A Study of Plasma Dynamics in HIT-SI using Ion Doppler Spectroscopy

Aaron Clark Hossack

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2015

Reading Committee:

Thomas R. Jarboe, Chair

Brian A. Nelson

Uri Shumlak

Program Authorized to Offer Degree:
William E. Boeing Department of Aeronautics and Astronautics

University of Washington

Abstract

A Study of Plasma Dynamics in HIT-SI
using Ion Doppler Spectroscopy

Aaron Clark Hossack

Chair of the Supervisory Committee:
Professor Thomas R. Jarboe
William E. Boeing Department of Aeronautics and Astronautics

The HIT-SI device is a unique experiment which uses inductive helicity injectors to form and sustain a spheromak plasma. The $n = 1$ symmetry of the injectors enables stable spheromak sustainment by imposed-dynamo current drive (IDCD). The experiment is diagnosed with spectroscopy, interferometry, imaging, and internal and surface magnetic probes. Two methods of density reduction and control are presented. A helicon preionization source enables plasma breakdown and operations at densities an order of magnitude lower than previously possible. The system is critical for all operations at injector frequencies greater than 14.5 kHz and single-injector operations. Additionally, a high-speed piezoelectric gas injection valve was developed to enable dynamic injector fueling adjustable on a timescale of ~ 0.5 ms. The focus of this work is on results from an ion Doppler spectrometer (IDS) which was upgraded to multi-chord capability. Two coherent, linear fiber optic cables with small, wide-angle lenses simultaneously collect light from 30 – 40 chords. Additionally, biorthogonal decomposition is used as a novel filtering method for raw data. Impurity radiation measurements of high power plasmas show no toroidal flow associated with toroidal current and temperature evolution which rises during toroidal current ramp-up and falls during current sustainment. Coherent velocity fluctuations show rigid, oscillatory motion of the spheromak plasma driven by the helicity injectors. The coherent motion combined with a lack of magnetic instabilities indicates that the spheromak is stable. Comparisons

with NIMROD and PSI-TET simulations show similar chord-averaged velocity oscillations but fail to show the observed coherent, rigid motion of the spheromak. Additionally, strong flows and reconnection events in simulations which are not observed in the experiment indicate that agreement may improve with higher viscosity. The measured C III temperatures lie between the two codes' estimates. Line-radiation from high-frequency operations (53.5 and 68.5 kHz) was too weak for the camera so an image intensifier had to be used, yielding a sparse data set. Composite measurements from multiple discharges show higher ion temperatures and lower intensity fluctuations than at 14.5 kHz. Negligible toroidal flow is measured at 68.5 kHz, supporting the conclusion that an outward Shafranov shift is due to pressure confinement rather than rotation. Measured β values assuming $T_i = T_e$ and $n_i = n_e$ are in agreement with Grad-Shafranov fitting to magnetic equilibria supporting the claim that high-β pressure confinement was achieved in high frequency operations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The entire HIT-SI team has been extraordinarily supportive and helpful in this research. I would like to thank Professor Jarboe for the opportunity to work on the HIT-SI project and for insight and guidance, Professor Nelson for tirelessly supporting my research endeavors, John Rogers, Jonathan Hayward, and Susan Griffith for technical expertise and assistance, and all the students and postdocs past and present who've contributed ideas and time: David Ennis, Brian Victor, Jonathan Wrobel, Cihan Akcay, Derek Sutherland, Chris Everson, Rian Chandra, Takafumi Hanao, Griff O'Neill, and Mark Chilenski. I especially need to acknowledge Chris Hansen and Kyle Morgan for running dynamic pressure simulations which have been critical for interpreting my data. I would like to thank Professor Nagata for loaning the spectrometer to our group, Eagle Harbor Technologies, Inc. for building and testing the electrodeless plasma source, and the US DoE for funding this research.

I couldn't have done it without the support of my friends and family: my parents, extended family, Sean Wood, Naomi Hubert, and Derek.

Chapter 1

INTRODUCTION

1.1 Motivation for Fusion

It is widely accepted that innovations in non-fossil fuel energy sources are needed to meet the world's growing demand for energy. Wind and solar power are already being implemented, but neither is an optimal solution. They both occupy large swaths of land and their inconsistent nature necessitates huge advances in energy storage and transfer technology in order to provide a significant fraction of the world's energy [21]. The present generation of wind, solar, and other carbon-free energy sources remains uncompetitive with oil, the accelerating use of which poses a threat to the global environment [42]. Nuclear energy has neither the carbon emission problem of petroleum nor the land area and inconsistency problems of solar and wind.

Nuclear fission has been providing many countries with a relatively safe supply of energy for decades, but it too has drawbacks. Risks of fission include nuclear weapons proliferation [7] and failure of containment, as demonstrated by the Fukishima disaster. Many byproducts of fission remain radioactive for hundreds or thousands of years, posing environmental and health risks to future generations.

Nuclear fusion is an attractive energy source because it addresses all the aforementioned problems. A fusion reactor will provide a steady supply of clean energy without the waste problems associated with nuclear fission. The most attractive fuels for fusion are deuterium and tritium (produced from lithium), both in effectively unlimited supply across the globe [50]. Nuclear fusion reactions do not directly produce radioactive waste. The only radioactive waste associated with fusion is the activation of the reactor itself, and materials research promises to reduce or eliminate this problem [11].

Because nuclear fusion has the potential to provide the world with a clean, safe, and nearly limitless supply of energy, many nations are pursuing domestic fusion research and

engaging in collaborations such as the ITER project [1].

1.2 Physics of Nuclear Fusion

The energy released by a nuclear reaction comes from the difference in binding energy between the reactants and the products. In nuclear fission, very heavy elements such as uranium are split apart into lighter elements. The combined mass of the resulting lighter elements is less than the original uranium and the difference comes from the binding energies holding the various nuclei together. The missing mass is released as energy according to the theory of special relativity, $E = mc^2$.

Energy is also released when very light elements are combined in the fusion process, forming more tightly bound products. Iron has the maximum binding energy per nucleon of all the elements— energy is released by fusing elements lighter than iron and breaking apart elements heavier than iron. The most common reaction [36] for fusion energy is

$$\mathrm{D} + \mathrm{T} \rightarrow^4 \mathrm{He}(3.5\ \mathrm{MeV}) + \mathrm{n}(14.1\ \mathrm{MeV}) \tag{1.1}$$

where D is deuterium, a naturally occurring isotope of hydrogen (with one neutron) found in seawater, and T is tritium (a hydrogen isotope with two neutrons). Tritium is not found in nature and undergoes β-decay with a half-life of 12.5 years. Fortunately, tritium can be produced in the shell of a fusion reactor using lithium and the high energy neutron from the D-T reaction:

$$\mathrm{n} +^6 \mathrm{Li} \rightarrow^4 \mathrm{He} + \mathrm{T} + 4.8\ \mathrm{MeV} \tag{1.2}$$

$$\mathrm{n} +^7 \mathrm{Li} \rightarrow^4 \mathrm{He} + \mathrm{T} + \mathrm{n}. \tag{1.3}$$

The 14 MeV neutron poses a serious engineering challenge to reactor designs because it degrades most materials and the reactor wall must incorporate significant shielding. Some

alternative fusion reactions are

$$\mathrm{D} + \mathrm{D} \rightarrow \mathrm{p}(3.02\ \mathrm{MeV}) + \mathrm{T}(1.01\ \mathrm{MeV}) \tag{1.4}$$

$$\mathrm{D} + \mathrm{D} \rightarrow \mathrm{n}(2.45\ \mathrm{MeV}) + {}^{3}\mathrm{He}(0.82\ \mathrm{MeV}) \tag{1.5}$$

The reaction in equation 1.4 yields subsequent D-T reactions. While the deuterium fuel is easier to obtain than tritium, the plasma physics becomes more challenging. The "cross section" of the D-D reactions is much lower than for D-T, meaning the probability of a fusion reaction per collision is lower. A D-D fusion reactor must therefore operate at a higher temperature and with better particle confinement than a D-T reactor.

1.3 Sustaining Fusion

The Coulomb repulsion between deutrium and tritium nuclei is very strong and the nuclei must collide at high relative velocity in order to fuse together. Simply colliding them with a particle accelerator is inefficient because the majority of collisions will be elastic, resulting in wasted energy and no fusion. In order to sustain a fusion reaction the nuclei must be confined in thermal equilibrium so they can elastically collide with minimal losses until they fuse together. The optimum temperature for fusion is of order 100,000,000 Kelvin, which no solid material can withstand.

1.3.1 Confinement Methods

There are essentially three methods for confining particles of such temperature. The most basic is gravitational confinement, which occurs in stars. Their mass is so large that they retain hydrogen isotopes of thousands of degrees on the surface and millions of degrees at their cores. On Earth, the two options at our disposal are inertial confinement and magnetic confinement. Inertial confinement means that the inertia of the particles holds them in close proximity for a sufficient time that fusion occurs. This method of confinement is essentially the beginning of an explosion, so once most of the particles have a "line of sight" to escape the reactions stop. The third method of confinement is with magnetic fields. When deuterium and tritium gas is heated beyond a few thousand degrees the

particles ionize and the gas becomes a plasma of ions and free electrons. Charged particles gyrate around magnetic fields, so by arranging magnetic fields in closed loops or nested "flux surfaces" in a vacuum chamber the plasma can be contained away from the solid walls. This is the method pursued by most fusion experiments including HIT-SI and will be discussed in more detail in the next section.

1.3.2 Triple Product

The standard figure of merit for fusion efficiency is called the triple product,

$$n\tau_E T \tag{1.6}$$

where n is the electron density in m^{-3}, T is the temperature in keV (1 eV = 11,600 K), and τ_E is the energy confinement time in seconds. The energy confinement time is the total energy of the plasma divided by its power loss and is expected to be approximately 1 second for a fusion reactor. The product $n\tau_E T$ must exceed certain values to achieve *breakeven* and *ignition*. Scientific breakeven is defined as the point at which the total fusion power out equals the power needed to sustain the plasma. For the D-T fusion reaction the value is $\approx 1 \times 10^{21}$ m^{-3} s keV. This benchmark is not practical because most of the fusion power escapes with the neutron and is converted into electrical power or waste heat. The point at which is the fusion reaction is truly self sustaining is called ignition. The other product of D-T fusion, ^{4}He, stays in the plasma and transfers its energy to the deuterium and tritium. Ignition is when the plasma is fully sustained by this process, known as *α-heating*. The $n\tau_E T$ value for D-T ignition is $\approx 2 \times 10^{21}$ m^{-3} s keV.

1.4 Magnetic Confinement of Plasmas

The principal force acting on charged particles in a magnetic field is the Lorentz force:

$$\vec{F} = q\left[\vec{E} + \vec{v} \times \vec{B}\right]. \tag{1.7}$$

As mentioned earlier, a particle moving perpendicular to a magnetic field gyrates around the field due to the $\vec{v} \times \vec{B}$ component of the force, and its motion along the field is governed by the electric field, $\vec{E}$. Thus on a scale larger than the gyro motion charged particles are confined to magnetic fields like beads on a thread. It is also clear that applying an electric field drives ions and electrons in opposite directions resulting in a current.

Most fusion-relevant magnetic confinement schemes aim to contain the fusion plasma on fully *closed* magnetic fields (as opposed to *open* fields which intersect the wall). A simple way to imagine a closed field is a solenoid wrapped around to form a "donut" or *toroidal* shape. Plasmas in purely toroidal fields cannot achieve radial force-balance due to the "tire tube" force (toroidal objects under pressure expands radially outward in the absence of an external, counteracting force). Another weakness of predominantly toroidal fields is the *interchange instability*, where field lines confining hot plasma tend to switch places with fields confining cooler plasma until the fields with hot plasma collide with the outer wall of the chamber.

