a flag to choose which type of Fourier mesh to use.
70 % 0 means using the exact definition of k
71 % 1 means using the approximation of k and k^2
72 % See the makeFourierMesh2D function for more information on this.
73 % Depending on the initial conditions chosen, mesh type 1 might need to be chosen to get
    far enough in time
74 FourierMeshType = 1;
75
76 % This is a flag to determine the type of de-aliasing algorithm to use
77 % 0 means using no de-aliasing algorithm
78 % 1 means using the zero pad method (more computationally expensive)
79 % See the convolve2D function for more information on this
80 aliasType = 0;
81
82 % Set the magnetic field in T
83 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
84 % This code really only works for B purely in the z direction (third element) at the
    moment

```

```

85 % So, only initialize the third element and leave the first two 0.
86 B = [0 0 5e-5];
87
88 % Set the neutral wind in m/s.
89 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
90 % This code really only works for u purely in the xy plane (first and second elements)
    at the moment
91 % Soe only initialize the first two elements (x and y respectively) and leave the third
    one 0.
92 u = [-500 0 0];
93
94 % The velocity is initialized using 2 sech^2 functions
95 % Set the left location for the sech^2 function in m
96 % Note that this has to be further left in x than xgPhi_right
97 xgPhi_left = 2.2e5;
98
99 % Set the right location for the sech^2 function in m
100 % Note that this has to be further right in x than xgPhi_left
101 % This case is defined based on the domain length but it does not have to be.
102 xgPhi_right = 5.25e5;
103
104 % Set the denominator term for the left part of potential profile in m
105 LgPhi_left = 3.4e4;
106
107 % Set the denominator term for the left part of potential profile in m
108 LgPhi_right = 1e4;
109
110 % Set the level of the background velocity in m/s
111 V0 = -1000;
112
113 % Calculate the equivalent potential such that ExB = V0
114 % Note that this assumes a purely z direction B field
115 phi0 = V0 * B(3);
116
117 % The density is initialized using a set of 4 hyperbolic tangent functions
118 % If we consider the normalized density, this variable determines the left level
119 % This can be any arbitrary number that is greater than 0
120 levels(1) = 1.6;
121
122 % If we consider the normalized density, this variable determines the middle level
123 % This is typically 1, but it can be any arbitrary number that is greater than 0
124 levels(2) = 1;
125
126 % If we consider the normalized density, this variable determines the right level
127 % This can be any arbitrary number that is greater than 0
128 levels(3) = 2.5;
129
130 % Set the location for the left density gradient in meters
131 % Note that this location has to be further left in x than xgN_right
132 xgN_left = 2.2e5;

```

```

133
134 % Set the location for the right density gradient in meters
135 % Note that this location has to be further right in x than xgN_leftt
136 xgN_right = 5.25e5;
137
138 % Set the denominator term for the left density gradient in meters
139 % Note if this is too small, this can result in fast accumulation of numerical error
140 % If this is too big, the density might not be periodic enough within the domain and the
    code will crash
141 LgN_left = 5e4;
142
143 % Set the denominator term for the right density gradient in meters
144 % Note if this is too small, this can result in fast accumulation of numerical error
145 % If this is too big, the density might not be periodic enough within the domain and the
    code will crash
146 % Note that this is not required to be the same as LgN_left (it just is in this case).
147 LgN_right = 7.5e4;
148
149 % Set the level of the background density.
150 % The normalized set of hyperbolic tangents are multiplied by n0 to give the density
151 n0 = 1e11;
152
153 % Depending on how xg_left, xg_right, L_left, and L_right are set, you might have a
    function that
154 % does not quite look like a hyperbolic tangent.
155 % Play around with the different parameters to determine what works best for your
    simulation
156
157 % Set a constant ion temperature in K
158 % The ion temperature is assumed to be constant in the entire domain
159 Ti0 = 1000;
160
161 % Set a constant electron temperature in K
162 % The electron temperature is assumed to be constant in the entire domain
163 Te0 = 1000;
164
165 % Make a vector of the background constant multiplicative values (all the 0 variables)
    for each primitive variable
166 % This is used when calculating the perturbation
167 var_0 = [phi0, n0, Ti0, Te0];
168
169 % Set the neutral number density in 1/m^3
170 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
171 % See the calcCollFreqk function for more information.
172 % Note that the model assumes F region ionosphere, therefore if nn is too large, the
    code will crash
173 % The assumption that nu/Omega is small will break.
174 nn = 1e14;
175
176 % Set the radius of the neutral particles in m
177 % This is used to calculate the ion-neutral and electron-neutral collision frequencies

```

```

178 % See the calcCollFreqk function for more information.
179 rn = 152e-12; % This specific value assumed 0 is dominant neutral species
180
181 % Set the radius of the ions in m
182 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
183 % See the calcCollFreqk function for more information.
184 ri = 152e-12; % This specific value assumed 0+ is dominant ion species
185
186 % Set the mass of the neutrals in kg
187 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
188 % See the calcCollFreqk function for more information.
189 mi = 16 * 1.66053906660e-27; % This specific value assumed 0 is dominant neutral species
190
191 % Set the mass of the ions in kg
192 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
193 % See the calcCollFreqk function for more information.
194 mn = 16 * 1.66053906660e-27; % This specific value assumed 0+ is dominant ion species
195
196
197 % This is what is used to setup the initial perturbation for the instability growth
198 % See the pert function for more information
199 % For now, only have pertType be 0, 1, or 4. The other types aren't properly implemented
    yet
200 % Each pertType corresponds to a different type of perturbation to apply
201 % 0 is single mode
202 % 1 is multi mode
203 % 2 is integral of single mode (Note fully tested or implemented, DON'T USE!)
204 % 3 is integral of multimode (Note fully tested or implemented, DON'T USE!)
205 % 4 is random noise
206 % 5 is multimode times multimode (Note fully tested or implemented, DON'T USE!)
207 % 6 is integral of multimode times multimode (Note fully tested or implemented, DON'T
    USE!)
208 pertType = 4;
209
210 % Define pertParam depending on which perturbation is chosen
211 % The definition of what goes in pertParam is different for each pertType
212 switch pertType
213     case {0,2} % Single mode and integral of single mode
214         pertParam(1) = 1e-6; % Amplitude of x direction wave
215         pertParam(2) = 1e-6; % Amplitude of y direction wave
216         pertParam(3) = 4; % Mode to be perturbed in x
217         pertParam(4) = 6; % Mode to be perturbed in y
218
219     case {1,3,5,6} % Multi mode, integral of multimode, multimode*multimode, integral of
        multimode*multimode
220         pertParam(1) = 1.e-6; % Minimum random amplitude
221         pertParam(2) = 1.e-6; % Maximum random amplitude
222         pertParam(3) = 0; % Minimum random phase shift
223         pertParam(4) = 2*pi; % Maximum random phase shift
224         pertParam(5) = 64; % Number of modes to excite in x
225         pertParam(6) = 128; % Number of modes to excite in y

```

```

226
227     case 4 % Random noise
228         pertParam(1) = -1.e-6; % Minimum random noise
229         pertParam(2) = 1.e-6; % Maximum random noise
230         pertParam(3) = 1;    % If 1, only apply random noise to left half of domain.
231     end
232
233 % This determines which variable to add the perturbation to (phi, n, Ti, or Te) (1, 2,
234 % 3, or 4)
235 % For this type of simulation (GDI with 2 tanh for density and no velocity), always
236 % perturb density
237 % Note that pertVar can be a vector allowing multiple variables to be perturbed
238 pertVar = [2];
239
240 % Depending on the perturbation, we may want to multiply it by a Gaussian function so
241 % that it acts only at a particular location
242 % The following are the parameters for the Gaussian function
243
244 % This sets the width of the left Gaussian (if applicable) in 1/m
245 % This is presently defined based on LgN_left but it does not have to be
246 gaussParam(1) = 2 / LgN_left;
247
248 % This sets the width of the right Gaussian (if applicable) in 1/m
249 % This is presently defined based on LgN_right but it does not have to be
250 gaussParam(2) = 2 / LgN_right;
251
252 % This is a switch for the left Gaussian.
253 % 0 is off
254 % 1 is on
255 gaussParam(3) = 0;
256
257 % This is a switch for the right Gaussian.
258 % 0 is off
259 % 1 is on
260 gaussParam(4) = 1;
261
262 % Note that if both the left and the right are switched off, then the code assumes you
263 % want
264 % the perturbation to exist in the entire domain.
265 % Therefore, it multiplies the perturbation by 1 instead of a set of Gaussians
266
267 % This sets the location of the left Gaussian in m
268 % It is presently based on xgN_left but it does not have to be
269 gaussParam(5) = xgN_left;
270
271 % This sets the location of the right Gaussian in m
272 % It is presently based on xgN_right but it does not have to be
273 gaussParam(6) = xgN_right;
274
275 % This sets the direction we want to apply the perturbation
276 % i.e. is the perturbation a function of x or y?

```

```

273 % First index is x, second is y.
274 % Both can be set to 0 or 1 (off or on).
275 pertDir = [0 1];
276
277 % The code calculates the collision frequencies self-consistently
278 % There may be situations in which you want to specify a constant set of frequencies
279 % constCollSwitch determines if you want to use the self-consistent or pre-defined
      frequencies
280 % 0 is off which means that you are using the self-consistent collision frequencies
281 % 1 is on which means that you are using the pre-defined constant collision frequencies
282 constCollSwitch = 0;
283
284 % If constCollSwitch is 0, then neglect the next few lines
285 % If constCollSwitch is 1, then set the collision frequencies
286 nuin = 0; % ion-neutral
287 nuii = 0; % ion-ion
288 nuie = 0; % ion-electron
289 nuen = 0; % electron-neutral
290 nuei = 0; % electron-ion
291 nuee = 0; % electron-electron
292
293 % The calcCollFreqk function takes all of the above set of a parameters in as a single
      vector
294 constColls = [constCollSwitch , nuin , nuii , nuie , nuen , nuei , nuee];
295
296 % This is a flag to determine if the ion-neutral frictional heating source terms will be
      included
297 % 0 means off
298 % 1 means on
299 inHeatingSwitch = 0;
300
301 % Set the neutral temperature in K
302 % This is only needed if inHeatingSwitch == 1
303 % Otherwise, this is not used in the code
304 % Note that for the code to work properly, this needs to be a 2D matrix
305 % This can allow for non-uniform neutral temperatures
306 Tn = 1000 * ones(n);
307
308 % This is a flag to determine if the ion-electron frictional heating source terms will
      be included
309 % 0 means off
310 % 1 means on
311 ieHeatingSwitch = 0;
312
313 % This is a flag to determine if the thermal conduction source terms will be included
314 % 0 means off
315 % 1 means on
316 condSwitch = 0;
317
318 % Choose frame to restart.

```

```

319 % 0 means that the simulation will start from the beginning and make the initial
      conditions, then run.
320 % For any other integer (> 0), it will go to that frame and rerun the simulation from
      that start point
321 restartFrame = 0;
322
323 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
324 % Everything past here does not need to be changed to run this simulation.
325 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
326
327 % Set the counter to the frame to restart so that new frames are saved accordingly and
      not overwritng old ones
328 counter = restartFrame;
329
330 % Setup max time iterations so that the code has some finite end iteration.
331 iter_max = 100000;
332
333
334 % Setup max iterations for potential equation.
335 % If the iterations go over this, the code will stop because the potential is not
      converging.
336 phi_iter_max = 500;
337
338 % Set fundamental physical constants
339 me = 9.1093837015e-31; % kg
340 e = 1.602176634e-19; % C
341 kb = 1.380649e-23; % J/K
342 eps0 = 8.8541878128e-12; % F/m
343
344 % Get computational time
345 tic;
346
347 % Calculate the magnitude of B and its square
348 Bmag = norm(B);
349 B2 = Bmag^2;
350
351 % Calculate gyrofrequencies
352 Oce = e * Bmag / me;
353 Oci = e * Bmag / mi;
354
355 % Calculate Cm which is a constant that shows up frequencly in the potential source term
356 Cm = (1 / Oci + 1/Oce)^-1;
357
358 % Make spatial mesh
359 [XX, YY, kmax] = makeSpatialMesh2D(L, n);
360
361 % Get a vector of just x
362 xVec = XX(:,1);
363
364 % Get a vector of just y
365 yVec = YY(1,:)';

```