One way to alleviate the interchange problem is to drive a current through the plasma in the toroidal direction. The plasma current generates a *poloidal* field which, when combined with the toroidal field, results in a helical total field. Helical fields form nested *flux surfaces* and can't slip out to the walls as easily. Additionally, to counteract the "tire tube" force a *vertical field* is applied parallel to the central axis of the toroid. The interaction of the plasma current with the vertical field results in a $\vec{j} \times \vec{B}$ force inward. This is the essence of the most common magnetic confinement schemes, namely the tokamak [36] and reversed-field pinch [8].

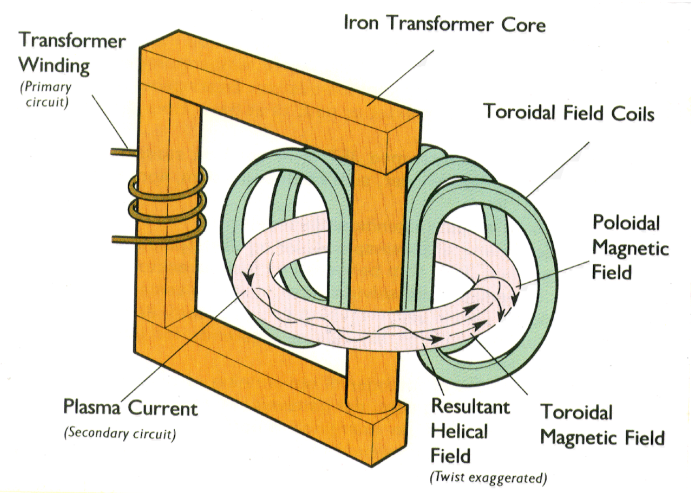

Figure 1.1: Key aspects of tokamak coils and magnetic fields. Image from http://www.fys.uio.no/plasma/plasma/norsk/node4.html

Devices using toroidal confinement chambers have two main drawbacks— how to drive and maintain the toroidal current and the costs associated with a reactor. The best method for driving current in a closed loop is through induction. Most tokamaks have a transformer "primary coil" in the center of the device. When the current through the primary coil in changed a current is induced in the plasma which acts as the "secondary coil". The problem with this method is that it can only be used once during a plasma discharge— there is a practical limit to the current which can be driven though the primary coil to maintain the plasma current. Thus for steady-state plasma confinement less efficient methods for driving current must be used, most involving radio-frequency coupling to ions or electrons [16].

The predominant reactor cost associated with an externally generated toroidal magnetic field is the coils [39]. Modern reactor designs call for large, superconducting coils which must be cooled to at least liquid nitrogen temperatures and must be well shielded from neutron radiation. Alternate confinement designs which eliminate the need for these coils represent a opportunity for significant cost savings. Two designs which have toroidal symmetry but with no coils linking the plasma are the field reversed configuration (FRC) [51] and spheromak [29, 31].

1.5 Spheromaks

Like a tokamak, spheromaks have toroidal and poloidal magnetic fields. However, both fields in a spheromak are generated by currents in the plasma as opposed to using external coils. The magnetic field at the outer edge of a spheromak is purely poloidal and the currents running along these field lines generate a toroidal field in the center (magnetic axis). The currents running along the toroidal field at the magnetic axis generate the poloidal field at the edge, and the fields in between are helical- transitioning between the two extremes (see figure 1.2).

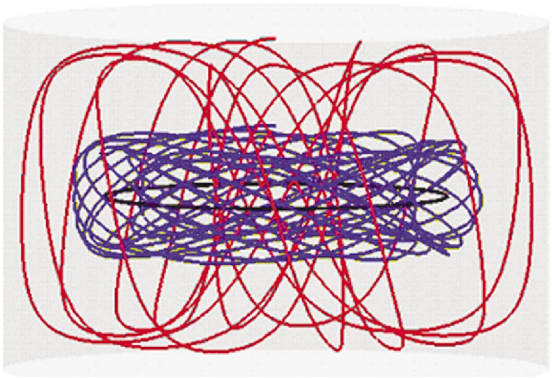

Figure 1.2: Traces of three spheromak field lines in a "tuna can" flux conserver. At the magnetic axis the field is purely toroidal (black) and farther out the field becomes more poloidal (blue, red). Image from Jarboe [31].

1.5.1 Magnetic Helicity

The magnetic topology of spheromaks cannot be discussed further without first reviewing magnetic helicity. Helicity is a measure of the "twist" or linkage of flux. A magnetized plasma can be thought of as a collection of many "flux tubes"— so the helicity of the plasma is the twistedness of each tube and the linkage of the tubes with each other. Helicity has been shown [49] to be the best constant of motion in a plasma. A plasma configuration will organize itself to be in equilibrium and reduce its magnetic energy without changing helicity.

Mathematically, helicity is

$$K = \int \vec{A} \cdot \vec{B}\, dV \tag{1.8}$$

where $\vec{A}$ is the magnetic vector potential and $\vec{B}$ is the magnetic field. This definition is difficult to use experimentally and requires that the volume boundary be closed. To solve these problems, the concept of relative helicity [6] was developed, where

$$K_{rel} = \int (\vec{A} + \vec{A}_{vac}) \cdot (\vec{B} - \vec{B}_{vac})\, dV. \tag{1.9}$$

The plasma and vacuum fields have the same boundary conditions, thus the volume of integration can disregard external fields and still be gauge invariant. Taking the time derivative of this expression yields the change in helicity,

$$\dot{K} = 2 \int (\vec{E}_{vac} \cdot \vec{B}_{vac}) - (\vec{E} \cdot \vec{B})\, dV. \tag{1.10}$$

The first term on the right hand side is the *injection* term and represents the externally imposed E parallel to B. The volume integral can be converted to a surface integral over the boundary and written in terms of applied voltage (V) and magnetic flux (ψ), giving a helicity injection rate of $2V\psi$. The second term is the resistive decay term, which can also be written as $\eta\vec{j} \cdot \vec{B}$, where $\vec{j}$ is the current density and η is the resistivity. Thus helicity decays only by resistive diffusion, which is generally slower than timescales of interest in dynamic plasmas.

1.5.2 Magnetic Topology

Taylor conjectured [48] that spheromaks relax toward a state of minimum energy while conserving helicity. In the minimum energy state magnetic fields obey the relation

$$\vec{\nabla} \times \vec{B} = \lambda \vec{B}, \tag{1.11}$$

where λ is a constant throughout the volume. Since $\vec{\nabla} \times \vec{B} = \mu_0 \vec{j}$ and the current is parallel to B in this model,

$$\lambda = \mu_0 \frac{j_{\|}}{B}, \tag{1.12}$$

effectively fixing the ratio of current to magnetic field throughout the volume. It can thus be shown through perturbation analysis that if the current is driven higher than average in a local area, the plasma will tend to even out the current distribution so that λ is constant. This is the physical prinicple behind most spheromak formation and sustainment schemes.

1.5.3 The m = 1 Source

One experiment which demonstrated an unstable, asymmetric plasma relaxing into a stable spheromak in equilibrium was the $m = 1$ source on CTX [15], shown in figure 1.3. A screw pinch plasma configuration was formed in a cylindrical chamber with an imposed axial field. Driving current between electrodes on the end walls caused the magnetic structure to become helical with the addition of the current-generated azimuthal field. The current was intentionally driven too high so that the plasma went "kink-unstable" and moved out through a hole in the cylinder to a neighboring, larger vessel and formed a spheromak.

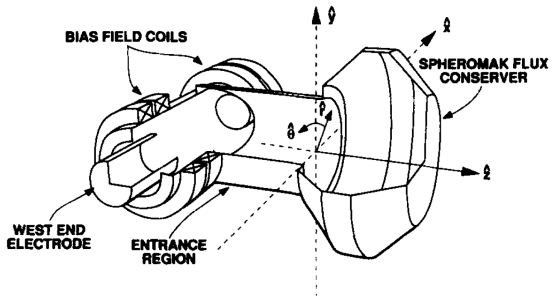

Figure 1.3: Drawing of the $m = 1$ source and spheromak flux conserver. Electrodes are oppositely biased on either end of the cylinder at left and axial field is provided by external coils. The kinked plasma travels through the tunnel in the middle of the figure into the CTX flux conserver where it relaxes. Figure from Fernandez [15].

The plasma reduced its magnetic energy to the minimum energy state through reconnection and relaxation. By expanding into a larger volume the global value of λ was reduced. In order to form a closed spheromak and achieve the minimum energy state (with minimum λ) some field lines had to be broken and reconnected. The process of relaxation while conserving helicity was observed to even out λ to its final, minimum value.

1.5.4 Coaxial Helicity Injection

The most commonly used spheromak formation method is coaxial helicity injection (CHI). CHI has been used on CTX [32], SPHEX [43], FACT [38] and SSPX [24]. This method employs "inner" and "outer" coaxial electrodes with vacuum magnetic flux spanning the gap between them (see figure 1.4). The formation sequence starts by applying a high voltage across the electrodes which creates an arc, ionizing the gas in the chamber and creating a plasma. Once current begins to flow between the electrodes, toroidal magnetic flux builds up behind the current. The $\vec{j} \times \vec{B}$ force pushes the current out into the chamber and the "bubble burst" condition is met when the $\vec{j} \times \vec{B}$ force overpowers the magnetic tension in the vacuum fields. The fields carrying oppositely directed currents near the electrodes begin to reconnect, forming closed poloidal fields and enclosing the toroidal field.

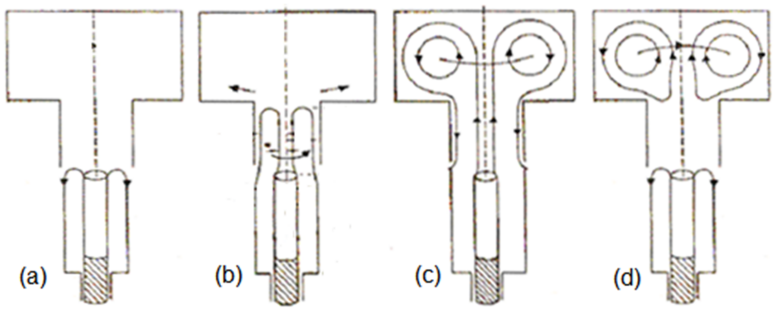

Figure 1.4: Stages of CHI formation. Vacuum flux connects the electrodes (a), the current begins to flow along the field lines (b), flux and current burst into the flux conserver (c), and the spheromak separates from the electrodes by reconnection (d).

Once the spheromak is formed, it is sustained by driving current along the outer poloidal

fields. The direct effect is to sustain the outer poloidal currents and inner toroidal field. Without reconnection and relaxation the inner toroidal current would resistively decay and the λ would become lower in the middle than the edge. Reconnection events tend to even out the λ profile, but the necessity of breaking magnetic fields and disturbing surfaces of "closed flux" has been the biggest obstacle to success for spheromaks.

The other problem with both of the above methods is that they employ electrodes and must drive a significant amount of current through the plasma-metal interface. Introducing metal impurities into the plasma is a serious concern, especially as concepts are scaled up to larger and hotter experiments. One other experiment, S-1 [28] at the Princeton Plasma Physics Laboratory, formed a spheromak using an inductive flux core. Both the toroidal and poloidal fields were induced by coils, then the plasma was detached from the core to form a spheromak in the middle. This device avoided the use of electrodes but suffered from instabilities due to the lack of a close-fitting flux conserver and the plasma could not be sustained.

HIT-SI is the first spheromak experiment designed to both form and sustain a spheromak using only induction [30].