```

366
367 % Set the indices of the two regions from which to get the E field energy spectrum
368 domain_inds{1} = 1:n(1)/4;
369 domain_inds{2} = ( n(1)/4+1 ) : n(1)/2;
370
371 % Set fileIDs and names of energy spectrum data
372 fileIDs(1) = fopen('Energy_left.txt','w');
373 fileIDs(2) = fopen('Energy_right.txt','w');
374
375 % Make a Fourier mesh
376 [k, ksqu, ksqu_4inv] = makeFourierMesh2D(L, n, FourierMeshType);
377
378
379 % Set fileIDs for time vector
380 timeFileID = fopen('tVec.txt','w');
381
382 % Note that the definition of prim_var is
383 % 1: phi
384 % 2: ne
385 % 3: Ti
386 % 4: Te
387 % 5: Pi
388 % 6: Pe
389
390 % Make the names for each primitive variable
391 varNames = {'phi' , 'ne' , 'Ti' , 'Te' , 'Pi' , 'Pe'};
392
393 % Check if we are restarting or starting a new simulations
394 if restartFrame == 0 % This means we are starting a new simulation
395
396     % Set the initial conditions for the background
397
398     % Preallocate the unperturbed variables as all zeros
399     prim_var_unpert = zeros([n 6]);
400
401     % Set the background electric potential
402     prim_var_unpert(:,:,1) = logcoshIC(XX, [LgPhi_left, LgPhi_right], ...
403         [xgPhi_left, xgPhi_right], L(1), phi0);
404
405
406     % Set the background density
407     % They are based on a set of 2 hyperbolic tangent functions
408     % See the tanh function for more information
409     prim_var_unpert(:,:,2) = tanh4IC(XX, levels, ...
410         [xgN_left, xgN_right], [LgN_left, LgN_right], L(1), n0);
411
412     % Set the background ion temperature
413     prim_var_unpert(:,:,3) = Ti0;
414
415     % Set the background electron temperature
416     prim_var_unpert(:,:,4) = Te0;

```

```

417
418
419 % Preallocate all perturbations as all zeros
420 prim_var_pert = zeros([size(XX) 6]);
421
422 % Iterate through all of the variables to be perturbed
423 for j = 1:length(pertVar)
424     % Initialize some perturbation
425     prim_var_pert(:,:,pertVar(j)) = pert(XX, YY, L, pertParam, ...
426         pertType, gaussParam, pertDir, var_0(pertVar(j)) );
427 end
428
429 % Get total variable by adding background and perturbation
430 prim_var(:,:,1:4) = prim_var_unpert(:,:,1:4) + prim_var_pert(:,:,1:4);
431
432 % Calculate pressures
433 % Unperturbed
434 prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
435 prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
436
437 % Total
438 prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
439 prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
440
441 % Perturbed
442 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
443 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
444
445 % Save data
446 save3DMat(prim_var_unpert(:,:,1:4), varNames, '_unpert');
447 save3DMat(prim_var_pert(:,:,1:4), varNames, '_0');
448
449 % Initialize a time vector
450 t = 0;
451 iter_start = 1;
452
453 save('t_0.txt','t','-ascii');
454
455 % Save mesh
456 save('X.txt','XX','-ascii');
457 save('Y.txt','YY','-ascii');
458
459 % Save u and B
460 save('u.txt','u','-ascii');
461 save('B.txt','B','-ascii');
462
463 % Save ion mass
464 save('mi.txt','mi','-ascii');
465
466 % Save Fourier mesh type
467 save('FourierMeshType.txt','FourierMeshType','-ascii');

```

```

468
469 % If using constant collision frequencies, save them here
470 if constCollSwitch == 1
471     collParam = [nuin , nuie , nuui , nuen , nuee , nuei];
472     save('collisions.txt','collParam','-ascii');
473 end
474
475 % Save other parameters
476 savedParam = [mi mn FourierMeshType nn ri rn];
477 save('param.txt','savedParam','-ascii');
478
479 % Save diffusion constants
480 save('D.txt','D','-ascii');
481
482 % Make and write to a log file
483 fid = fopen('log.txt','w');
484 fprintf(fid, 'L = [%.2e %.2e]\n', L(1), L(2));
485 fprintf(fid, 'n = [%d %d]\n', n(1), n(2));
486 fprintf(fid, 'B = [%.2e %.2e %.2e]\n', B(1), B(2), B(3));
487 fprintf(fid, 'u = [%.2e %.2e %.2e]\n', u(1), u(2), u(3));
488 fprintf(fid, 'FourierMeshType = %d\n', FourierMeshType);
489 fprintf(fid, 'Dn = %.2e\n', D);
490 fprintf(fid, 'mi = %.2e\n', mi);
491 if constColls(1) == 1
492     fprintf(fid, 'nuin = %.2e\n', nuin);
493     fprintf(fid, 'nuie = %.2e\n', nuie);
494     fprintf(fid, 'nuen = %.2e\n', nuen);
495     fprintf(fid, 'nuei = %.2e\n', nuei);
496     fprintf(fid, 'nuui = %.2e\n', nuui);
497     fprintf(fid, 'nuee = %.2e\n', nuee);
498 else
499     fprintf(fid, 'mn = %.2e\n',mn);
500     fprintf(fid, 'nn = %.2e\n',nn);
501     fprintf(fid, 'ri = %.2e\n',ri);
502     fprintf(fid, 'rn = %.2e\n',rn);
503 end
504
505 if aliasType == 0
506     fprintf(fid, '\nUsing no dealiasing algorithm\n\n');
507 elseif aliasType == 1
508     fprintf(fid, '\nUsing zero-padding dealiasing\n\n');
509 end
510 fclose(fid);
511
512 else % If restarting from a dataset
513     % Load variables
514     for j = 1:4
515         prim_var_pert(:, :, j) = load([varNames{j} '_' num2str(counter) '.txt']);
516         prim_var_unpert(:, :, j) = load([ varNames{j} '_unpert.txt']);
517     end
518

```

```

519 % Get total variables
520 prim_var = prim_var_pert + prim_var_unpert;
521
522 % Calculate pressures
523 % Unperturbed
524 prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
525 prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
526
527 % Total
528 prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
529 prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
530
531 % Perturbed
532 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
533 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
534
535 % Loading time data
536 t = load(['t_' num2str(restartFrame) '.txt']);
537
538 timeFileID = fopen('tVec.txt','a');
539
540 % Open electric field energy data
541 fileIDs(1) = fopen('Eenergy_left.txt','a');
542 fileIDs(2) = fopen('Eenergy_right.txt','a');
543
544 % Set the start of the iterations
545 iter_start = 1;
546 end
547
548 % Run simulation
549 runSimulation_inertia_RK4(prim_var_unpert, prim_var_pert, k, ksqu, ksqu_4inv, kmax, t,
    CFL, dt_min, ...
550 dt_max, tend, u, B, mi, ri, mn, nn, rn, D, iter_start, iter_max, err_max, ...
551 phi_iter_max, saveFrequency, counter, varNames, inHeatingSwitch, ieHeatingSwitch, ...
552 constColls, condSwitch, aliasType, xVec, yVec, domain_inds, fileIDs, timeFileID,
    pertType);
553
554 % Close file IDs
555 fclose('all');
556
557 % Get computational time
558 elapsed_time = toc;
559
560 % Finish up some logging
561 fid = fopen('log.txt','a');
562 fprintf('Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
563 fprintf(fid,'Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
564 fprintf('\n\n\n\n\n\n\n\n');
565 fprintf('Done!\n\n');

```

C.4 GDI_KHI_tanh2N_tanhV

This input file is a combination of the input files from Section C.1 and C.3. The density is initialized using Eq. C.5. The velocity is initialized using Eq. C.13. The electric potential is initialized using Eq. C.14. See Figures C.1 and C.3(b-c) for example plots of these initial conditions.

C.4.1 Input File

Listing C.4: Input file for a GDI and KHI combination simulation that approximates the density with 2 hyperbolic tangent functions and approximates the velocity with 4 hyperbolic tangent functions.

```

1 % This script is for a GDI simulation with 2 tanh functions to initialize the density
2 % and with a tanh function to initialize the velocity
3
4 % Note: you will want to run this script in the directory in which you want your data to
   % be saved
5
6 % Close and clear everything;
7 close all; clearvars; clc;
8
9 % Set up the path here. This needs to be changed depending on where you have the code
   % saved
10 addpath('location/to/spacesciencescodes/matlabCodes/keskinen/');
11
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 % Everything in this section is for setting up the simulation.
14 % These parameters are what you want to change for running different simulations.
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % Save data every saveFrequency number of time steps.
18 % For more frames, decrease this number.
19 % For fewer frames, increase this number.
20 saveFrequency = 20;
21
22 % This is the maximum allowable time step in s.
23 % This makes sure that the time step does not skip over important physics
24 % This is set with some a priori understanding of the simulation
25 % It might need to be iterated upon and refined to get an idea of what's going on.
26 dt_max = 20;
27
28 % This is the minimum allowable time step in s.
29 % It is possible that due to the electric potential diverging, the velocity goes to
   % infinity.
30 % This results in the time step decreasing to 0.
```

```

31 % This can result in the simulation taking a long time despite nothing substantial
    happening
32 % This is used as a check to cancel the simulation since it means the potential is
    blowing up.
33 dt_min = 1e-8;
34
35 % This is the end time in s.
36 % Set this based on when you think you will have enough temporal evolution to your liking
37 tend = 500;
38
39 % This is the maximum allowable error for the iterative method used to calculate the
    electric potential
40 % Increasing this will yield better results but at the cost of additional computational
    effort
41 % It has not been tested yet but setting this too low will result in the required error
    to be below machine precision.
42 % That would result in the potential never being able to be solved (since the error can
    only go as low as machine precision)
43 err_max = 1e-8;
44
45 % Set the Courant-Friedrichs-Lewy condition for numerical stability
46 % This is dependent on the time integration scheme used
47 % Typically, this is left at 0.9 or 1.
48 % Note that there is additional physics that exist in this system of equations
49 % When calculating the time step, the CFL might be met but it might not consider some
    other important physical time scale
50 CFL = 1;
51
52 % Set the domain length.
53 % First element is the domain length in x direction
54 % Second element is the domain length in y direction
55 Lx = 5000;
56 L = [Lx*2 2*pi*100/.44];
57
58 % Set the number of grid points in each direction
59 % First element is the number of grid points in the x direction
60 % Second element is the number of grid points in the y direction
61 n = [1024 64];
62
63 % Set the artificial diffusion constant for the density source term
64 % This is needed to dissipate numerical error
65 % Note that as this number is increased, everything will generally take longer to run
66 % If this is increased too high, from experience, the simulation might crash.
67 % The solution is to decrease dt_max (however, this results in the code taking very long
    to run).
68 D = 1e1;
69
70 % This is a flag to choose which type of Fourier mesh to use.
71 % 0 means using the exact definition of k
72 % 1 means using the approximation of k and k^2
73 % See the makeFourierMesh2D function for more information on this.

```