Chapter 2

THE HIT-SI EXPERIMENT

2.1 Physical Characteristics

The HIT-SI experiment is a made up of a "bow tie" spheromak flux conserver, 1.1 m in diameter, and two semi-toroidal injector ducts on either side (see figure 2.1). The copper shell serves as a vacuum chamber and a passive flux conserver for the time scale of the experiment (a few milliseconds). The close-fitting bow tie shape was chosen [44] to stabilize the spheromak against tilt and shift instabilities and, combined with a hollow current profile, to maximize β, the ratio of plasma pressure to magnetic pressure.

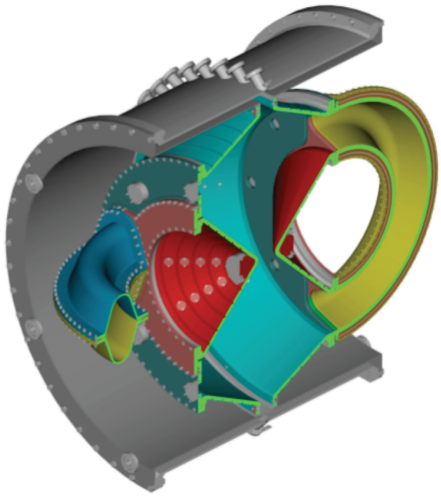

Figure 2.1: Cutaway view of HIT-SI flux conserver and vacuum chamber. The "bow tie" cross section of the flux conserver is visible, as well as the injector ducts on opposite sides of the chamber, oriented 90° to each other toroidally. Image courtesy of John Rogers and Paul Sieck.

Each injector has a "flux coil" wrapped around the duct like a solenoid which injects magnetic flux. There is also a "voltage coil" routed along the duct (and across the side

of the chamber). This is the transformer primary coil and induces a secondary current in the plasma inside the injector (in exactly the same way as the central solenoid on a tokamak). Both coil sets are illustrated in figure 2.2. There are insulating gaps along the length of the injector ducts so that the magnetic flux can penetrate quickly. The injectors are also electrically isolated from the spheromak flux conserver so the secondary current is not inadvertently driven through the spheromak flux conserver (rather than the plasma).

The upgrade to HIT-SI3 is nearly complete and replaces one injector (The "Y" injector) with three smaller injectors evenly spaced around the same annulus. The injector in the other side ("X") is retained but not used.

2.2 Inductive Helicity Injection Current Drive

HIT-SI was designed to use the same relaxation current drive principle as many other spheromaks. The magnetic fields surrounding the spheromak at the edge of the flux conserver would be driven at a high value of λ such that they would reconnect into the spheromak, redistributing the current and sustaining the configuration. To avoid introducing impurities from electrodes, the externally imposed voltage for helicity injection on HIT-SI is an induced loop voltage. Each injector voltage primary coil is oscillated sinusoidally in phase with the magnetic flux coil, giving time-dependent helicity injection. Recall that the injection term in generalized helicity is:

$$\dot{K}_{inj} = 2 \int \vec{E}_{vac} \cdot \vec{B}_{vac} \, dV. \tag{2.1}$$

The vacuum electric field along a closed path through the injector can be evaluated as the induced loop voltage, V_{loop}. The total magnetic field parallel to V_{loop} is the injected magnetic flux, ψ_{inj}. Therefore the helicity injection term for a single injector is:

$$\dot{K}_{inj} = 2 V_{loop} \psi_{inj}, \tag{2.2}$$

$$\dot{K}_{inj} = 2 V_0 \sin(\omega t) \; \psi_0 \sin(\omega t). \tag{2.3}$$

HIT-SI accomplises *steady* helicity injection by operating two injectors 90° out of phase, yielding total helicity injection:

$$\dot{K} = 2V_0\ \psi_0 \sin^2(\omega t) + 2V_0\ \psi_0 \cos^2(\omega t), \tag{2.4}$$

$$\dot{K} = 2V_0\ \psi_0[\sin^2(\omega t) + \cos^2(\omega t)], \tag{2.5}$$

$$\dot{K} = 2V_0\ \psi_0. \tag{2.6}$$

A thin layer of insulating material (Al_2O_3) coats the inside of the flux conserver and injectors preventing plasma arcs to the copper walls and ensuring that helicity injection is purely inductive.

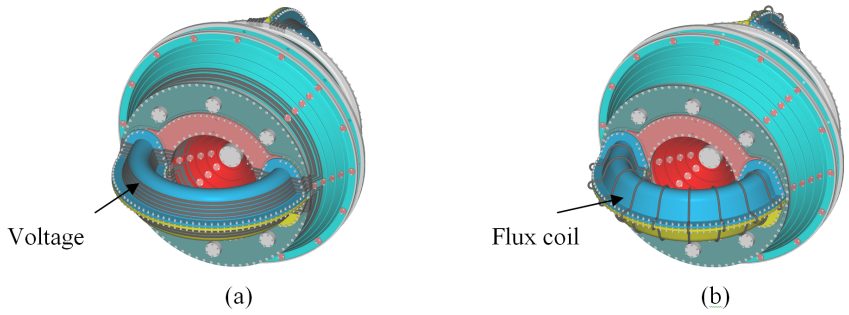

Figure 2.2: CAD drawing of HIT-SI with voltage (a) and flux (b) coils drawn separately and colored gray. Electrically isolated sections of the flux conserver are denoted by different colors. Image courtesy of John Rogers.

The pathway for spheromak formation and sustainment has been shown [56] to include three steps. First, the fields from each injector fill the volume on a direct path from one injector mouth to the other (of the same injector). Some plasma reconnects to form a separate state with $n = 1$ symmetry. The $n = 1$ state has a lower value of λ than the injectors (12 vs 15 – 20 m^{-1}) and thus is a lower energy state which the plasma tends to relax toward. The injectors are well coupled to this state because they also have $n = 1$ symmetry. Next, the $n = 1$ state undergoes a major reconnection and relaxation event, transforming into the $n = 0$ symmetric spheromak with λ of 10.3 m^{-1}. Finally, with the

spheromak magnetic fields dominant, the injector fields "wrap around" [53] the spheromak and sustain it through reconnection and relaxation.

2.3 Imposed-Dynamo Current Drive

The discovery of imposed-dynamo current drive (IDCD) [33] showed that current could be driven across closed magnetic surfaces without reconnection. This greatly improves the spheromak as a reactor concept [47] since many critics [23] believe spheromaks can not be sustained without "opening" magnetic surfaces, thus degrading particle and energy confinement.

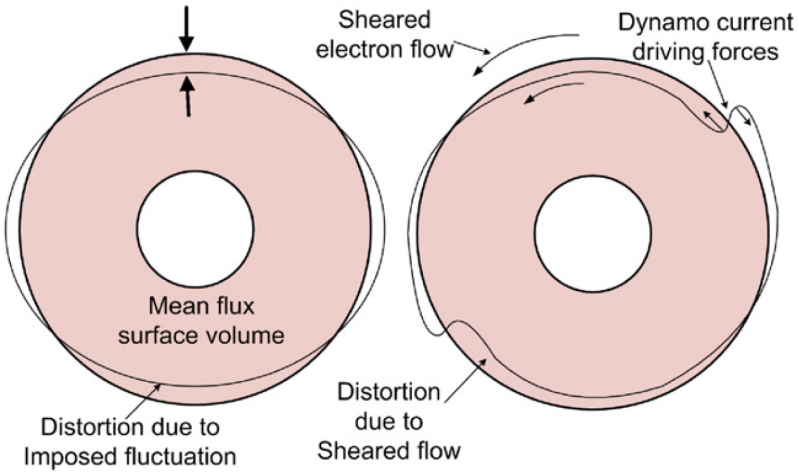

Figure 2.3: Sketch of flux surface distortion. The shaded area represents the volume inside a mean toroidal flux surface with lower electron velocity than outside. An imposed fluctuation of the surface, exaggerated for clarity, becomes distorted by the sheared electron flow. The figure is taken from reference [33].

IDCD works by distorting magnetic flux surfaces which separate zonal electron flows, as illustrated in figure 2.3. A magnetic drag force tends to equilibrate the flows, thus driving current in the region of lower electron flow and retarding electron current in the other region. This technique is implemented on HIT-SI by directly driving current on the injector fields at a higher value of λ than the spheromak. Thus, the electron flow outside the separatrix is

higher than inside. Perpendicular magnetic fluctuations are imposed by the nature of the oscillating injector fields.

The current drive can quantified by balancing the relevant fluctuations against the plasma current and its rate of change in two-fluid Ohm's law. Using mean dynamo theory, the quantities are separated into equilibrium and fluctuating components. The two-fluid, parallel, generalized Ohm's law [34] is

$$E_{\|} = -\langle \delta v_e \times \delta B \rangle_{\|} + \eta j_{\|}, \tag{2.7}$$

where v_e is the electron fluid velocity, η is the resistivity, j is the current density, and $\|$ denotes parallel to the equilibrium field B. The fluctuating current is generated primarily by electron flow rather than ion flow since the electron fluid is frozen to the magnetic field. Therefore we can make the substitution

$$\delta v_e \simeq -\frac{\delta j}{ne}, \tag{2.8}$$

where n is the electron density and e is the electron charge, and express Ohm's law as

$$-\langle \delta j \times \delta B \rangle_{\|}/ne = \eta j_{\|} - E_{\|}. \tag{2.9}$$

In order to estimate the perturbations of a flux surface required to drive current inside the surface, consider a cylinder with uniform B_z and j_z. The current driving force, $ne\eta j_{\|} - neE_{\|}$, can be integrated over the volume and the left hand side of equation (2.9) can be expressed as the Maxwell stress-induced force on the surface:

$$-\int \frac{\delta B_{\perp} \delta B_z}{\mu_0} da = \int ne(\eta j_z - E_z) dV, \tag{2.10}$$

where $\delta B_{\perp}$ is the perturbation perpendicular to the surface. The maximum current driving force occurs when the perturbations become grossly distorted and δB_z is due to the bending of the imposed $\delta B_{\perp}$, as illustrated in figure 2.3. Assuming the integrand of the perturbation

force saturates at $(\delta B_{\perp}^2)/2\mu_0$, equation (2.10) can be evaluated in toroidal geometry as

$$\frac{(\delta B_{\perp rms})^2}{2\mu_0} 2\pi R_0 2\pi r \geq (\eta j_{tor} - E_{tor}) ne\pi r^2 2\pi R_0, \tag{2.11}$$

where r and R_0 are the minor and major radii, respectively, and $\delta B_{\perp rms}$ is the fluctuating magnetic field perpendicular to the flux surface. The left and right hand sides of equation (2.11) are equal when some "slippage" occurs between the magnetic fields on either side of the flux surface. The perturbation force balances the current and its rate of change. However, if the perturbations are sufficiently large such that adjacent magnetic fields are locked together, no additional current drive is gained from larger perturbations, hence the inequality.

To gain intuition and compare to expermentally measureable parameters, equation (2.11) can be converted to depend on only the toroidal current, injector current, electron density, and resistive decay time. For uniform $j_{\parallel}$, the inductance per length is $\mu_0/4\pi$ and resistance per length is $\eta/\pi a^2$, giving a resistive decay time

$$\tau_{L/R} = \frac{\mu_0 a^2}{4\eta}, \tag{2.12}$$

evaluated at minor radius $r = a$. The electric field can be expressed as

$$E_{tor} = \frac{\mu_0}{4\pi} \dot{I}_{tor} \tag{2.13}$$

and I_{tor} is $\pi a^2 j_{tor}$. The fluctuating, perpendicular magnetic field is directly related to injector current by

$$\delta B_{\perp rms} = \frac{\mu_0 I_{inj}}{4\pi a}, \tag{2.14}$$

where I_{inj} is the quadrature injector current, $I_{inj} = \sqrt{I_{inj,x}^2 + I_{inj,y}^2}$. Equation (2.14) is Ampere's law evaluated for the injector current path predicted by equilibrium modeling.