```

74 % Depending on the initial conditions chosen, mesh type 1 might need to be chosen to get
    far enough in time
75 FourierMeshType = 1;
76
77 % This is a flag to determine the type of de-aliasing algorithm to use
78 % 0 means using no de-aliasing algorithm
79 % 1 means using the zero pad method (more computationally expensive)
80 % See the convolve2D function for more information on this
81 aliasType = 0;
82
83 % Set the magnetic field in T
84 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
85 % This code really only works for B purely in the z direction (third element) at the
    moment
86 % So, only initialize the third element and leave the first two 0.
87 B = [0 0 5e-5];
88
89 % Set the neutral wind in m/s.
90 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
91 % This code really only works for u purely in the xy plane (first and second elements)
    at the moment
92 % Soe only initialize the first two elements (x and y respectively) and leave the third
    one 0.
93 u = [0 0 0];
94
95 % The velocity is initialized using 2 sech^2 functions
96 % Set the left location for the sech^2 function in m
97 % Note that this has to be further left in x than xgPhi_right
98 xgPhi_left = .3*Lx;
99
100 % Set the right location for the sech^2 function in m
101 % Note that this has to be further right in x than xgPhi_left
102 % This case is defined based on the domain length but it does not have to be.
103 xgPhi_right = .7*Lx;
104
105 % Set the denominator term for the left part of potential profile in m
106 LgPhi_left = 100;
107
108 % Set the denominator term for the left part of potential profile in m
109 LgPhi_right = 100;
110
111 % Set the level of the background velocity in m/s
112 V0 = 300;
113
114 % Calculate the equivalent potential such that ExB = V0
115 % Note that this assumes a purely z direction B field
116 phi0 = V0 * B(3);
117
118 % The density is initialized using a set of 2 hyperbolic tangent functions

```

```

119 % If we consider the normalized density, this variable determines the value to set in
    the middle region
120 % Typically, this is kept as one but can be any arbitrary number that is greater than 0
121 middle = 1;
122
123 % If we consider the normalized density, this variable detemrines the value to set in
    the outer regions
124 % This can be any arbitrary number that is greater than 0
125 outer = 3;
126
127 % Set the location for the left density gradient in meters
128 % Note that this location has to be further left in x than xgN_right
129 xgN_left = .3*Lx;
130
131 % Set the location for the right density gradient in meters
132 % Note that this location has to be further right in x than xgN_leftt
133 xgN_right = 2*Lx - .3*Lx;
134
135 % Set the denominator term for the left density gradient in meters
136 % Note if this is too small, this can result in fast accumulation of numerical error
137 % If this is too big, the density might not be periodic enough within the domain and the
    code will crash
138 LgN_left = 100;
139
140 % Set the denominator term for the right density gradient in meters
141 % Note if this is too small, this can result in fast accumulation of numerical error
142 % If this is too big, the density might not be periodic enough within the domain and the
    code will crash
143 % Note that this is not required to be the same as LgN_left (it just is in this case).
144 LgN_right = 100;
145
146 % Set the level of the background density.
147 % The normalized set of hyperbolic tangents are multiplied by n0 to give the density
148 n0 = 1e11;
149
150 % Depending on how xg_left, xg_right, L_left, and L_right are set, you might have a
    function that
151 % does not quite look like a hyperbolic tangent.
152 % Play around with the different parameters to determine what works best for your
    simulation
153
154 % Set a constant ion temperature in K
155 % The ion temperature is assumed to be constant in the entire domain
156 Ti0 = 1000;
157
158 % Set a constant electron temperature in K
159 % The electron temperature is assumed to be constant in the entire domain
160 Te0 = 1000;
161
162 % Make a vector of the background constant multiplicative values (all the 0 variables)
    for each primitive variable

```



```

163 % This is used when calculating the perturbation
164 var_0 = [phi0, n0, Ti0, Te0];
165
166 % Set the neutral number density in 1/m^3
167 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
168 % See the calcCollFreqk function for more information.
169 % Note that the model assumes F region ionosphere, therefore if nn is too large, the
      code will crash
170 % The assumption that nu/Omega is small will break.
171 nn = 0;
172
173 % Set the radius of the neutral particles in m
174 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
175 % See the calcCollFreqk function for more information.
176 rn = 152e-12; % This specific value assumed O is dominant neutral species
177
178 % Set the radius of the ions in m
179 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
180 % See the calcCollFreqk function for more information.
181 ri = 152e-12; % This specific value assumed O+ is dominant ion species
182
183 % Set the mass of the neutrals in kg
184 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
185 % See the calcCollFreqk function for more information.
186 mi = 16 * 1.66053906660e-27; % This specific value assumed O is dominant neutral species
187
188 % Set the mass of the ions in kg
189 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
190 % See the calcCollFreqk function for more information.
191 mn = 16 * 1.66053906660e-27; % This specific value assumed O+ is dominant ion species
192
193
194 % This is what is used to setup the initial perturbation for the instability growth
195 % See the pert function for more information
196 % For now, only have pertType be 0, 1, or 4. The other types aren't properly implemented
      yet
197 % Each pertType corresponds to a different type of perturbation to apply
198 % 0 is single mode
199 % 1 is multi mode
200 % 2 is integral of single mode (Note fully tested or implemented, DON'T USE!)
201 % 3 is integral of multimode (Note fully tested or implemented, DON'T USE!)
202 % 4 is random noise
203 % 5 is multimode times multimode (Note fully tested or implemented, DON'T USE!)
204 % 6 is integral of multimode times multimode (Note fully tested or implemented, DON'T
      USE!)
205 pertType = 0;
206
207 % Define pertParam depending on which perturbation is chosen
208 % The definition of what goes in pertParam is different for each pertType
209 switch pertType
210     case {0,2} % Single mode and integral of single mode

```

```

211     pertParam(1) = 1e-6; % Amplitude of x direction wave
212     pertParam(2) = 1e-6; % Amplitude of y direction wave
213     pertParam(3) = 1;   % Mode to be perturbed in x
214     pertParam(4) = 1;   % Mode to be perturbed in y
215
216     case {1,3,5,6} % Multi mode, integral of multimode, multimode*multimode, integral of
                multimode*multimode
217         pertParam(1) = 1.e-6; % Minimum random amplitude
218         pertParam(2) = 1.e-6; % Maximum random amplitude
219         pertParam(3) = 0;   % Minimum random phase shift
220         pertParam(4) = 2*pi; % Maximum random phase shift
221         pertParam(5) = 64;  % Number of modes to excite in x
222         pertParam(6) = 128; % Number of modes to excite in y
223
224     case 4 % Random noise
225         pertParam(1) = -1.e-3; % Minimum random noise
226         pertParam(2) = 1.e-3; % Maximum random noise
227         pertParam(3) = 1;     % If 1, only apply random noise to left half of domain.
228     end
229
230 % This determines which variable to add the perturbation to (phi, n, Ti, or Te) (1, 2,
    3, or 4)
231 % For this type of simulation (GDI with 2 tanh for density and no velocity), always
    perturb density
232 % Note that pertVar can be a vector allowing multiple variables to be perturbed
233 pertVar = [2];
234
235 % Depending on the perturbation, we may want to multiply it by a Gaussian function so
    that it acts only at a particular location
236 % The following are the parameters for the Gaussian function
237
238 % This sets the width of the left Gaussian (if applicable) in 1/m
239 % This is presently defined based on LgN_left but it does not have to be
240 gaussParam(1) = 2 / LgN_left;
241
242 % This sets the width of the right Gaussian (if applicable) in 1/m
243 % This is presently defined based on LgN_right but it does not have to be
244 gaussParam(2) = 2 / LgN_right;
245
246 % This is a switch for the left Gaussian.
247 % 0 is off
248 % 1 is on
249 gaussParam(3) = 0;
250
251 % This is a switch for the right Gaussian.
252 % 0 is off
253 % 1 is on
254 gaussParam(4) = 1;
255
256 % Note that if both the left and the right are switched off, then the code assumes you
    want

```

```

257 % the perturbation to exist in the entire domain.
258 % Therefore, it multiplies the perturbation by 1 instead of a set of Gaussians
259
260 % This sets the location of the left Gaussian in m
261 % It is presently based on xgN_left but it does not have to be
262 gaussParam(5) = xgN_left;
263
264 % This sets the location of the right Gaussian in m
265 % It is presently based on xgN_right but it does not have to be
266 gaussParam(6) = xgN_right;
267
268 % This sets the direction we want to apply the perturbation
269 % i.e. is the perturbation a function of x or y?
270 % First index is x, second is y.
271 % Both can be set to 0 or 1 (off or on).
272 pertDir = [0 1];
273
274 % The code calculates the collision frequencies self-consistently
275 % There may be situations in which you want to specify a constant set of frequencies
276 % constCollSwitch determines if you want to use the self-consistent or pre-defined
      frequencies
277 % 0 is off which means that you are using the self-consistent collision frequencies
278 % 1 is on which means that you are using the pre-defined constant collision frequencies
279 constCollSwitch = 0;
280
281 % If constCollSwitch is 0, then neglect the next few lines
282 % If constCollSwitch is 1, then set the collision frequencies
283 nuin = 0; % ion-neutral
284 nuii = 0; % ion-ion
285 nuie = 0; % ion-electron
286 nuen = 0; % electron-neutral
287 nuei = 0; % electron-ion
288 nuee = 0; % electron-electron
289
290 % The calcCollFreqk function takes all of the above set of a parameters in as a single
      vector
291 constColls = [constCollSwitch , nuin , nuii , nuie , nuen , nuei , nuee];
292
293 % This is a flag to determine if the ion-neutral frictional heating source terms will be
      included
294 % 0 means off
295 % 1 means on
296 inHeatingSwitch = 0;
297
298 % Set the neutral temperature in K
299 % This is only needed if inHeatingSwitch == 1
300 % Otherwise, this is not used in the code
301 % Note that for the code to work properly, this needs to be a 2D matrix
302 % This can allow for non-uniform neutral temperatures
303 Tn = 1000 * ones(n);
304

```

```

305 % This is a flag to determine if the ion-electron frictional heating source terms will
      be included
306 % 0 means off
307 % 1 means on
308 ieHeatingSwitch = 0;
309
310 % This is a flag to determine if the thermal conduction source terms will be included
311 % 0 means off
312 % 1 means on
313 condSwitch = 0;
314
315 % Choose frame to restart.
316 % 0 means that the simulation will start from the beginning and make the initial
      conditions, then run.
317 % For any other integer (> 0), it will go to that frame and rerun the simulation from
      that start point
318 restartFrame = 0;
319
320 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
321 % Everything past here does not need to be changed to run this simulation.
322 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
323
324 % Set the counter to the frame to restart so that new frames are saved accordingly and
      not overwritng old ones
325 counter = restartFrame;
326
327 % Setup max time iterations so that the code has some finite end iteration.
328 iter_max = 100000;
329
330
331 % Setup max iterations for potential equation.
332 % If the iterations go over this, the code will stop because the potential is not
      converging.
333 phi_iter_max = 500;
334
335 % Set fundamental physical constants
336 me = 9.1093837015e-31; % kg
337 e = 1.602176634e-19; % C
338 kb = 1.380649e-23; % J/K
339 eps0 = 8.8541878128e-12; % F/m
340
341 % Get computational time
342 tic;
343
344 % Calculate the magnitude of B and its square
345 Bmag = norm(B);
346 B2 = Bmag^2;
347
348 % Calculate gyrofrequencies
349 Oce = e * Bmag / me;
350 Oci = e * Bmag / mi;

```

```

351
352 % Calculate Cm which is a constant that shows up frequently in the potential source term
353 Cm = (1 / Oci + 1/Oce)^-1;
354
355 % Make spatial mesh
356 [XX, YY, kmax] = makeSpatialMesh2D(L, n);
357
358 % Get a vector of just x
359 xVec = XX(:,1);
360
361 % Get a vector of just y
362 yVec = YY(1,:)';
363
364 % Set the indices of the two regions from which to get the E field energy spectrum
365 domain_inds{1} = 1:n(1)/4;
366 domain_inds{2} = ( n(1)/4+1 ) : n(1)/2;
367
368 % Set fileIDs and names of energy spectrum data
369 fileIDs(1) = fopen('Energy_left.txt','w');
370 fileIDs(2) = fopen('Energy_right.txt','w');
371
372 % Make a Fourier mesh
373 [k, ksqu, ksqu_4inv] = makeFourierMesh2D(L, n, FourierMeshType);
374
375
376 % Set fileIDs for time vector
377 timeFileID = fopen('tVec.txt','w');
378
379 % Note that the definition of prim_var is
380 % 1: phi
381 % 2: ne
382 % 3: Ti
383 % 4: Te
384 % 5: Pi
385 % 6: Pe
386
387 % Make the names for each primitive variable
388 varNames = {'phi' , 'ne' , 'Ti' , 'Te' , 'Pi' , 'Pe'};
389
390 % Check if we are restarting or starting a new simulations
391 if restartFrame == 0 % This means we are starting a new simulation
392
393     % Set the initial conditions for the background
394
395     % Preallocate the unperturbed variables as all zeros
396     prim_var_unpert = zeros([n 6]);
397
398     % Set the background electric potential
399     prim_var_unpert(:, :, 1) = logcoshIC(XX, [LgPhi_left, LgPhi_right], ...
400         [xgPhi_left, xgPhi_right], L(1), phi0);
401

```