Substituting the relevant equations and solving for dI_{tor}/dt gives the IDCD equation,

$$\dot{I}_{tor} = \frac{C_1}{8\pi a^3 ne} I_{inj}^2 - \frac{I_{tor}}{\tau_{L/R}}. \tag{2.15}$$

C_1 is a fitting parameter which corrects for approximations, but should be of order unity.

The IDCD model has been validated on HIT-SI by predicting the toroidal current evolution, j/n dependence of the injector impedance, and the internal magnetic field profile, as described in reference [33]. On HIT-SI, the magnetic perturbations are larger than required for current drive inside the separatrix, resulting in "locked" electron flow and thus flat λ inside this region. The injector-linking region has the higher λ imposed by the injectors, so the total profile predicted by IDCD is a "step" as illustrated in figure 2.4.

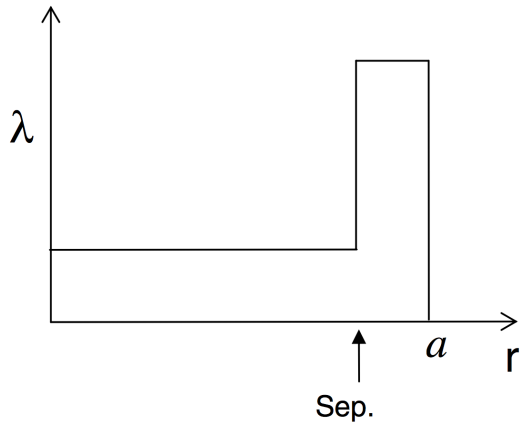

Figure 2.4: Diagram of the step λ profile predicted by IDCD. The figure is taken from reference [33].

If a magnetized plasma is stable it responds to perturbations with a restoring force and moves as a rigid object perpendicular to flux surfaces. The step profile illustrated in figure 2.4 is a time-averged model. The toroidal and poloidal paths of the injector fluxes change as a function of time [45, 55] because the AC injectors have odd symmetry. Thus, if the spheromak inside the separatrix is stable it will oscillate as a rigid object due to the temporally and spatially varying forces imposed by the injector currents and fluxes.

2.4 Diagnostics for HIT-SI

A variety of instruments are used to monitor plasma properties in HIT-SI. While this work focuses on light emission from ions, the following diagnostics have been categorized by measurements of magnetic fields, electron properties, and light emission.

2.4.1 Surface and Internal Magnetic Probes

The HIT-SI flux conserver has 96 three-axis magnetic probes embedded in the copper walls. Installation and initial calibration was done by Paul Sieck [45] and significant improvements were made by Jonathan Wrobel [56]. Each probe's nested windings are sensitive to poloidal, toroidal, and normal magnetic fields, although only the poloidal and toroidal signals are recorded. The probes are separated from the plasma by 1.9 mm thick stainless steel disks brazed into holes in the copper. The $\dot{B}$ signals are numerically integrated and frequency compensated to calculate the surface $\vec{B}$ field within a frequency range of 10 Hz to 200 kHz.

The probes are arranged in four poloidal arrays and two toroidal arrays, as shown in figure 2.5. The 16 poloidal magnetic field signals from each poloidal array are used to calculate the local toroidal current. The best available toroidal current measurement on HIT-SI is the average of the currents at the four locations. Two toroidal arrays of 16 probes each are located on either side of the mid plane diagnostic gap, spaced 22.5° apart. The primary function of these probes is to capture toroidal Fourier modes, up to $n = 7$.

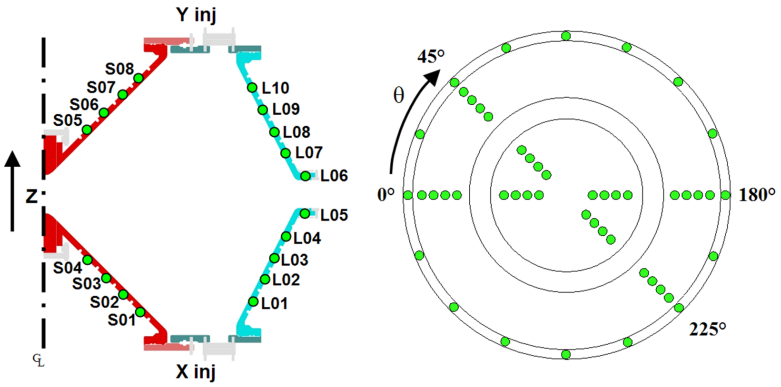

Figure 2.5: Poloidal cross section of HIT-SI (left) showing probe locations (green) of a poloidal array. Toroidal probe locations are shown at right. Note the four poloidal arrays at 0, 45, 180, and 225 degrees, and the 16 probe array along the outer circumference. Figure from Wrobel [56].

An internal magnetic probe is used to measure local magnetic fields from the diagnostic gap (major radius $\sim$ 55 cm) to approximately the magnetic axis (major radius 33.1 cm). The probe is inserted at toroidal angle 225°, toroidally equidistant from X and Y injector mouths. The 3D magnetic coils are spaced 1.27 cm apart inside stainless steel tubes which are insulated from the plasma by boron-nitride sheaths. The probe has three parallel stems which enables measurements of $\vec{\nabla} \times \vec{B}$ at discrete radii.

In order to measure local magnetic fields near the geometric axis of HIT-SI, a new, longer, 3D magnetic probe has been constructed. The probe is fully retractable and extends along a diameter in the mid plane through the geometric axis to the far-side magnetic axis. This probe was installed during the extended maintenance period when HIT-SI was upgraded to HIT-SI3.

2.4.2 FIR Interferometer, Langmuir Probe, and Thomson Scattering

Electron density on HIT-SI is measured with a Martin-Puplett far infrared interferometer [41, 52]. Line-integrated density is calculated from a measurement of the change in index of refraction of the plasma. The beam path is along a chord through the mid plane at an

impact parameter of 35.1 cm, just outside the magnetic axis.

The local electron temperature is measured with triple Langmuir probe [41], designed by Roger Smith and Griff O'Neill and maintained by Derek Sutherland. The probe is installed in the midplane and can be radially inserted to a depth 15 cm from the wall. Unfortunately, data from this probe are not considered trustworthy.

To simultaneously measure the electron temperature at multiple points in a non-perturbative way, a Thomson scattering [27] system has been installed on HIT-SI3. A ruby laser is housed in a separate room, the beam from which is focused to a region inside the HIT-SI3 flux conserver. Collection optics are able to view scattered emission from seven different points, but only four simultaneously. The system comprises many parts used by Will Hamp on the HIT-II experiment [18] and is being rebuilt by Kyle Morgan and Chris Everson.

2.4.3 SPRED, VUV Spectrometers, PMTs, and High Speed Camera

Impurity line identification and qualitative emission levels are measured with the "spectrometer, poor resolution, extended domain" (SPRED) diagnostic. Light is collected by two collimating lenses from the mid plane of HIT-SI and transmitted by fiber optics to the spectrometers. One spectrometer has a range from 200 to 350 nm and the other from 390 to 580 nm. The detectors integrate the signals over the entire shot, giving no time resolution.

Two vacuum ultraviolet (VUV) spectrometers are connected to the HIT-SI vacuum chamber with symmetric chordal views of the plasma. The spectrometers are evacuated and open to the main vacuum chamber during a shot because UV radiation does not penetrate glass or air. Typically, one spectrometer is tuned to detect C III emission at 117.6 nm and the other C IV emission at 154.8 nm. The relative abundance of each charge state gives an indication of electron temperature based on the extent to which carbon is ionized or "burned-through".

Time-resolved impurity emission in the visible spectrum is accomplished with photomultiplier tubes (PMTs) and bandpass filters. Typical emission lines recorded are He I (450 nm), H_α (656 nm), C II (280 nm) and C III (230 nm). The light for each signal is collected by a collimating lens and transmitted to the filter and PMT by fiber optic cable.

A high speed camera is used in conjunction with coherent fiber optic bundles to record movies of the HIT-SI plasma. Two 128 by 128 element fiber optic bundles can be used simultaneously, giving either two separate views of the plasma or a stereoscopic view if the bundles are positioned a few inches apart. The typical frame rate is 150,000 images per second and exposures are $1 - 2$ μs.

Chapter 3

PLASMA DENSITY REDUCTION AND CONTROL

3.1 Helicon Preionization Source

3.1.1 Introduction

One of the biggest obstacles to increasing plasma current in HIT-SI has been the building up of neutral density in the spheromak flux conserver. It is well established that optimal spheromak plasma performance requires a ratio of plasma current to electron density just above the Greenwald limit [17] of 1×10^{-14} Am (N.B., the Greenwald limit is typically a maximum density limit for a given current). A breakthrough [55] in HIT-SI performance was achieved by conditioning the hygroscopic Al_2O_3 walls with helium discharges so they would subsequently pump deuterium for 2 to 4 discharges before saturating. This technique improved performance, but the electron density and line emission would still rise steadily and exceed the Greenwald limit within 2 ms.

Breakdown on HIT-SI is achieved by applying sufficient loop voltage, magnetic flux, and neutral density through the injectors to meet the Paschen condition. Typically the loop voltage must exceed 400 V and magnetic field in an injector must be near 40 mT. The neutral density is not well known, but for helium discharges the electron density exhibits a spike up to $3 - 6 \times 10^{19}$ m^{-3} within 100 μs of breakdown, then falls to $2 - 3 \times 10^{19}$ m^{-3}. Therefore the neutral density in the confinement volume is sufficiently high at breakdown to produce the initial spike in plasma density. The injectors must be continuously fueled in the middle of the ducts and the confinement volume does not have significant vacuum pumping on a $\sim$2 ms time scale, so the neutral density builds up above optimal levels. Helium discharges have always performed worse than deuterium and never achieved $|j/n|$ above 1×10^{-14} Am. Plasma performance has been improved by using a helicon [9] based preionization source to initiate breakdown, thus reducing and delaying neutral gas fueling.

3.1.2 The Electrodeless Plasma Source

Design and Operation

Introducing high-Z impurities is a major concern for HIT-SI because at low temperatures (< 50 eV) line radiation can result in significant power loss. Previous experience [22] with plasma guns made by the SSI company resulted in significant injection of impurities. The primary benefit of a helicon based plasma source is the inductive, virtually impurity-free nature of the device. Previous attempts to achieve plasma breakdown using external, inductive antennas were unsuccessful because they did not project plasma far enough into the experiment chamber. The system developed by Eagle Harbor Technologies, Inc. consists of external coils to create an axial magnetic field, a helicon antenna wrapped around a quartz tube, puff gas feed, a small grid which arcs for only 25 μs to initiate the discharge (see figure 3.1), and associated power supplies.

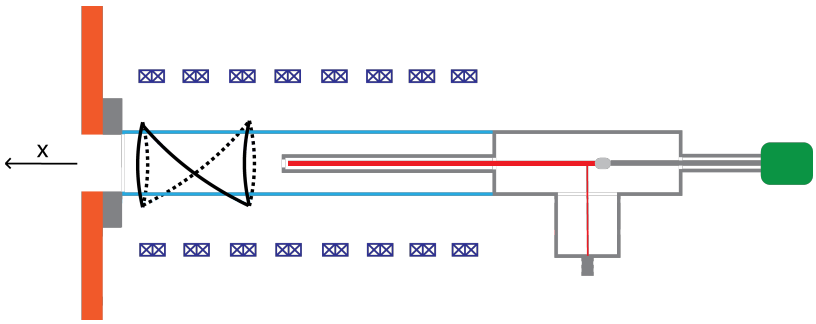

Figure 3.1: Sketch of the EPS. The gas puff valve (green) releases gas into the tube along the axis. A high voltage feed-through (bottom of "T") is used to briefly bias the section of gas tubing colored red, generating a small arc at the outlet of the two coaxial tubes (left). The helicon antenna (black) is wrapped around the quartz tube (cyan) and the axial field coils (blue) are mounted farther out. The HIT-SI flux conserver section is represented by the copper colored blocks on the far left of the figure.