```

402
403 % Set the background density
404 % They are based on a set of 2 hyperbolic tangent functions
405 % See the tanh function for more information
406 prim_var_unpert(:,:,2) = tanhIC(XX, outer, middle, [xgN_left, xgN_right], ...
407     [LgN_left, LgN_right], n0);
408
409 % Set the background ion temperature
410 prim_var_unpert(:,:,3) = Ti0;
411
412 % Set the background electron temperature
413 prim_var_unpert(:,:,4) = Te0;
414
415
416 % Preallocate all perturbations as all zeros
417 prim_var_pert = zeros([size(XX) 6]);
418
419 % Iterate through all of the variables to be perturbed
420 for j = 1:length(pertVar)
421     % Initialize some perturbation
422     prim_var_pert(:,:,pertVar(j)) = pert(XX, YY, L, pertParam, ...
423         pertType, gaussParam, pertDir, var_0(pertVar(j)) );
424 end
425
426 % Get total variable by adding background and perturbation
427 prim_var(:,:,1:4) = prim_var_unpert(:,:,1:4) + prim_var_pert(:,:,1:4);
428
429 % Calculate pressures
430 % Unperturbed
431 prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
432 prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
433
434 % Total
435 prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
436 prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
437
438 % Perturbed
439 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
440 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
441
442 % Save data
443 save3DMat(prim_var_unpert(:,:,1:4), varNames, '_unpert');
444 save3DMat(prim_var_pert(:,:,1:4), varNames, '_0');
445
446 % Initialize a time vector
447 t = 0;
448 iter_start = 1;
449
450 save('t_0.txt', 't', '-ascii');
451
452 % Save mesh

```

```

453 save('X.txt','XX','-ascii');
454 save('Y.txt','YY','-ascii');
455
456 % Save u and B
457 save('u.txt','u','-ascii');
458 save('B.txt','B','-ascii');
459
460 % Save ion mass
461 save('mi.txt','mi','-ascii');
462
463 % Save Fourier mesh type
464 save('FourierMeshType.txt','FourierMeshType','-ascii');
465
466 % If using constant collision frequencies, save them here
467 if constCollSwitch == 1
468     collParam = [nuin , nuie , nuui , nuen , nuee , nuei];
469     save('collisions.txt','collParam','-ascii');
470 end
471
472 % Save other parameters
473 savedParam = [mi mn FourierMeshType nn ri rn];
474 save('param.txt','savedParam','-ascii');
475
476 % Save diffusion constants
477 save('D.txt','D','-ascii');
478
479 % Make and write to a log file
480 fid = fopen('log.txt','w');
481 fprintf(fid, 'L = [%.2e %.2e]\n', L(1), L(2));
482 fprintf(fid, 'n = [%d %d]\n', n(1), n(2));
483 fprintf(fid, 'B = [%.2e %.2e %.2e]\n', B(1), B(2), B(3));
484 fprintf(fid, 'u = [%.2e %.2e %.2e]\n', u(1), u(2), u(3));
485 fprintf(fid, 'FourierMeshType = %d\n', FourierMeshType);
486 fprintf(fid, 'Dn = %.2e\n', D);
487 fprintf(fid, 'mi = %.2e\n', mi);
488 if constColls(1) == 1
489     fprintf(fid, 'nuin = %.2e\n', nuin);
490     fprintf(fid, 'nuie = %.2e\n', nuie);
491     fprintf(fid, 'nuen = %.2e\n', nuen);
492     fprintf(fid, 'nuei = %.2e\n', nuei);
493     fprintf(fid, 'nuui = %.2e\n', nuui);
494     fprintf(fid, 'nuee = %.2e\n', nuee);
495 else
496     fprintf(fid, 'mn = %.2e\n',mn);
497     fprintf(fid, 'nn = %.2e\n',nn);
498     fprintf(fid, 'ri = %.2e\n',ri);
499     fprintf(fid, 'rn = %.2e\n',rn);
500 end
501
502 if aliasType == 0
503     fprintf(fid, '\nUsing no dealiasing algorithm\n\n');

```

```

504     elseif aliasType == 1
505         fprintf(fid, '\nUsing zero-padding dealiasing\n\n');
506     end
507     fclose(fid);
508
509 else % If restarting from a dataset
510     % Load variables
511     for j = 1:4
512         prim_var_pert(:,:,j) = load([varNames{j} '_' num2str(counter) '.txt']);
513         prim_var_unpert(:,:,j) = load([ varNames{j} '_unpert.txt']);
514     end
515
516     % Get total variables
517     prim_var = prim_var_pert + prim_var_unpert;
518
519     % Calculate pressures
520     % Unperturbed
521     prim_var_unpert(:,:,5) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,3);
522     prim_var_unpert(:,:,6) = kb * prim_var_unpert(:,:,2) .* prim_var_unpert(:,:,4);
523
524     % Total
525     prim_var(:,:,5) = kb * prim_var(:,:,2) .* prim_var(:,:,3);
526     prim_var(:,:,6) = kb * prim_var(:,:,2) .* prim_var(:,:,4);
527
528     % Perturbed
529     prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
530     prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
531
532     % Loading time data
533     t = load(['t_' num2str(restartFrame) '.txt']);
534
535     timeFileID = fopen('tVec.txt','a');
536
537     % Open electric field energy data
538     fileIDs(1) = fopen('Eenergy_left.txt','a');
539     fileIDs(2) = fopen('Eenergy_right.txt','a');
540
541     % Set the start of the iterations
542     iter_start = 1;
543 end
544
545 % Run simulation
546 runSimulation_inertia_RK4(prim_var_unpert, prim_var_pert, k, ksqu, ksqu_4inv, kmax, t,
    CFL, dt_min, ...
547     dt_max, tend, u, B, mi, ri, mn, nn, rn, D, iter_start, iter_max, err_max, ...
548     phi_iter_max, saveFrequency, counter, varNames, inHeatingSwitch, ieHeatingSwitch, ...
549     constColls, condSwitch, aliasType, xVec, yVec, domain_inds, fileIDs, timeFileID,
    pertType);
550
551 % Close file IDs
552 fclose('all');

```



```

553
554 % Get computational time
555 elapsed_time = toc;
556
557 % Finish up some logging
558 fid = fopen('log.txt','a');
559 fprintf('Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
560 fprintf(fid,'Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
561 fprintf('\n\n\n\n\n\n\n\n');
562 fprintf('Done!\n\n');

```

C.5 plasma_cloud

This input file is for a plasma cloud simulation whose density is modeled by an elliptical Gaussian type function with no background velocity. Depending on the collision frequency, the result will be either a GDI dominated cloud or a KHI dominated cloud.

C.5.1 Initial Conditions

The density is modeled using an elliptical Gaussian type function of the form

$$n_p(x, y) = a \exp \left[- \left(\tilde{r}^2 \left[1 - b \cos(c\theta) \right] \right)^p \right] + d, \quad (\text{C.15})$$

where d is the background plasma density, a is the difference between the background plasma density and the cloud density, b is perturbation amplitude, c is the mode to perturb, p is the power on the Gaussian argument, and \tilde{r}^2 is

$$\tilde{r}^2 = \left(\frac{x - x_0}{r_{0x}} \right)^2 + \left(\frac{y - y_0}{r_{0y}} \right)^2, \quad (\text{C.16})$$

where x_0 and y_0 determine the location of the center of the cloud, r_{0x} determines the spatial extent of the cloud in the x direction, and r_{0y} determines the spatial extent of the cloud in the y direction. The angle for the perturbation, θ , is defined based on the center point of the Gaussian as

$$\theta = \text{atan2}(y - y_0, x - x_0) - \frac{\pi}{c}. \quad (\text{C.17})$$

The angle needs to be modified by $-\pi/c$ to ensure that the perturbation's trough is on the leading edge of the cloud. Note that this initial condition is applied only to the perturbed density to understand the evolution of the entire cloud along with the turbulence. Additionally, this simulation uses normalized values; hence, no units are provided.

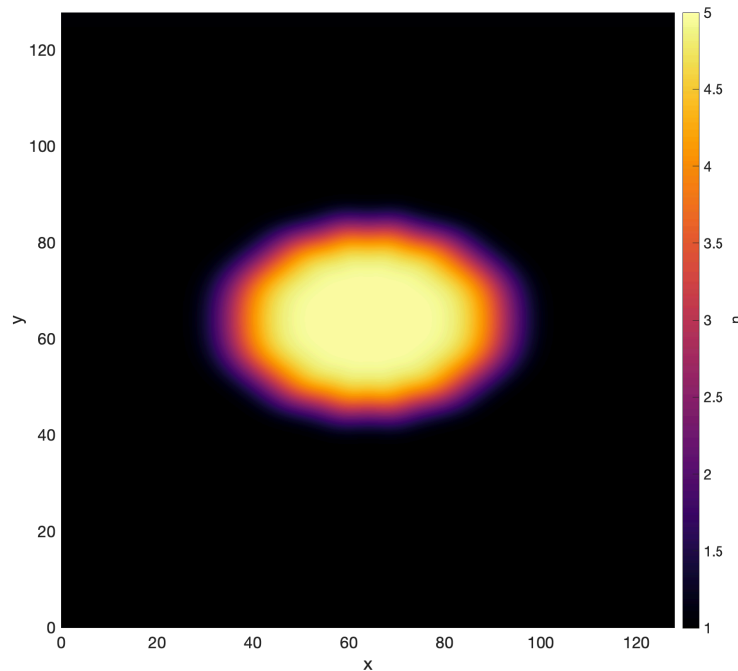


Figure C.4: An example of the density initial condition using Eq. C.15, with $a = 4$, $b = 0.015$, $c = 12$, $d = 1$, $p = 3$, $r_{0x} = 30$, $r_{0y} = 20$, $x_0 = 64$, and $y_0 = 64$. The length of the cloud in the x and y directions is determined by r_{0x} and r_{0y} , respectively.

To ensure that the density does not fall below 0, the quantities d and $a + d$ must both be sufficiently above 0. The exponent, p , determines how steep the density gradient is, which in turn determines how strong the GDI is. Figure C.4 shows an example of the density initial condition using Eq. C.15, with $a = 4$, $b = 0.015$, $c = 12$, $d = 1$, $p = 3$, $r_{0x} = 30$, $r_{0y} = 20$, $x_0 = 64$, and $y_0 = 64$.

C.5.2 Input File

Listing C.5: Input file for a plasma cloud simulation that approximates the density with an elliptical Gaussian type function with no background velocity.

```

1 % This script is for a plasma cloud simulation with an elliptical Gaussian type function
2 % with no background velocity.
3
4 % Note: you will want to run this script in the directory in which you want your data to
   be saved
5
6 % Close and clear everything;
7 close all; clearvars; clc;
8
```

```

9 % Set up the path here. This needs to be changed depending on where you have the code
   saved
10 % addpath('location/to/spacesciencescodes/matlabCodes/keskinen/');
11
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
13 % Everything in this section is for setting up the simulation.
14 % These parameters are what you want to change for running different simulations.
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16
17 % Save data every saveFrequency number of time steps.
18 % For more frames, decrease this number.
19 % For fewer frames, increase this number.
20 saveFrequency = 10;
21
22 % This is the maximum allowable time step in s.
23 % This makes sure that the time step does not skip over important physics
24 % This is set with some a priori understanding of the simulation
25 % It might need to be iterated upon and refined to get an idea of what's going on.
26 dt_max = 20;
27
28 % This is the minimum allowable time step in s.
29 % It is possible that due to the electric potential diverging, the velocity goes to
   infinity.
30 % This results in the time step decreasing to 0.
31 % This can result in the simulation taking a long time despite nothing substantial
   happening
32 % This is used as a check to cancel the simulation since it means the potential is
   blowing up.
33 dt_min = 1e-6;
34
35 % This is the end time in s.
36 % Set this based on when you think you will have enough temporal evolution to your liking
37 tend = 150;
38
39 % This is the maximum allowable error for the iterative method used to calculate the
   electric potential
40 % Increasing this will yield better results but at the cost of additional computational
   effort
41 % It has not been tested yet but setting this too low will result in the required error
   to be below machine precision.
42 % That would result in the potential never being able to be solved (since the error can
   only go as low as machine precision)
43 err_max = 1e-8;
44
45 % Set the Courant-Friedrichs-Lewy condition for numerical stability
46 % This is dependent on the time integration scheme used
47 % Typically, this is left at 0.9 or 1.
48 % Note that there is additional physics that exist in this system of equations
49 % When calculating the time step, the CFL might be met but it might not consider some
   other important physical time scale
50 CFL = 1;

```

```

51
52 % Set the domain length.
53 % First element is the domain length in x direction
54 % Second element is the domain length in y direction
55 L = [128 128];
56
57 % Set the number of grid points in each direction
58 % First element is the number of grid points in the x direction
59 % Second element is the number of grid points in the y direction
60 n = [256 256];
61
62 % Set the artificial diffusion constant for the density source term
63 % This is needed to dissipate numerical error
64 % Note that as this number is increased, everything will generally take longer to run
65 % If this is increased too high, from experience, the simulation might crash.
66 % The solution is to decrease dt_max (however, this results in the code taking very long
    to run).
67 D = .1;
68
69 % This is a flag to choose which type of Fourier mesh to use.
70 % 0 means using the exact definition of k
71 % 1 means using the approximation of k and k^2
72 % See the makeFourierMesh2D function for more information on this.
73 % Depending on the initial conditions chosen, mesh type 1 might need to be chosen to get
    far enough in time
74 FourierMeshType = 1;
75
76 % This is a flag to determine the type of de-aliasing algorithm to use
77 % 0 means using no de-aliasing algorithm
78 % 1 means using the zero pad method (more computationally expensive)
79 % See the convolve2D function for more information on this
80 aliasType = 0;
81
82 % Set the magnetic field in T
83 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
84 % This code really only works for B purely in the z direction (third element) at the
    moment
85 % So, only initialize the third element and leave the first two 0.
86 B = [0 0 1];
87
88 % Set the neutral wind in m/s.
89 % It is defined as a vector but only parts of the code are general enough to deal with
    all elements
90 % This code really only works for u purely in the xy plane (first and second elements)
    at the moment
91 % Soe only initialize the first two elements (x and y respectively) and leave the third
    one 0.
92 u = [10 0 0];
93
94 % The potential is initialized to be 0

```

```
95 phi0 = 0;
96
97 % Set the location in x for center of the plasma cloud
98 % It is presently set to be the center of the domain but it does not have to be
99 x0 = L(2)/2;%L(1) *.8;
100
101 % Set the location in y for center of the plasma cloud
102 % It is presently set to be the center of the domain but it does not have to be
103 y0 = L(2) / 2;
104
105 % Set the extent of the elliptical plasma cloud in the x direction
106 r0x = 20;
107
108 % Set the extent of the elliptical plasma cloud in the y direction
109 % This is set to be the same as r0x to make a circular cloud but it does not have to be
110 r0y = 20;
111
112 % Set the density of the background plasma
113 % Needs to be greater than 0
114 d = 1;
115
116 % Set the density difference between the plasma cloud and background plasma
117 % This should be defined such that d + a yields the desired plasma cloud density
118 % Therefore, set a such that d + a is positive.
119 a = 4;
120
121 % Set the power of the Gaussian function.
122 % A higher power yields a steeper gradient
123 % Which in turn means a faster GDI development
124 p = 3;
125
126 % Set the strength of the perturbation
127 b = 0.015;
128
129 % Set the mode to perturb
130 c = 12;
131
132 % The temperatures are set but are not really used in this simulation
133
134 % Set a constant ion temperature in K
135 % The ion temperature is assumed to be constant in the entire domain
136 Ti0 = 1000;
137
138 % Set a constant electron temperature in K
139 % The electron temperature is assumed to be constant in the entire domain
140 Te0 = 1000;
141
142 % Set the neutral number density in 1/m^3
143 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
144 % See the calcCollFreqk function for more information.
```

```
145 % Note that the model assumes F region ionosphere, therefore if nn is too large, the
      code will crash
146 % The assumption that nu/Omega is small will break.
147 nn = 1e14;
148
149 % Set the radius of the neutral particles in m
150 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
151 % See the calcCollFreqk function for more information.
152 rn = 152e-12; % This specific value assumed O is dominant neutral species
153
154 % Set the radius of the ions in m
155 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
156 % See the calcCollFreqk function for more information.
157 ri = 152e-12; % This specific value assumed O+ is dominant ion species
158
159 % Set the mass of the neutrals in kg
160 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
161 % See the calcCollFreqk function for more information.
162 mi = 16 * 1.66053906660e-27; % This specific value assumed O is dominant neutral species
163
164 % Set the mass of the ions in kg
165 % This is used to calculate the ion-neutral and electron-neutral collision frequencies
166 % See the calcCollFreqk function for more information.
167 mn = 16 * 1.66053906660e-27; % This specific value assumed O+ is dominant ion species
168
169 % The code calculates the collision frequencies self-consistently
170 % There may be situations in which you want to specify a constant set of frequencies
171 % constCollSwitch determines if you want to use the self-consistent or pre-defined
      frequencies
172 % 0 is off which means that you are using the self-consistent collision frequencies
173 % 1 is on which means that you are using the pre-defined constant collision frequencies
174 constCollSwitch = 1;
175
176 % If constCollSwitch is 0, then neglect the next few lines
177 % If constCollSwitch is 1, then set the collision frequencies
178 nuin = 1; % ion-neutral
179 nuui = 0; % ion-ion
180 nuie = 0; % ion-electron
181 nuen = 0; % electron-neutral
182 nuei = 0; % electron-ion
183 nuee = 0; % electron-electron
184
185 % The calcCollFreqk function takes all of the above set of a parameters in as a single
      vector
186 constColls = [constCollSwitch , nuin , nuui , nuie , nuen , nuei , nuee];
187
188 % This is a flag to determine if the ion-neutral frictional heating source terms will be
      included
189 % 0 means off
190 % 1 means on
191 inHeatingSwitch = 0;
```

```

192
193 % Set the neutral temperature in K
194 % This is only needed if inHeatingSwitch == 1
195 % Otherwise, this is not used in the code
196 % Note that for the code to work properly, this needs to be a 2D matrix
197 % This can allow for non-uniform neutral temperatures
198 Tn = 1000 * ones(n);
199
200 % This is a flag to determine if the ion-electron frictional heating source terms will
      be included
201 % 0 means off
202 % 1 means on
203 ieHeatingSwitch = 0;
204
205 % This is a flag to determine if the thermal conduction source terms will be included
206 % 0 means off
207 % 1 means on
208 condSwitch = 0;
209
210 % Choose frame to restart.
211 % 0 means that the simulation will start from the beginning and make the initial
      conditions, then run.
212 % For any other integer (> 0), it will go to that frame and rerun the simulation from
      that start point
213 restartFrame = 0;
214
215 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
216 % Everything past here does not need to be changed to run this simulation.
217 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
218
219 % Set the counter to the frame to restart so that new frames are saved accordingly and
      not overwritng old ones
220 counter = restartFrame;
221
222 % Setup max time iterations so that the code has some finite end iteration.
223 iter_max = 100000;
224
225 % Setup max iterations for potential equation.
226 % If the iterations go over this, the code will stop because the potential is not
      converging.
227 phi_iter_max = 500;
228
229 % Set pertType. This is not used.
230 % Code needs to be chagned a bit to allow for different diagnostics during run time
231 pertType = 4;
232
233 % Set fundamental physical constants
234 me = 9.1093837015e-31; % kg
235 e = 1.602176634e-19; % C
236 kb = 1.380649e-23; % J/K
237 eps0 = 8.8541878128e-12; % F/m

```

```

238
239 % Get computational time
240 tic;
241
242 % Calculate the magnitude of B and its square
243 Bmag = norm(B);
244 B2 = Bmag^2;
245
246 % Calculate gyrofrequencies
247 Oce = e * Bmag / me;
248 Oci = e * Bmag / mi;
249
250 % Calculate Cm which is a constant that shows up frequently in the potential source term
251 Cm = (1 / Oci + 1/Oce)^-1;
252
253 % Make spatial mesh
254 [XX, YY, kmax] = makeSpatialMesh2D(L, n);
255
256 % Get a vector of just x
257 xVec = XX(:,1);
258
259 % Get a vector of just y
260 yVec = YY(1,:)';
261
262 % Set the indices of the two regions from which to get the E field energy spectrum
263 domain_inds{1} = 1:n(1)/4;
264 domain_inds{2} = ( n(1)/4+1 ) : n(1)/2;
265
266 % Set fileIDs and names of energy spectrum data
267 fileIDs(1) = fopen('Energy_left.txt','w');
268 fileIDs(2) = fopen('Energy_right.txt','w');
269
270 % Make a Fourier mesh
271 [k, ksqu, ksqu_4inv] = makeFourierMesh2D(L, n, FourierMeshType);
272
273
274 % Set fileIDs for time vector
275 timeFileID = fopen('tVec.txt','w');
276
277 % Note that the definition of prim_var is
278 % 1: phi
279 % 2: ne
280 % 3: Ti
281 % 4: Te
282 % 5: Pi
283 % 6: Pe
284
285 % Make the names for each primitive variable
286 varNames = {'phi' , 'ne' , 'Ti' , 'Te' , 'Pi' , 'Pe'};
287
288

```



```

289
290 % Check if we are restarting or starting a new simulations
291 if restartFrame == 0 % This means we are starting a new simulation
292
293     % Set the initial conditions for the background
294
295     % Preallocate the unperturbed variables as all zeros
296     prim_var_unpert = zeros([n 6]);
297
298     % Because there is no background velocity, potential is already set to 0
299
300     % Because we want to see the evolution of the entire cloud, the background density
301     % is set to 0
302     % All of the density will be initialized in the perturbed variable
303
304     % Set the background ion temperature
305     prim_var_unpert(:, :, 3) = Ti0;
306
307     % Set the background electron temperature
308     prim_var_unpert(:, :, 4) = Te0;
309
310     % Preallocate all perturbations as all zeros
311     prim_var_pert = zeros([size(XX) 6]);
312
313     % Initialize the density in the perturbed variables
314
315     % Calculate the normalized and squared radius based on the center of the cloud
316     rtilde2 = (XX - x0).^2/r0x^2 + (YY-y0).^2/r0y^2;
317
318     % Calculate the angle with respect to the center point (x0, y0);
319     % Note that it needs to be modified by -pi/c to make sure that a perturbation
320     % exists on the leading edge
321     theta = atan2(YY-y0, XX- x0) - pi/c;
322
323     % Set the density initial condition
324     prim_var_pert(:, :, 2) = a * exp( - ( rtilde2 .* (1 - ...
325         b * cos( c * theta ) ) ).^p ) + d;
326
327     % Get total variable by adding background and perturbation
328     prim_var(:, :, 1:4) = prim_var_unpert(:, :, 1:4) + prim_var_pert(:, :, 1:4);
329
330     % Calculate pressures
331     % Unperturbed
332     prim_var_unpert(:, :, 5) = kb * prim_var_unpert(:, :, 2) .* prim_var_unpert(:, :, 3);
333     prim_var_unpert(:, :, 6) = kb * prim_var_unpert(:, :, 2) .* prim_var_unpert(:, :, 4);
334
335     % Total
336     prim_var(:, :, 5) = kb * prim_var(:, :, 2) .* prim_var(:, :, 3);
337     prim_var(:, :, 6) = kb * prim_var(:, :, 2) .* prim_var(:, :, 4);
338

```