The antenna is a left-handed Nagoya Type III half wave helical geometry with 4 cm diameter and 8 cm length. The antenna is the inductor in a series resonant circuit and is driven at the resonant frequency of 1 MHz. Current in the antenna is measured with a Pearson current transformer and rings up to 1,800 A_{pp} in vacuum, decreasing to 1,200 A_{pp}

when loaded by the plasma. The output of the switching power supply is driven at 500 V_{pp}. The resistance of the entire circuit is 278 mΩ and a typical plasma load can be approximated as an additional resistance of 140 mΩ, yielding 25 kW of power delivered to the plasma.

For typical operations the axial field coils are ramped up to a current of 800 Amps over 200 μs, giving an approximate field strength of 38 mT. Then the starter-arc is energized for 25 μs and the antenna is driven for 200 μs. By driving at a lower frequency than the standard 13.56 MHz, we are able to use switching power supplies with lower cost and higher pulsed-power than conventional RF power supplies. The axial field does not significantly penetrate the 1.27 cm thick chromium copper HIT-SI flux conserver on the time scale of an EPS discharge. Keeping the neutral particle density low is a priority for HIT-SI, so discharges are initiated without pre-fill gas, unlike most helicons. The experiment base pressure is in the range of 10^{-8} Torr. Neutral gas with a feed pressure of 1,000 – 2,000 Torr is released by the puff valve approximately 3.6 ms before the discharge initiates. The time is found empirically to be the minimum required to build up sufficient neutral density for plasma breakdown. The neutral density in the EPS is not known.

Plasma Density Characterization

Before installation on HIT-SI, testing was performed on a large vacuum chamber using hydrogen. With gas supplied at 30 PSIG, the throughput of the Parker Series 9 puff valve was measured to be 6.7×10^{18} particles per ms in steady state. A double Langmuir probe in ion saturation was used to measure the axial plasma density profile downstream of the EPS (the "x" direction as denoted in figure 3.1). The electron temperature was estimated to be 20 eV, based on measurements of similar helicon devices [58]. The density was observed to decrease by an order of magnitude in $\sim$17 cm (see figure 3.2).

Impurity Injection Testing

Since impurity control is a primary concern for HIT-SI, two sets of witness plate tests were performed. In the first test an aluminum plate was imaged with a scanning electron microscope (SEM) before and after firing 300 hydrogen pulses directly at the plate. Selected

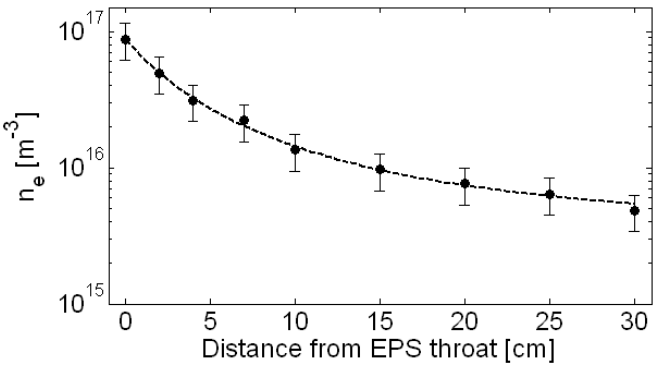

Figure 3.2: Electron density of hydrogen plasma as measured by a Langmuir probe. The plotted least squares fit to a $1/r^2$ function agrees well with the data. Axial magnetic field for this test was 52.5 mT (higher than is used on HIT-SI).

regions of the plate were also scanned with energy dispersive X-ray spectroscopy (EDS) and revealed no significant change or impurity capture. A 1 μm size particle was noticed on the witness plate and EDS analysis suggests that it is quartz (see figure 3.3). A second test was performed with a graphite witness plate since there is no known source of carbon in the EPS device. X-ray photoelectron analysis (XPS) scans were performed before and after firing 500 hydrogen pulses at the plate. The second scan revealed increased oxygen content, most likely due to transferring the sample in air, but no indication of materials present in the helicon source such as metals or quartz.

3.1.3 Density Reduction in HIT-SI Discharges

An initial test of the EPS on HIT-SI explored the lower limit of density with helium discharges. We found that a spheromak could be formed with $n_e \approx 6 \times 10^{18}$ m^{-3}, $I_{TOR} \approx -15$ kA, and $|j/n|$ at or above the Greenwald limit (see figure 3.4). The large variations of density and current early in the pulse are due to the helicity injectors being under-fueled as gas begins to flow into them. In order to inject more power and increase the toroidal current earlier, a typical pulse has $n_e \approx 1 \times 10^{19}$ m^{-3} and $|j/n|$ near the Greenwald limit. A helium discharge without the EPS is also plotted for comparison— note the large, initial

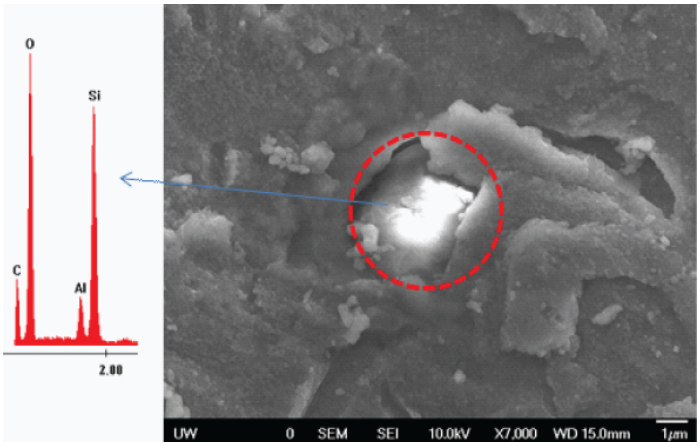

Figure 3.3: SEM image of aluminum witness plate (right) showing ∼1 μm size particle. EDS analysis (left) suggests the particle is quartz.

density spike and oscillations.

The density reduction in deuterium discharges is not as pronounced as for helium, and the initial density spike could not be eliminated. A set of comparison shots are presented in figure 3.5. In the low density limit, pulse 128155, n_e is initially steady at $\sim 2 \times 10^{18}$ m^{-3} before slowly rising into the 10^{19} m^{-3} range. j/n is too high to increase toroidal current during the first half of the pulse; this value must fall below $\sim$ 2.5 $\times$ 10^{-14} Am before the current increases. A typical deuterium discharge, 128163, has slightly higher density ($\sim$ 5 $\times$ 10^{18} m^{-3}) but achieves toroidal current similar to that of a pre-EPS comparison pulse (127491).

Further optimization and gas fueling control is needed to improve toroidal current performance in deuterium discharges since the electron density still increases and j/n falls below the Greenwald limit. The EPS has reduced the initial density spike, overall density levels, and j/n throughout the discharge.

3.1.4 Enabling Single-Injector Discharges

The EPS has enabled the discovery that each helicity injectors preferred direction of toroidal current is not robust at driving frequencies higher than 5.8 kHz. A previous study [14] found

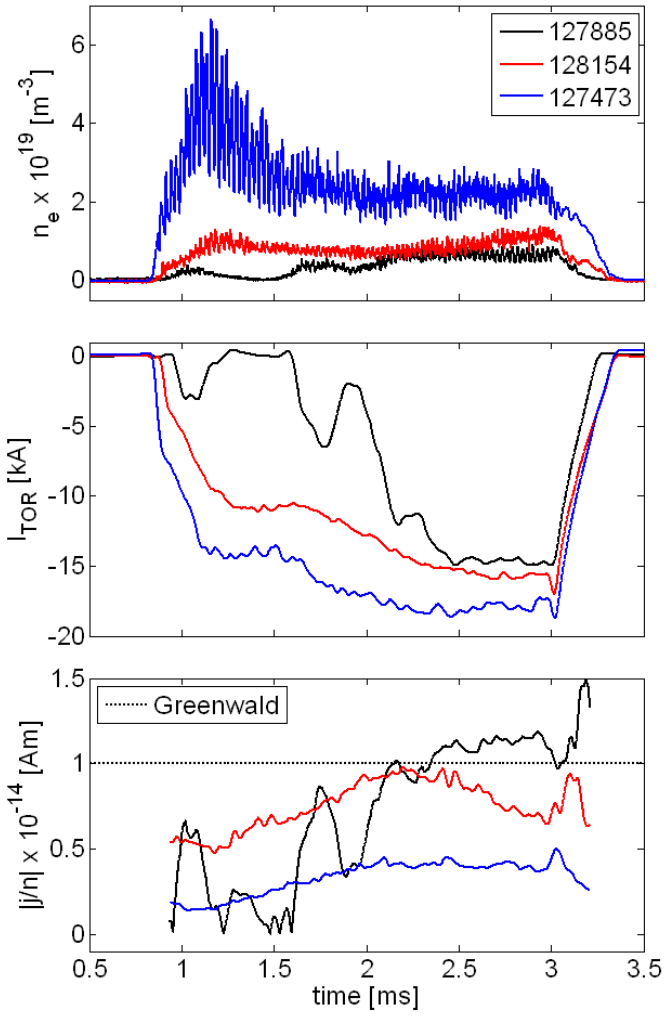

Figure 3.4: Density, I_{TOR}, and $|j/n|$ plotted for three HIT-SI discharges. Pulse 127885 represents the low density limit and exhibits severe variations in density and current during the first half of the pulse, but later achieves the best absolute value of j/n. Pulse 128154 represents a typical helium discharge initiated with the EPS and 127473 is a comparison from before the EPS was installed. Note the large initial density spike and much lower absolute value of j/n. The HIT-SI injector-driving frequency is 36.8 kHz.

that while both injectors work synergistically to drive a spheromak with either positive or negative toroidal current, driven individually they always form and sustain spheromaks with opposite toroidal current. The HIT-SI injector frequency has since been increased to 14.5,

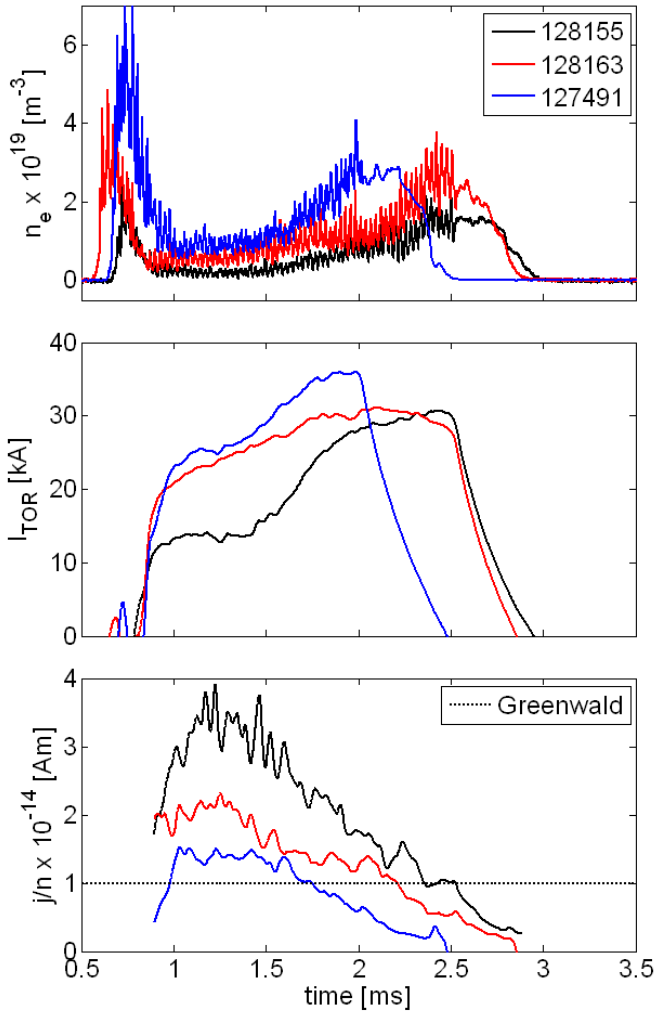

Figure 3.5: Density, ITOR, and j/n plotted for three deuterium discharges. Pulse 128155 represents the lowest density spheromak achieved, 128163 is a typical deuterium pulse, and 127491 was taken before the EPS was installed. The initial density spike could not be eliminated, but it was reduced. With the EPS, discharges are possible with j/n above and below optimal values.

36.8, 53.5, and 68.5 kHz. Previous attempts to achieve breakdown using only one helicity injector at 14.5 kHz failed (although an experiment was successful by injecting far more gas than desirable for spheromak sustainment). All attempts to achieve breakdown with one injector at 36.8 kHz failed without the use of the EPS.

Using the EPS to initiate discharges, each injector has been operated individually at all of the frequencies mentioned above greater than 5.8 kHz. The toroidal current direction at these frequencies is either random or follows a trend which is not robust. The physical mechanisms involved are not well understood.

3.1.5 Discussion

One explanation for why the EPS significantly reduces the breakdown density is that the initial plasma aligns the electric and magnetic fields. The Paschen breakdown density can be lowered by increasing the length of the breakdown path. The conducting flux conserver forces the injector magnetic flux to be parallel to the walls and close within the volume. Because of the insulating coating on the flux conserver, the induced electric field in vacuum is mostly perpendicular to the walls and poorly aligned with the closed magnetic field. By introducing plasma into the volume, charged particles follow the electric field to the insulating walls and effectively cancel any $E_\perp$. The electric and magnetic fields are then much better aligned since they have the same boundary conditions, $E_\perp \approx B_\perp \approx 0$.

The results of this section have been published in Review of Scientific Instruments [25].

3.2 Piezoelectric Gas Puff Valves

3.2.1 Introduction

The gas fueling system on HIT-SI was composed of pneumatically actuated valves and a constricting throttle [52]. The gas flow profile in time for this system was difficult to change, and no part of the system was adjustable during a plasma pulse. Achieving optimal fueling of HIT-SI was a difficult trade-off because the injectors need a steady gas supply to function efficiently, but the spheromak lifetime is limited by a buildup of neutral particles in the main flux conserver.

We were unable to find a commercially available valve which offered several hundred Torr liter per second flow rates with sub-millisecond throttle control. A common valve in the plasma physics community is the Series 9 pulse valve by Parker. This valve opens in approximately 200 μs but takes over a millisecond to close and is not adjustable. The

Proportional Piezo Valve by Cedrat Technologies does not have a high enough flow rate and the Fast Gas Injection System from General Atomics does not offer fast enough time response. Bates and Burrell [5] were able to modify a Veeco PV-10 valve to achieve the desired flow rate and response time, but this valve is no longer manufactured.

3.2.2 Valve Design and Control

In conjection with the upgrade from HIT-SI to HIT-SI3, new valves were developed which use a piezoelectric crystal to actuate a standard ultra-high vacuum (UHV) valve. The assembly (see figure 3.6) consists of an aluminum housing with the bottom portion tapped for affixing to a Swagelok SS-4BK-V51-1C UHV needle valve. The piezoelectric disk, model P-286.23 from Physik Instrumente, is mounted in the upper portion and attaches securely to the valve stem. From below, a specially chamfered washer is raised with shims to rest against the bottom of the disk center. From above, another washer is secured with two thin nuts. The disk is only capable of 20 (10) Newtons of push (pull) force, so a spring is carefully selected to balance the gas pressure on the bottom of the needle valve and the valve's bellows such that the valve is normally barely closed. The spring used is part number LC 046FG 04M by Lee Spring Company. Notice the spring pushes on a special "stove pipe hat" shaped aluminum piece which transfers the force to the top of the valve stem. The spring is compressed and held in place by a washer and three #6 screws in the middle of the housing.

Voltage regulation is achieved by varying the duty cycle of a high voltage power supply made by Eagle Harbor Technologies, Inc. Each power supply uses a fast ($\sim$40 ns) IGBT switch and can output voltages from 0 to $-1,000$ V, adjusted with a potentiometer. The piezoelectric crystal and a parallel resistor act as a low-pass filter for the pulse-width modulated (PWM) high voltage.

3.2.3 Characterizing the Prototype Valve

Tests were conducted which raised, lowered, and turned off the voltage to assess the valve response times (see figure 3.7). 80 PSIG of helium was supplied to the valve and a small,

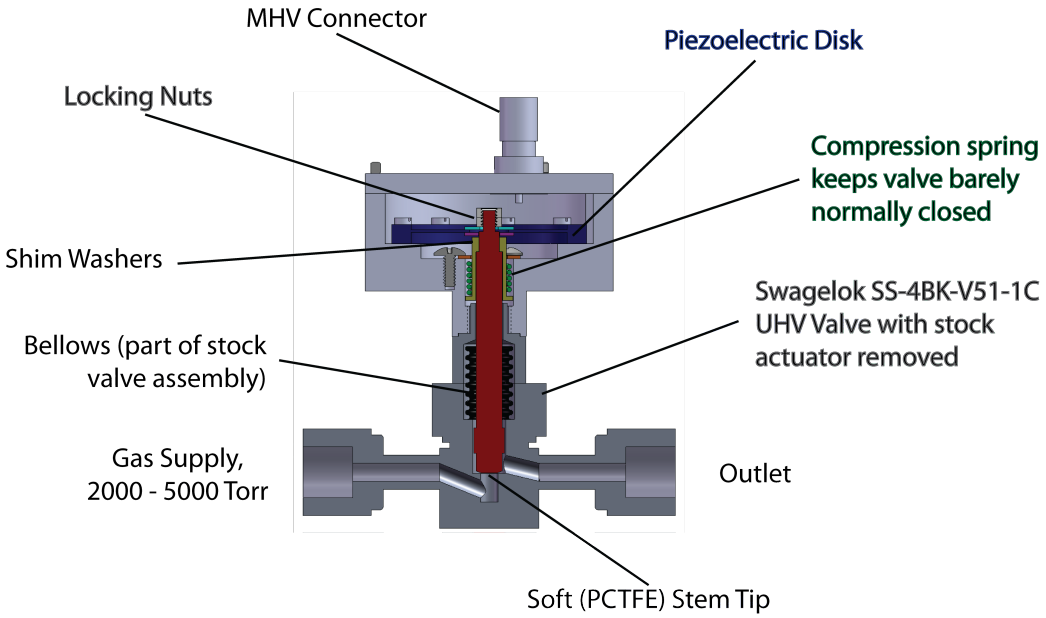

Figure 3.6: Cutaway CAD rendering of the valve assembly with pertinent details labeled.

closed volume downstream was filled (initially at atmospheric pressure). In general, the valve achieves the desired flow rate approximately 0.5 ms after the voltage begins to change. So as not to damage the crystal, large changes in voltage are not applied in under 200 μs. One observation is that the crystal has a natural resonance near 2.5 kHz which is excited by sudden changes in voltage. The effect is more pronounced when lowering the voltage than raising it. For example, tests were done where the voltage transition near 2 ms was much sharper— similar to the initial voltage turn-on. This caused the valve to fully close and reopen (i.e.: bounce) before coming to equilibrium at the desired flow rate. Thus the natural RC decay to a lower voltage serves to lower the valve stem gently and reduce over-shoot.

The flow rates were determined by averaging the applied voltages and finding the slopes of linear fits to pressure during the times from 0.6 to 2.6 ms and 3.1 to 4.2 ms. The results are displayed in figure 3.8 and show flows from 10 to over 325 Torr liters per second by varying the voltage from 250 to 665 volts. As expected, the trend is roughly linear because the crystal is opposing the bellows and counter-spring. The pressure on the tip of the valve stem is constant.

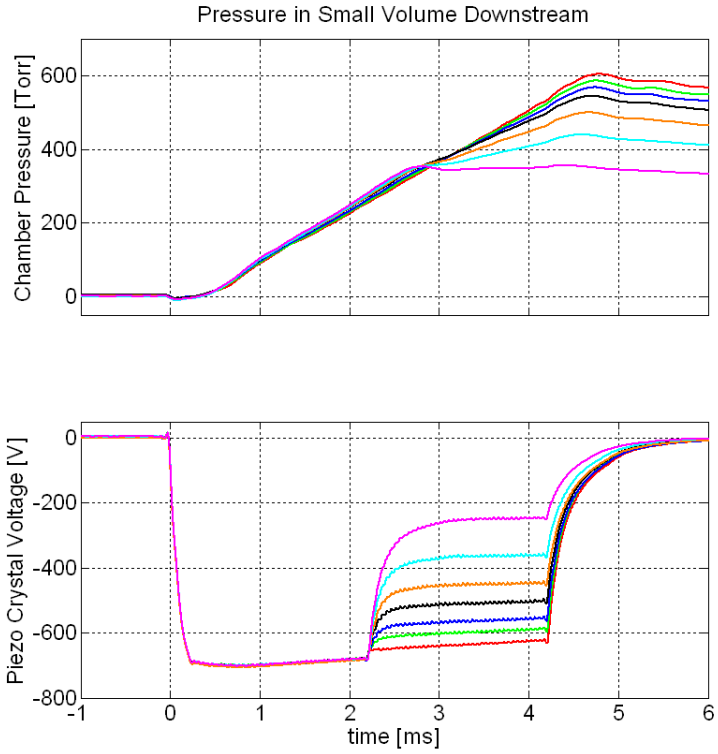

Figure 3.7: Pressure in a 2.26 cm^3 volume downstream of the valve vs. time (upper) and corresponding voltage applied to the piezoelectric crystal (lower). Note that when the applied voltage is lowered to $\sim$ 220 V the pressure stops increasing and therefore the valve is closed.

3.2.4 Implementation on HIT-SI3

For installation on HIT-SI3, three features are needed downstream of the valve: 1) a fast, piezoelectric pressure transducer is used to measure gas injection, 2) the valve must be electrically isolated from the injector copper because the valve body is part of the sensitive piezoelectric crystal circuit, and 3) a special adapter is needed to mount on the injector. In order to maximize the conductance from the valve to the injector the path should be as short as possible. A special valve base was designed to satisfy the above requirements with a single, welded assembly, thus eliminating VCR fittings which add length (see figure 3.9).