```

339 % Perturbed
340 prim_var_pert(:,:,5) = prim_var(:,:,5) - prim_var_unpert(:,:,5);
341 prim_var_pert(:,:,6) = prim_var(:,:,6) - prim_var_unpert(:,:,6);
342
343 % Save data
344 save3DMat(prim_var_unpert(:,:,1:4), varNames, '_unpert');
345 save3DMat(prim_var_pert(:,:,1:4), varNames, '_0');
346
347 % Initialize a time vector
348 t = 0;
349 iter_start = 1;
350
351 save('t_0.txt','t','-ascii');
352
353 % Save mesh
354 save('X.txt','XX','-ascii');
355 save('Y.txt','YY','-ascii');
356
357 % Save u and B
358 save('u.txt','u','-ascii');
359 save('B.txt','B','-ascii');
360
361 % Save ion mass
362 save('mi.txt','mi','-ascii');
363
364 % Save Fourier mesh type
365 save('FourierMeshType.txt','FourierMeshType','-ascii');
366
367 % If using constant collision frequencies, save them here
368 if constCollSwitch == 1
369     collParam = [nuin , nuie , nuii , nuen , nuee , nuei];
370     save('collisions.txt','collParam','-ascii');
371 end
372
373 % Save other parameters
374 savedParam = [mi mn FourierMeshType nn ri rn];
375 save('param.txt','savedParam','-ascii');
376
377 % Save diffusion constants
378 save('D.txt','D','-ascii');
379
380 % Make and write to a log file
381 fid = fopen('log.txt','w');
382 fprintf(fid, 'L = [%.2e %.2e]\n', L(1), L(2));
383 fprintf(fid, 'n = [%d %d]\n', n(1), n(2));
384 fprintf(fid, 'B = [%.2e %.2e %.2e]\n', B(1), B(2), B(3));
385 fprintf(fid, 'u = [%.2e %.2e %.2e]\n', u(1), u(2), u(3));
386 fprintf(fid, 'FourierMeshType = %d\n', FourierMeshType);
387 fprintf(fid, 'Dn = %.2e\n', D);
388 fprintf(fid, 'mi = %.2e\n', mi);
389 if constColls(1) == 1

```

```

390     fprintf(fid, 'nuin = %.2e\n', nuin);
391     fprintf(fid, 'nuie = %.2e\n', nuie);
392     fprintf(fid, 'nuen = %.2e\n', nuen);
393     fprintf(fid, 'nuei = %.2e\n', nuei);
394     fprintf(fid, 'nuui = %.2e\n', nuei);
395     fprintf(fid, 'nuee = %.2e\n', nuei);
396 else
397     fprintf(fid, 'mn = %.2e\n', mn);
398     fprintf(fid, 'nn = %.2e\n', nn);
399     fprintf(fid, 'ri = %.2e\n', ri);
400     fprintf(fid, 'rn = %.2e\n', rn);
401 end
402
403 if aliasType == 0
404     fprintf(fid, '\nUsing no dealiasing algorithm\n\n');
405 elseif aliasType == 1
406     fprintf(fid, '\nUsing zero-padding dealiasing\n\n');
407 end
408 fclose(fid);
409
410 else % If restarting from a dataset
411     % Load variables
412     for j = 1:4
413         prim_var_pert(:, :, j) = load([varNames{j} '_' num2str(counter) '.txt']);
414         prim_var_unpert(:, :, j) = load([ varNames{j} '_unpert.txt']);
415     end
416
417     % Get total variables
418     prim_var = prim_var_pert + prim_var_unpert;
419
420     % Calculate pressures
421     % Unperturbed
422     prim_var_unpert(:, :, 5) = kb * prim_var_unpert(:, :, 2) .* prim_var_unpert(:, :, 3);
423     prim_var_unpert(:, :, 6) = kb * prim_var_unpert(:, :, 2) .* prim_var_unpert(:, :, 4);
424
425     % Total
426     prim_var(:, :, 5) = kb * prim_var(:, :, 2) .* prim_var(:, :, 3);
427     prim_var(:, :, 6) = kb * prim_var(:, :, 2) .* prim_var(:, :, 4);
428
429     % Perturbed
430     prim_var_pert(:, :, 5) = prim_var(:, :, 5) - prim_var_unpert(:, :, 5);
431     prim_var_pert(:, :, 6) = prim_var(:, :, 6) - prim_var_unpert(:, :, 6);
432
433     % Loading time data
434     t = load(['t_' num2str(restartFrame) '.txt']);
435
436     timeFileID = fopen('tVec.txt', 'a');
437
438     % Open electric field energy data
439     fileIDs(1) = fopen('Eenergy_left.txt', 'a');
440     fileIDs(2) = fopen('Eenergy_right.txt', 'a');

```

```
441
442     % Set the start of the iterations
443     iter_start = 1;
444 end
445
446 % Run simulation
447 runSimulation_inertia_RK4(prim_var_unpert, prim_var_pert, k, ksqu, ksqu_4inv, kmax, t,
    CFL, dt_min, ...
448     dt_max, tend, u, B, mi, ri, mn, nn, rn, D, iter_start, iter_max, err_max, ...
449     phi_iter_max, saveFrequency, counter, varNames, inHeatingSwitch, ieHeatingSwitch, ...
450     constColls, condSwitch, aliasType, xVec, yVec, domain_inds, fileIDs, timeFileID,
    pertType);
451
452 % Close file IDs
453 fclose('all');
454
455 % Get computational time
456 elapsed_time = toc;
457
458 % Finish up some logging
459 fid = fopen('log.txt','a');
460 fprintf('Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
461 fprintf(fid,'Total elapsed time: %.2f hours\n\n', elapsed_time / 3600);
462 fprintf('\n\n\n\n\n\n\n\n');
463 fprintf('Done!\n\n');
```

Appendix D

Post-Processing Scripts

D.1 Best Fit Exponential

For all of the instabilities considered in this dissertation, the growth occurs as an increase in the electric field. In the linear growth phase, the growing component of a plasma instability is assumed to be of the form

$$E(t) = E_1 \exp(\gamma t), \quad (\text{D.1})$$

where γ is the growth rate and E_1 is the initial perturbation. In this case, because the electric field is a vector quantity, it is easier to study the growth of the domain-integrated electric field energy, which is of the form $\int E^2 d\Omega = \int (E_x^2 + E_y^2 + E_z^2) d\Omega$. Typically, if the actual value of the electric field energy is desired, it is also multiplied by $\varepsilon_0/2$. However, in this case, only the growth is important, which is unchanged by any sort of constant multiplicative factor. This electric field energy should instead follow

$$\int E^2(t) d\Omega = \tilde{E}_1 \exp(2\gamma t), \quad (\text{D.2})$$

where \tilde{E}_1 is some initial energy perturbation. The energy uses this exponential form because it is the square of the electric field. Typically, this is a straight forward calculation that any standard least squares regression package can solve. However, the approximation in Eq. D.1 is only valid in the linear regime. As the instability grows, it enters the nonlinear regime, which does not have an analytical approximation for its growth.

As an example, the domain-integrated electric field energy from a single mode slab GDI simulation described by Table 3.2 with a y domain length of 7.8125 km is plotted versus time in Figure D.1. Early in time, the energy appears to follow an exponential, but it clearly does not late in time. Visually, the exponential might fit best from $t = 0$ s to $t = 2000$ s. This, however, is not an ideal method as it introduces a subjective uncertainty. Instead, a method can be used that looks at the many possibilities for the linear growth region and chooses the exponential curve with the best fit from those possibilities.

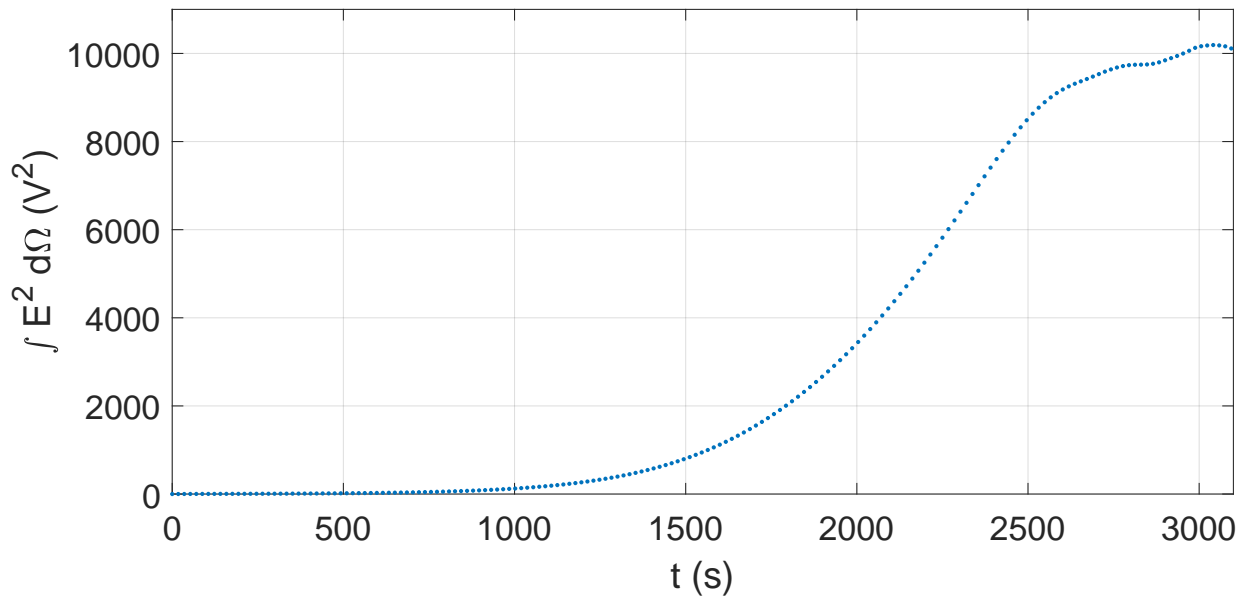


Figure D.1: Domain-integrated electric field energy vs time for the single mode slab GDI simulation described by Table 3.2 with a y domain length of 7.8125 km. Note how it appears to follow the form of an exponential early in time. At later times, nonlinear effects begin to dominate, with the growth no longer resembling Eq. D.1.

Consider a set of data with points indexed from 1 to N . First, an arbitrary starting point, N_{start} , is chosen. The following method will work regardless of choice, but from experience, a starting point of 1 ($t = 0$ s) often results in the best fit. A minimum number of points, N_{min} , is used to make sure that the fit spans a reasonable number of points. Otherwise, fits of just two points would be shown to have a perfect fit. Typically, this minimum is set to at least 10, but this also depends on the amount of data available. The total number of points in the dataset is defined as N .

With a starting point and a minimum number of points, the obvious next step is to use a least square regression on this subset of points (N_{start} to $N_{\text{start}} + N_{\text{min}} - 1$) to find the best fit exponential function and the associated r^2 value. The goal is to find the subset of points that gives an r^2 value closest to 1. Since r^2 is always between 0 and 1, this is equivalent to maximizing the r^2 value. Therefore, by increasing the number of points one by one, i.e., N_{start} to $N_{\text{start}} + N_{\text{min}}$, N_{start} to $N_{\text{start}} + N_{\text{min}} + 1$, N_{start} to $N_{\text{start}} + N_{\text{min}} + 2$, ..., N_{start} to N , and using the least squares regression for an exponential function, the linear regime can be found by using the subset of data with the maximum r^2 value. This results in a total of $N - N_{\text{min}} - N_{\text{start}} + 2$ iterations.

For example, consider a set of data with a total of $N = 100$ data points, minimum of $N_{\text{min}} = 10$ points for a fit, and a start point of $N_{\text{start}} = 3$. Then, the first iteration will find the best fit exponential using points 3 to 12 (N_{start} to $N_{\text{start}} + N_{\text{min}}$). The next iteration will

find the best fit exponential using points 3 to 13 (N_{start} to $N_{\text{start}} + N_{\text{min}} + 1$); the iteration after will do the same using points 3 to 14 (N_{start} to $N_{\text{start}} + N_{\text{min}} + 2$). This pattern continues until the subset is points 3 to 100 (N_{start} to N) for a total of 89 ($N - N_{\text{min}} - N_{\text{start}} + 2$) iterations. At each iteration, the r^2 value is saved. The set of points that gives the maximum r^2 value is then defined as the linear region and used to calculate the growth rate.

The method described will definitely find the linear growth region, but it will also be computationally inefficient for large datasets. With the example of 100 points, the solution will be reached in a reasonable amount of time. But, as the number of points increases, the computational time greatly increases. Not only is the number of iterations increased ($N - N_{\text{min}} - N_{\text{start}} + 2$), the time for each successive least squares regression increases because each regression has to fit an additional point. Therefore, a better method is developed to minimize the number of iterations required to obtain the best fit. As opposed to brute forcing through each possible subset of points, only certain subsets are considered in order to converge to a solution more quickly. The method works by iterating through specific subsets from coarse to fine resolution to obtain the linear regime.

The initial order of magnitude, p , is defined as $p = \text{floor}(\log_{10} N)$. The subsets used for the first round of iterations are N_{start} to 10^p , N_{start} to 2×10^p , ..., N_{start} to $j \times 10^p$, until $j \times 10^p > N$, at which point the last subset considered is N_{start} to N . Therefore, there can be at most 10 subsets for this initial set of iterations. If any of these subsets have fewer points than the minimum N_{min} points, then those subsets are discarded for this analysis. For the remaining subsets, a least squares regression to an exponential function is run and the subset with the largest r^2 value is considered the best fit. The end point of this best fit is defined as N_{best} . This best fit only has a resolution of 10^p points. To obtain a higher resolution, the next lower order of magnitude, $p - 1$, is considered about N_{best} . Since the subset of points from N_{start} to N_{best} has the best r^2 value at a resolution of 10^p , it is assumed that the best r^2 value at a resolution of 10^{p-1} exists somewhere between the endpoints of $N_{\text{best}} - 9 \times 10^{p-1}$ and $N_{\text{best}} + 9 \times 10^{p-1}$. Therefore, the next set of subsets to iterate through are N_{start} to $N_{\text{best}} - 9 \times 10^{p-1}$, N_{start} to $N_{\text{best}} - 8 \times 10^{p-1}$, ..., N_{start} to $N_{\text{best}} + 8 \times 10^{p-1}$, and N_{start} to $N_{\text{best}} + 9 \times 10^{p-1}$. Note that the r^2 value for N_{start} to N_{best} is already known from the previous set of iterations and does not need to be recalculated. This results in a maximum of 18 additional iterations (9 on either side of N_{best}). Any cases where the endpoints are greater than N or the total number of points is less than N_{min} are discarded. A least squares regression to an exponential is run on the remaining subsets to obtain a new best r^2 value at a new N_{best} . Then this process is continued using 10^{p-2} , 10^{p-3} , etc. until the resolution reaches $10^0 = 1$ point. The final resulting N_{best} is the best fit linear regime for instability growth. This set of points is used to obtain the growth rate. This method results in iterating through at most $10 + 18(p - 1)$ subsets of data, which will always be fewer than the earlier method that uses $N - N_{\text{min}} - N_{\text{start}} + 2$.

As an example, consider a set of data with $N = 958$, $N_{\text{min}} = 100$, and $N_{\text{start}} = 5$. The order of magnitude of the number of points is $p = \text{floor}(\log_{10}[958]) = 2$, resulting in an initial resolution of $10^2 = 100$ points. Therefore, the first set of subsets are from points 5 to 100,

points 5 to 200, points 5 to 300, ..., points 5 to 900, and points 5 to 958. Because the first subset, points 5 to 100, has less than 100 (N_{\min}) points, it is discarded, leaving 9 subsets. A least squares regression is run on each subset. Suppose the best fit occurs using the subset of points 5 to 900. The next set of iterations considers a resolution of $10^{p-1} = 10^{2-1} = 10^1 = 10$ points. Thus, these subsets are from points 5 to 810, points 5 to 820, ..., points 5 to 890, points 5 to 910, points to 920, ..., points 5 to 980, and points 5 to 990. The last 4 subsets (ending at points 960, 970, 980, and 990, respectively) are discarded because the endpoint is greater than 958 (N), leaving 14 subsets. Note that the subset of points 5 to 900 is not used because the least squares regression for this subset was already calculated. Suppose in this case that the best least squares regression for these subsets is from points 5 to 880. Then the next, and in this case last, set of iterations considers a resolution of $10^{p-2} = 10^{2-2} = 10^0 = 1$ point with the subsets being points 5 to 871, points 5 to 872, ..., points 5 to 879, points 5 to 881, points 5 to 882, ..., points 5 to 888, and points 5 to 889 for a total of 18 subsets. Whatever subset has the highest r^2 value in this case would be the best fit for the regime of linear instability growth. For this example, this method only took 41 iterations whereas the previous method would have taken 855 iterations.

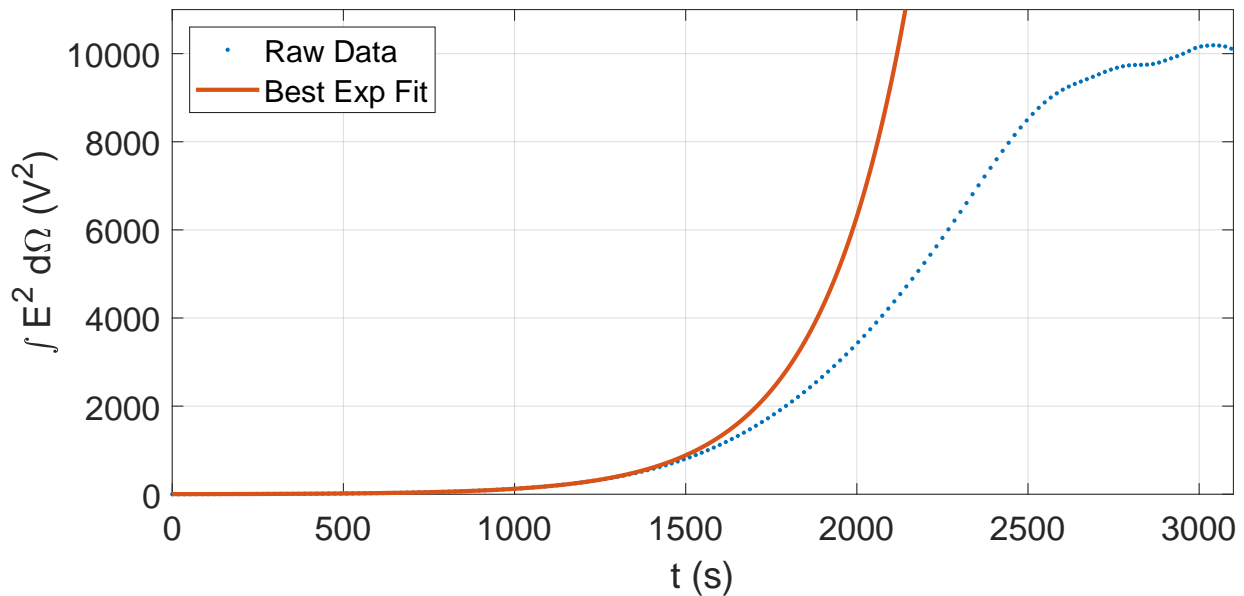


Figure D.2: A plot of the domain-integrated electric field energy vs time showing the raw data (blue dots) and the best fit exponential (red line) for the same case as in Figure D.1. The results show that the best subset of points is point 1 to point 688, which corresponds to a time range of $t = 0$ to $t = 1207$ s and a growth rate of $\gamma = 0.002 \text{ s}^{-1}$.

Figure D.2 shows the same domain-integrated electric field energy data (blue dots) as in Figure D.1. Using the method described above, the best fit exponential curve, starting at point 1 or $t = 0$ s, spans the regime from points 1 to 688, which corresponds to a time span of $t = 0$ to $t = 1207$ s. This is significantly different from the visual approximation

provided earlier, which shows that humans cannot reliably determine where the exponential approximation ends. The resulting growth rate is $\gamma = 0.002 \text{ s}^{-1}$ with $r^2 = 0.99994128389163$. Therefore, this method can be used to find the linear regime of any set of data. The code is shown in Listing D.1.

Listing D.1: Function that finds the best fit exponential within a set of datapoints.

```

1 function [abest, bbest, rsquaredmax, bestNumPoints] = getNumGrowthFun(x, y, startIndex)
2
3 % This function will take in data (x and y) and fit it to the best exponential
4 % It will iterate through the data to get the best fit
5 % If specified, it will start at startIndex, else, start at 1
6
7 minNumPoints = 25;
8 % Default is 1
9 if ~exist('startIndex','var')
10     startIndex = 1;
11 end
12
13 % Since there are potentially a large amount of points, start with a broad search.
14 % Assume that we can narrow the search by looking only around the the point of highest
15     rsquared from the previous broad search.
16
17 % Start out by getting the order of magnitude of the number of points
18 order = floor(log10(length(x)));
19
20 % Set rsquaredmax to 0.
21 rsquaredmax = -1000;
22
23 % Start a for loop.
24 % We will search only by orders of magnitude starting with the highest and ending at
25     10^0.
26 % Thus, we will have accuracy to 10^0 points without having to iterate through the
27     entire dataset.
28 % At most, we look at 10 + 19*(order-1) sets of points, as opposed to length(x) sets of
29     points.
30 % This hinges on the assumption that we can narrow down the search by looking only at
31     the highest rsquared point from the last search.
32
33 NX = length(x);
34
35 for i = order:-1:0
36     fprintf('Looking at order %d\n',i);
37     if i == order
38         % This is the first iteration. Set the stop condition to the end of the dataset
39             since no previous search occurred.
40
41         % The endIndex represents the index of the last point in the dataset that we want
42             to look at.
43         % For example, if we want to look at only the first 205 points of data, then

```

```

    endIndex = 205.
38     % endIndex will change since we want to iterate through different sets of points
    hence we will set it as our looping index
39     endIndex = [(1:floor(length(x) / 10^order)) * 10^order NX];
40
41     else
42         % We now have a value for bestNumPoints since we have gone through the first
            iteration.
43         % Based on our previous search, we only have resolution to the previous order of
            magnitude.
44         % We will now look at a finer order of manitude resolution.
            endIndex = (bestNumPoints - 10^(i+1)) : 10^i : (bestNumPoints + 10^(i+1));
45     end
46     endIndex( endIndex > NX ) = [];
47
48     % Start another loop to iterate through the loop end points.
49     for j = endIndex
50         % Only do this if we have enough points.
51         % Set the minimum number of points to 100. Can change later if necessary.
52         if j - startIndex > minNumPoints
53             % Fit the points from start to j to an exponential.
54             [f , rsquared] = fit(x(startIndex:j)-x(startIndex),y(startIndex:j) , 'exp1');
55             % Print the rsquared value.
56             % rsquared.rsquare
57             % If the rsquared value is higher than what we previously obtained, then save
58             these points.
59             % Note that this exponential is of the form a*exp(b*x)
60             if rsquared.rsquare >= rsquaredmax
61                 rsquaredmax = rsquared.rsquare;
62                 abest = f.a;
63                 bbest = f.b;
64                 % bestNumPoints tells is which point we found the best fit so we can
                    narrow our search.
65                 bestNumPoints = j;
66             end
67         end
68     end
69 end
70 fprintf('Done! \n\n');

```

D.2 Calculating Growth Rate as Function of Wavenumber

For a single mode simulation, the method from Appendix D.1 is sufficient to obtain the growth rate. However, for simulations with multiple growing modes, the energy that is integrated over the entire 2D domain will not provide the correct growth rate, as it cannot

differentiate between the growth of each mode. The energy of each individual mode can be found by first calculating the electric field energy as normal (not domain integrated). Then, the one dimensional Fast Fourier Transform (FFT) is taken along a specified direction, either x or y . In all of the results shown in this dissertation, only the y direction is considered based on the chosen geometry of the instability development. Then, the Fourier transformed energy in the chosen direction is integrated over the other direction, i.e., a Fourier transformed energy in y gets integrated in x and vice versa. The result is a set of line integrated electric field energies for each frequency within the domain, as defined in Section 3.4.1 through Eqs. 3.117, 3.118, and 3.120. The code for this method is shown in Listing D.2. These energies can be calculated at each time step to obtain a set of energies for each frequency as a function of time. Then, the method described in Appendix D.1 and Listing D.1 can be used for each frequency, resulting in a growth rate for each wavenumber. ‘

Listing D.2: Function that takes the FFT in one direction and integrates in the other.