Assembling and testing the three new valves proved more challenging than the prototype

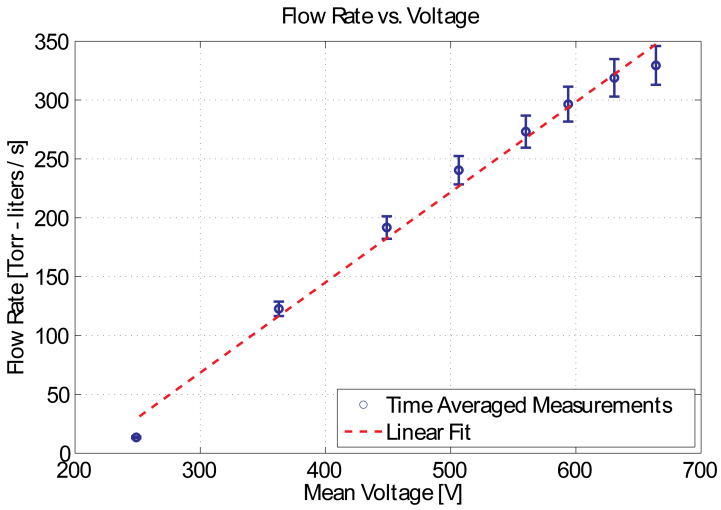

Figure 3.8: Flow rate vs. voltage applied, calculated by fitting to linear regions of pressure and voltage traces. An uncertainty of 5% is estimated based on variation in flow rate for different pulses during the first linear time period.

Figure 3.9: Valve base assembly. The opening at far left holds a Viton o-ring on the shoulder and secures to the injector with two screws. The ceramic insulator is to its right, followed by a socket for a PCB Piezotronics S112A21 pressure sensor. To the right of that is a Swagelok valve base with the downstream VCR fitting cut off.

because initially 2 of the 3 leaked with no apparent cause. After much testing we found that in order to maintain an acceptable seal against up to 80 PSIG, the valve assembly must stay closed without the piezoelectric crystal attached, i.e.: the spring holds the valve stem closed, not the crystal. Since the force of the valve stem on the seat is so small, the tip must be very well aligned. After testing several valves we concluded that this requirement is stricter than the manufacturing tolerances of the Swagelok valves because some valves

seal and some do not. Once we identified two more UHV valves which sealed, we converted their bases to the custom design and proceeded with testing.

As part of a conservative design strategy, pneumatically operated Swagelok valves are installed just upstream of the piezoelectric valve assemblies. These valves are reliably leak tight and prevent excess gas from flooding the HIT-SI3 chamber if the piezoelectric valves leak too much or fail. Once the valves were installed on HIT-SI3 the lines were filled to approximately 4000 Torr and the backup valves were held open. This raised the vessel base pressure by approximately an order of magnitude, from 10^{-8} to 10^{-7} Torr. This is not ideal, especially for injecting deuterium, so for normal operations the backup valves are opened for only 1 – 3 seconds before a discharge.

Fast Ion Gauge (FIG) Measurements

A FIG is installed on the HIT-SI3 vacuum vessel outside the midplane ring. In order to test the flow rate of the piezoelectric valves they were pulsed individually into the volume. The raw data from these tests are shown in appendix A.1. Even though the valves were only pulsed for 5.5 or 6 ms (depending on the test) it takes ∼40 ms for the gas in the confinement volume to equilibrate with the outer vacuum tank. This is because the gas, which is in the molecular regime, can only leave through small holes in the diagnostic ring. The slow pressure decay due to pumping is noticeable later in time. To estimate the amount of gas injected, the FIG pressures at $t = 40$ ms for all three valves at various line pressures are plotted on the left of figure 3.10. The pressures are converted to flow rate by the following formula:

$$Flow(Amps) = 10\frac{PV}{t} \tag{3.1}$$

where P is the pressure (Torr), V is the volume (~ 570 liters), and t is the pulse duration (seconds). The flow rates are shown on the right of figure 3.10.

During assembly the shims and springs in the valves were optimized for maximum flow rate, but the mechanical adjustments are coarse, so the flow rates and responses to pressure could not be made identical. Valve C, for example, has a relatively high flow rate at low line

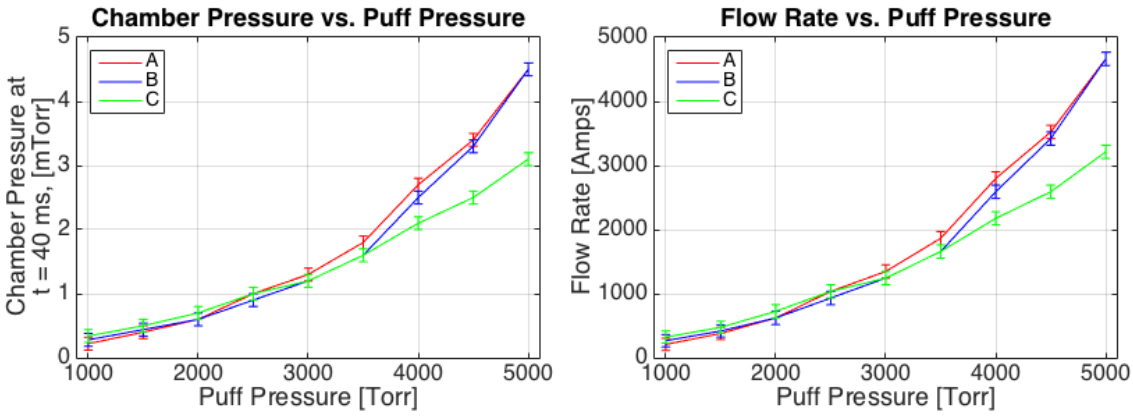

Figure 3.10: Equilibrium chamber pressure (left) converted to flow rate (right). 1 Torr liter per second equals 10 Amps of gas.

pressure and low flow rate at high line pressure. The flow rates can be changed with the charge voltage of the power supplies by adjusting a potentiometer through the front panel. The easiest way to adjust flow rates, even during a shot, is with pulse width modulation (PWM) of the power supplies.

The valves worked successfully during the HIT-SI3 run campaign of 2014 when the experiment was operated at 14.5 and 47.5 kHz. Unfortunately the valve on injector B leaked badly after the December-January bake and could not be repaired. In the interim, a pneumatic valve with a constricting needle upstream was installed in February, 2015. Due to the difficult construction and adjustment inherent to the piezoelectric valve design all valves will be replaced with Parker Series 9 valves.

Chapter 4

ION DOPPLER SPECTROSCOPY

4.1 Theory of Doppler Spectroscopy

Plasmas emit a variety of electromagnetic radiation including Bremsstrahlung from electron-ion Coulomb collisions, recombination radiation from electron capture, and line emission from bound electrons. The latter is of particular interest to laboratory plasma experiments like HIT-SI because the discrete nature of bound electron transitions can be used to glean information about the emitting atom or ion. When a bound electron transitions from a state of higher to lower energy it emits a photon with energy equal to the difference between the states, $h\nu_{ij} = E_i - E_j$, conserving energy and angular momentum. The rate of spontaneous decay is governed by the quantum mechanical lifetime of the excited state, given by the Einstein coefficient A_{ij}. An order of magnitude estimate [27] of the time to decay is $\approx 10Z^{-4}$ ns, considerably faster than most times of interest in a plasma (Z is the ion charge number). In principle, the emitted photon could be absorbed by another atom or ion to induce the reverse transition, but this process is insignificant in small, laboratory plasmas. Since these plasmas are optically thin, radiation is assumed to escape. Electrons most often transition to excited states through electron impact ionization, the probability of which can be calculated as a cross section $\sigma_{ij}(v)$. For a Maxwellian distribution of electron velocities, the rate coefficient, $\langle \sigma_{ij} v \rangle$, can be tabulated as a function of temperature. When the above processes are balanced, the plasma is said to be in *coronal equilibrium.*

Emitted line radiation is not perfectly monoenergetic— some natural line broadening occurs due to electromagnetic fluctuations which slightly shift the energies of quantum states. This effect can be neglected for emission in the visible spectrum and for densities in HIT-SI we may assume that line radiation has a precise, known wavelength in the rest frame of the atom or ion. Thus, the velocity of a light-emitting ion relative to a stationary

observer can be easily calculated from the Doppler shift,

$$v = c\frac{\Delta\lambda}{\lambda}. \tag{4.1}$$

where c is the speed of light. When observing a macroscopic plasma, the mean velocity shift of the observed light is called the *ion* or *flow velocity*.

In collisional plasmas like HIT-SI, ion velocities are assumed to equilibrate locally into a Maxwellian distribution, which in one dimension can be written as

$$f(v_i) = \sqrt{\frac{m_i}{2\pi k_B T}} \exp\left[\frac{-m_i v_i^2}{2k_B T}\right], \tag{4.2}$$

where k_B is Boltzmann's constant, m_i is the ion mass, and T is the temperature. Converting velocity to wavelength shift and solving for temperature in units of electron-volts, one arrives at the expression

$$T_i(eV) = \frac{\sigma^2 c^2 m_i}{\lambda_0^2 k_B}, \tag{4.3}$$

where λ_0 is the wavelength of light observed and σ is the standard deviation of the Gaussian wavelength distribution. Doppler temperature broadening produces a Gaussian line shape.

In high density plasmas the microelectric fields of nearby particles can produce an effect known as *Stark broadening*. This effect produces a Lorentzian line shape but is expected to be negligible in HIT-SI.

4.2 The IDS Instrument

The ion Doppler spectrometer (IDS) is a model MC-100 one meter spectrometer on loan from Professor Nagata of the University of Hyogo in Japan. Some specifications of the instrument are summarized in table 4.1.

In its original configuration, a 2.54 cm diameter collimating lens focused plasma light onto an incoherent fiber optic bundle. At the spectrometer entrance slit the fibers were arranged in a line along the slit. At the output of the spectrometer an exit slit and column lens broadened a narrow wavelength range. The detector was a 16 channel, multi-anode,

Wavelength Range	250 - 700 nm
Collimator Focal Length	1000 mm
Focal Ratio	f/8.5
Groove Density	1,800 per mm
Grating Blazed Wavelength	250 nm
Entrance Slit Width	80 μm, adjustable
Exterior Dimensions	109 $\times$ 53 $\times$ 28 cm, 80 kg

Table 4.1: IDS specifications.

photomultiplier tube (MAPMT) digitized at 1 MHz.

4.2.1 Diagnostic Upgrade

In order to allow simultaneous data collection from many chords, new optics, fiber optics, and a different detector were integrated with the spectrometer. The fiber bundle consists of a 1 cm diameter ferrule at the spectrometer containing 72 fiber optics arranged linearly. Two bundles leave the ferrule, each with 36 fibers, and terminate slightly less than 3 meters away in 7 mm diameter ferrules. Plasma light is imaged onto each linear array of 36 fibers by a wide-angle "Micro Video Lens" from Edmund Optics, which has a focal length of 2.2 mm. Each fiber maps to a cone shaped volume from which light is accepted. The cones are separated by 2.95° and at a distance of ∼110 cm, the approximate diameter of HIT-SI, the cones are 5.1 cm in diameter.

The small size of the lens-fiber assemblies (<2 cm diameter) allows the lenses to be placed on any 2.5 cm window on HIT-SI, including inside reentrant ports. Two different orientations are used for light collection in the toroidal mid plane of HIT-SI. One utilizes an existing reentrant port at 71° toroidal, as shown in the left image of figure 4.1. Due to obstructions, only 20 of 36 chords receive light, giving data out to an impact parameter (from the geometric axis) of ∼27 cm. The most important data should come from near the magnetic axis of HIT-SI, which is located at a major radius of 34 cm. An additional reentrant port was installed aiming nearly tangent to the magnetic axis as shown in the right image in figure 4.1, yielding observations out to ∼42 cm. Observations were also made

from this port with the fiber optic assembly rotated 90°, thus spanning the z direction at a major radius of ∼29 cm.