```

1 function fftData = FFTmat(data, intX, matIntIndex, fftOptions)
2
3 % Takes the FFT of a 2D matrix indexed by (i,j)
4 % We will have two options.
5 % FFT_options = 0 : We will integrate the data first in the matIntIndex direction. Then
   take the FFT in the other direction
6 % FFT_options = 1 : We will take the FFT first in not the matIntIndex direction. Then
   integrate in the matIntIndex direction.
7 % matIntIndex can be either 1 or 2. This defines the direction in which to integrate.
8 % intX is the vector used for trapezoidal integration
9
10 % Set the index for taking the FFT. It is the direction opposite to the integration.
11 if matIntIndex == 1
12     FFTIndex = 2;
13 elseif matIntIndex == 2
14     FFTIndex = 1;
15 end
16
17 % Get length of matrix in FFTIndex direction
18 L = size(data,FFTIndex);
19
20 % Set the length of the transform to be the next power of 2 above L
21 n = 2^nextpow2(L);
22
23
24 if fftOptions == 0 % Integrate first. Then FFT
25     intData = trapz(intX, data, matIntIndex); % Integrate
26     fullPower = abs( fft(intData,n, FFTIndex) ); % Take FFT % Why divide by L?
27
28
29 elseif fftOptions == 1 % Take FFT first, then integrate
30     fullPower = abs( fft( data,n, FFTIndex) ); % Take FFT
31     fullPower = trapz(intX, fullPower, matIntIndex); % Integrate
32 end

```

```

33
34 % Take one half of the spectrum
35 fftData = fullPower(1:n/2+1);
36
37 % Multiply that half by 2 except for the first element (which is the steady state value)
38 fftData(2:end-1) = 2*fftData(2:end-1);
39 fftData(end) = [];

```

D.3 Turbulence Energy Cascade

When studying turbulent structures, it is useful to understand how the energy flows from larger scales to smaller scales. The turbulence energy cascade shows how the energy scales as a power law with the wavenumber. This can be analyzed by finding the spectrum for a turbulent quantity. In the case of the spectra shown in this dissertation, the perturbed density normalized by the total density, n_p/n , and the perturbed electric potential, ϕ_p , are used. First, the relevant portion of the domain with the turbulence is chosen. This part is somewhat subjective in that there can exist turbulence that might not be visually seen yet. However, the power law that stems from this is not sensitive to the choice of area. Once the area is chosen, the function described in Appendix D.2 and Listing D.2 is used on this subset of data. The result yields an x (y) direction spectrum integrated in y (x). This spectrum is then plotted versus the wavenumber on a log-log scale. In this scale, the result should have a region that generally follows a straight line. From this data, either a general power law can be approximated by overlaying different power laws over the data, or a least squares linear regression can be used to obtain the best fit line for the data. For the work in this dissertation, the first method of approximation by overlaying lines is used.

Listing D.3: Function that takes finds a power law fit for density and electric potential data. Note that this makes use of the FFTMat function from Listing D.2.

```

1 % Get power spectrum to look at turbulence cascade
2 close all; clearvars; clc;
3 % Do this based off of density and potential data.
4
5
6 % This code at present only takes subsection based on the x points
7 % This can be easily generalized to also taking a subsection in y
8
9 % Get the range of x values we want to consider.
10 % These are the indices
11 xStart = 70;
12 xEnd = 250;
13
14 % Load X and Y data
15 XX = load('X.txt');
16 YY = load('Y.txt');

```

```

17
18 % Make a quick plot of density in this point to see if it is a good area
19 frame2Plot = 8; % This is the frame to plot of the late state density.
20
21 ne = load('ne_unpert.txt') + load(['ne_' num2str(frame2Plot) '.txt']);
22 figure;
23 pcolor(XX(xStart:xEnd,:), YY(xStart:xEnd,:), ne(xStart:xEnd,:));
24 colormap inferno;
25 shading flat;
26 colorbar;
27 caxis([9.95e10 2.505e11]);
28
29 %% If it looks good, then move on and run the rest of the script
30
31 clc; close all;
32 % Make a directory name
33 mkdir('powerSpectrum');
34
35 % Make the Fourier mesh
36 n = size(XX);
37 dx = XX(2,1) - XX(1,1);
38 dy = YY(1,2) - YY(1,1);
39 L(1) = max(max(XX)) + dx;
40 L(2) = max(max(YY)) + dy;
41
42 % Make a vector for x direction
43 xVec = XX(:,1);
44 yVec = YY(1,:)';
45
46 % Make a vector for ky
47 kyVec = makeK(L(2), n(2));
48 kyVec = kyVec(1:n(2)/2)*1e3;
49
50 % Make a vector for kx
51 kxVec = makeK(xVec(xEnd) - xVec(xStart) + dx, 2^(nextpow2(xEnd - xStart + 1)));
52 kxVec = kxVec(1:length(kxVec)/2)*1e3;
53
54 k = makeFourierMesh2D(L, n,1);
55
56 % Set parameters to make good subplots.
57 % otherwise we will not get good looking plots.
58 mR = 0.005; % Right margin
59 mL = 0.06; % Left margin
60 mB = 0.125; % Bottom margin
61 mT = 0.059; % Top margin
62 sH = 0.04; % Horizontal spacing between plots
63 sV = 0.02; % Vertical spacing between plots
64
65 H = (1 - mT - mB - sV)/2;
66 W = (1 - mL - mR - sH)/2;
67 fontSize = 16;

```

```

68 ne_bg = load('ne_unpert.txt');
69 % Iterate through i. Define the i you want to iterate through here
70 % These are the frames you want to analyze
71 for i = [ 6 7 8]
72     fprintf('i=%d\n',i);
73     % Load perturbed phi data
74     phi_pert = load(['phi_' num2str(i) '.txt']);
75     ne_pert = load(['ne_' num2str(i) '.txt']);
76     phik = fft2(phi_pert);
77
78     % Get dn over n
79     dn_over_n = ne_pert ./ (ne_bg+ne_pert);
80
81     % Take fft in x direction and integrate in y direction to get spectrum
82     dn_spectrum_x = FFTmat(dn_over_n(xStart:xEnd, :), yVec, 2, 1)';
83     phi_spectrum_x = FFTmat(phi_pert(xStart:xEnd, :), yVec, 2, 1)';
84
85     % Take fft in y direction and integrate in x direction to get spectrum
86     dn_spectrum_y = FFTmat(dn_over_n(xStart:xEnd, :), xVec(xStart:xEnd), 1, 1)';
87     phi_spectrum_y = FFTmat(phi_pert(xStart:xEnd, :), xVec(xStart:xEnd), 1, 1)';
88
89     % Get electric field
90     E = real(ifft2(- calcDerivk(phik, k)));
91
92     f = figure('visible','off','Position',[0 0 1264 935]);
93
94     hAxis = subplot(2,2,1);
95     hAxis.Position = [mL, mB + sV + H, W, H];
96     % Plot density spectrum for kx
97     loglog(kxVec(2:end), dn_spectrum_x(2:end), 'LineWidth',2, 'DisplayName', 'Simulation
98         Data');
99     % Modify this part as needed.
100    % These next 2 variables define the lines to overlay
101    snx1 = -2;
102    fnx1 = 10^6.3*kxVec.^(snx1);
103    hold on;
104    loglog(kxVec(14:end),
105        fnx1(14:end), '--r', 'DisplayName', sprintf('k_x^{%.1f}',snx1), 'LineWidth',2);
106    xticks([.1 1]);
107    xticklabels({'',''});
108    legend('show', 'Location', 'southwest');
109    grid on;
110    ylabel('n_p/n Spectrum');
111    set(gca, 'FontSize',fontSize);
112
113    hAxis = subplot(2,2,2);
114    hAxis.Position = [mL+sH+W, mB + sV + H, W, H];
115    % Plot density spectrum for ky
116    loglog(kyVec(2:end), dn_spectrum_y(2:end), 'LineWidth',2, 'DisplayName', 'Simulation
117        Data');
118    % Modify this part as needed.

```

```

116 % These next 6 variables define the lines to overlay
117 sny1 = -.5;
118 sny2 = -2;
119 sny3 = -6;
120 fny1 = 105.5*kyVec.^(sny1);
121 fny2 = 104*kyVec.^(sny2);
122 fny3 = 103.5*kyVec.^(sny3);
123 hold on;
124
125 loglog(kyVec(13:40),
126     fny1(13:40), '--r', 'DisplayName', sprintf('k_y^{%.1f}', sny1), 'LineWidth', 2);
127 loglog(kyVec(38:end),
128     fny2(38:end), '--g', 'DisplayName', sprintf('k_y^{%.1f}', sny2), 'LineWidth', 2);
129 axis([kyVec(2) kyVec(end) ylim]);
130 loglog(kyVec(38:end),
131     fny3(38:end), '--k', 'DisplayName', sprintf('k_y^{%.1f}', sny3), 'LineWidth', 2);
132 legend('show', 'Location', 'southwest');
133 xticks([.01 1]);
134 xticklabels({'', ''});
135 grid on;
136 set(gca, 'FontSize', fontSize);
137
138 hAxis = subplot(2,2,3);
139 hAxis.Position = [mL, mB , W, H];
140 % Plot potential spectrum for kx
141 loglog(kxVec(2:end), phi_spectrum_x(2:end), 'LineWidth', 2, 'DisplayName', 'Simulation
142     Data');
143 % Modify this part as needed.
144 % These next 2 variables define the lines to overlay
145 spx1 = -1;
146 fpx1 = 108.6*kxVec.^(spx1);
147 hold on;
148 loglog(kxVec(15:end),
149     fpx1(15:end), '--r', 'DisplayName', sprintf('k_x^{%.1f}', spx1), 'LineWidth', 2);
150
151 legend('show', 'Location', 'southwest');
152 xticks([0.1 1]);
153 xticklabels({'0.1', '1'});
154 grid on;
155 ylabel('\phi_p Spectrum')
156 xlabel('k_x (1/km)');
157 set(gca, 'FontSize', fontSize);
158
159 hAxis = subplot(2,2,4);
160 hAxis.Position = [mL+sH+W, mB , W, H];
161 % Plot potential spectrum for ky
162 loglog(kyVec(2:end), phi_spectrum_y(2:end), 'LineWidth', 2, 'DisplayName', 'Simulation
163     Data');
164 % Modify this part as needed.
165 % These next 4 variables define the lines to overlay
166 spy1 = -2;

```

```
161     spy2 = -8;
162     fpy1 = 10^6.5*kyVec.^(spy1);
163     fpy2 = 10^4.1*kyVec.^(spy2);
164     hold on;
165     loglog(kyVec(20:end),
166            fpy1(20:end), '--r', 'DisplayName', sprintf('k_y^{%.1f}', spy1), 'LineWidth', 2);
167     axis([kyVec(2) kyVec(end) ylim]);
168     loglog(kyVec(32:end),
169            fpy2(32:end), '--k', 'DisplayName', sprintf('k_y^{%.1f}', spy2), 'LineWidth', 2);
170     legend('show', 'Location', 'southwest');
171     xticks([ 0.01 .1 ]);
172     xticklabels({'0.01', '0.1'});
173     grid on;
174     xlabel('k_y (1/km)');
175     set(gca, 'FontSize', fontSize);
176
177     print(['powerSpectrum/fig_' num2str(i)], '-dpng');
178
179     close all;
180 end
181 fprintf('Done\n');
```