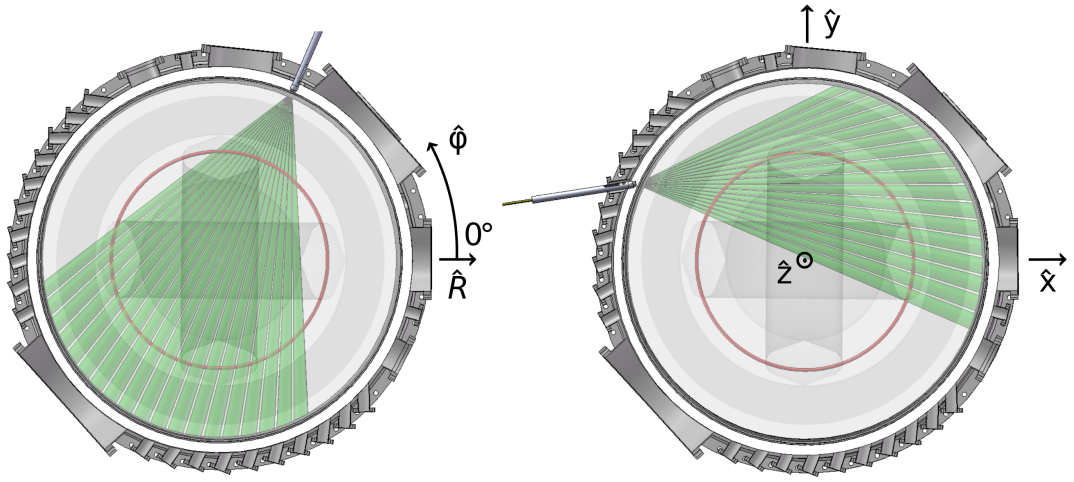

Figure 4.1: Viewing chords in the HIT-SI toroidal mid plane using an existing reentrant port (left) and a new, custom port (right). A semi transparent rendering of the flux conserver and injectors is superimposed for reference. The X injector is oriented horizontally "behind" the images and the Y injector is vertical "in front of" the images. The magnetic axis is represented by a 1 cm diameter red tube. HIT-SI's polar coordinate system is shown on the left and the equivalent Cartesian coordinates on the right. For the left configuration, "positive" impact parameters are arbitrarily defined as passing above the geometric axis.

Observations spanning poloidal sections of HIT-SI have been made from four locations so far. One of the lenses has been placed at 45° toroidal on an axial port looking in the z direction at the opposite wall, as shown in the left image of figure 4.2. The same lens has also been placed at 270° so that some of the chords look into the opposite helicity injector mouth. Both fiber optic bundles have also been placed opposite each other at 135°, as shown in the right image of figure 4.2.

The exit slit and MAPMT detector were replaced with a Phantom v710 high speed camera. Since the horizontal and vertical focal planes of the spectrometer do not focus at the same axial location, a large cylindrical lens was used to achieve this. For HIT-SI operations at 36.8, 53.5, and 68.5 kHz, a micro channel plate (MCP) based image intensifier by Stanford Computer Optics was placed in front of the camera to amplify the relatively

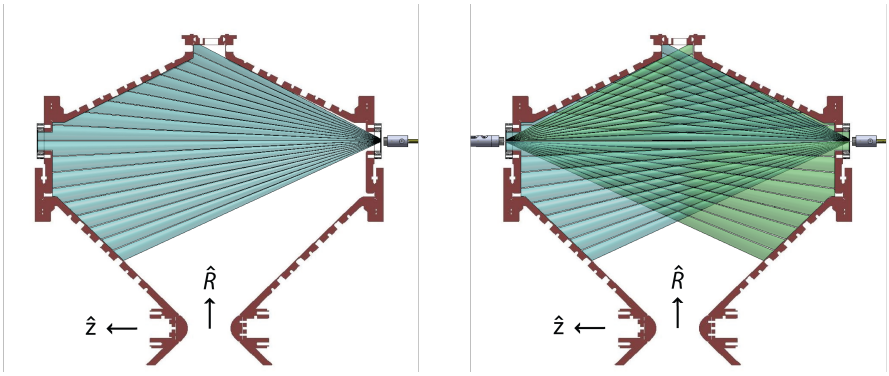

Figure 4.2: Chords in the poloidal plane of HIT-SI. A single view is depicted on the left, as used at 45° and 270° toroidal. Opposing views at 135° are depicted at right.

weak signals. The intensifier tended to "saturate" easily and had a recharge time of order 100 μs, so exposures of $3-7$ μs could only be captured every few injector cycles. For HIT-SI operations at 14.5 kHz with toroidal currents exceeding 80 kA, there was enough emission to remove the image intensifier, yielding 10 exposures per injector cycle at nearly 100% duty cycle.

4.3 Calibration Procedures

The dispersion (wavelength separation of camera pixels), relative intensity of each fiber, and instrument broadening must all be measured empirically. The relative intensity and instrument broadening are found by removing the wide angle lens and placing a spectral source in front of the fiber bundle. A ground glass diffuser is used to ensure even illumination of the fibers. Lamps used with the image intensifier have included argon, mercury, and neon. Without the image intensifier, only the mercury lamp has a bright enough emission line near plasma lines of interest, at 435 nm.

The spectrometer wavelength is set so that the calibration line is imaged onto the same location on the CCD as the plasma line. Even though the "velocity" is estimated from plasma data, this accounts for relative offsets between fibers due to curvature from the

optics. The light from each fiber is individually fit to a 2D normal distribution,

$$f(x,y) = \frac{V}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left(\frac{(x-x_0)^2}{\sigma_x^2} + \frac{(y-y_0)^2}{\sigma_y^2}\right)\right) + f_0, \tag{4.4}$$

where x is the spatial direction, y is the wavelength direction, V is the "volume" of the distribution, σ is the standard deviation, x_0 and y_0 are the centroids, and f_0 is the offset. See figure 4.3 for an example of calibration data and corresponding fits. The instrument broadening in teh wavelength direction fits the Gaussian line shape very well. Using a normal distribution with a volume parameter rather than a Gaussian with an amplitude parameter ensures that none of the fitting parameters are coupled. Since the fitting parameters in the spatial dimension are prone to interference from neighboring fibers and are not necessary, the relative intensity used is the area under the distribution in the wavelength direction only, rather than the full volume. The instrument broadening is obtained from σ_y and can easily be converted to an instrument temperature based on the mass and wavelength of the observed plasma emission. These values for the carbon data presented range between 6 and 9 eV, depending on the fiber, and 8 to 12 eV for oxygen due to the higher mass.

There are two ways to calculate the dispersion— by imaging a doublet of known wavelength separation, or by moving the spectrometer grating at a steady, known rate with a stepper motor. The separation between doublet lines or the number of pixels traversed per time (and thus wavelength) is calculated individually for each fiber to account for slightly varying dispersion across the CCD.

Finding the absolute value of velocity is made difficult on IDS for multiple reasons. The grating adjustment is not accurate to less than $\sim$10 km/s and calibration spectral lines are rarely close enough to plasma lines in wavelength space to avoid retuning the instrument. Additionally, the grating does not lock securely, so bumps or vibrations can easily cause the grating to move. One method employed to calibrate velocity is placing both fiber optic bundles on opposite sides of the volume facing each other, as shown in figure 4.2. The average velocity of the fibers directly facing each other during spheromak decay is established as zero (the decay is used to avoid injector effects, and there should be almost no poloidal flow along the chords). Another method, employed in post processing, averages

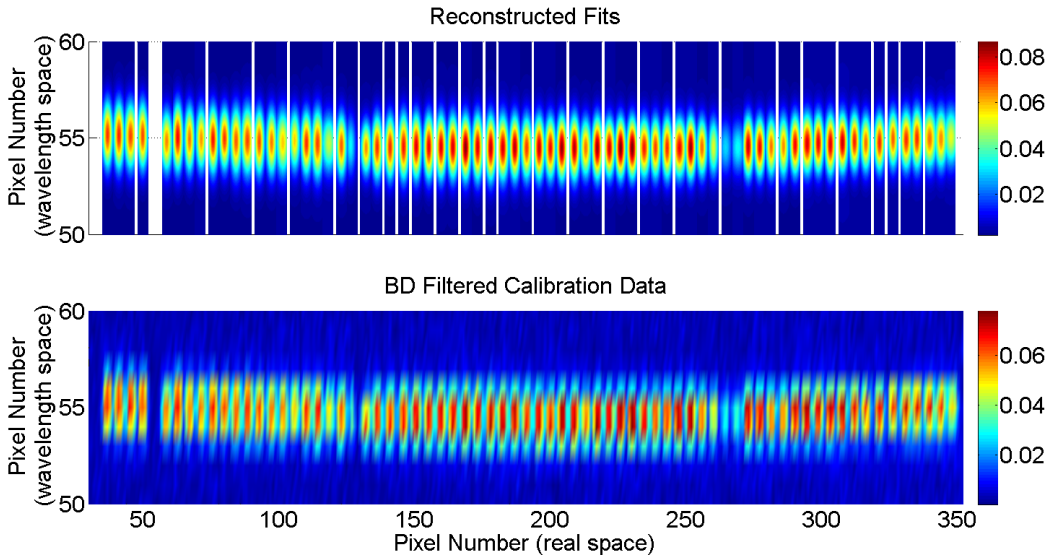

Figure 4.3: Filtered calibration data for all fibers (bottom) and fitted 2D Gaussian functions (top). The white lines between the fits are gaps between the fitting domains. The apparent "slant" in the raw data is an artifact of the coarse surface plot.

the toroidal velocities from opposite toroidal current shots during the decay phase.

It should be noted that the dominant physical effect on line shape (Doppler broadening) and the instrument broadening are both Gaussian, so the total measured spectral line can be fit to a 2D Gaussian and analytically deconvolved by subtracting the instrument "temperature".

4.4 Biorthogonal Decomposition of Data

For most applications of IDS, a 2D Gaussian function is fit to the measured spectrum to obtain the velocity and temperature for each point in space and time. Due to the high noise levels of the upgraded IDS detector, especially at short exposures, biorthogonal decomposition [13] (BD) analysis is employed to isolate coherent structures in the data. The technique was pioneered on HIT-SI by Brian Victor to isolate coherent modes in the surface magnetic fields.

To perform the analysis, data must first be arranged in a rectangular matrix with the

temporal index increasing by column and spatial index increasing by row. For "movie" data, each frame is collapsed into a column and concatenated, thus converting the 3D array to a 2D array in space vs. time. The Matlab function "svd.m" is used to decompose the data according to the formula

$$\mathbf{Y} = \phi \mathbf{A} \psi^T, \tag{4.5}$$

where $\mathbf{Y}$ is the data matrix, ϕ is an array of "topos" (spatial eigenfunctions), $\mathbf{A}$ is a square, diagonal matrix of "weights" (eigenvalues), and ψ is an array of "chronos" (temporal eigenfunctions). A data point can be reconstructed by summing the spatial point, time point, and weights over all eigenmodes:

$$y\left(x_i, t_j\right) = \sum_{k}^{K} A_k \phi_k\left(x_i\right) \psi_k\left(t_j\right). \tag{4.6}$$

Additionally, the k^{th} individual mode can be reconstructed by multiplying the k^{th} weight, topo, and chrono together. This technique yields "coherent modes" of monotonically decreasing amplitude.

As an example, the weights and first four chronos from HIT-SI pulse 129499 are plotted in figure 4.4 and the first four topos are plotted in figure 4.5. The first chrono changes very little in time and corresponds to the topo containing most of the background light and approximately half of the spectral emission. The second chrono looks remarkably similar to other emission time traces from HIT-SI, and corresponds to the steady growth of emission amplitude. Thus, when the first two modes are reconstructed they show the steady growth and decay of the spectral lines. The third and fourth chronos oscillate strongly at the injector frequency and they are small amplitude deviations from the Gaussian. Adding the third and fourth modes to the first two begins to show the effect the injectors have on the spectral lines.

Based on the plot of weights, most of the data should be in the first $\sim$10 modes, and the remainder are expected to be "noise" modes. Reconstructing the first 10 modes yields a "movie" which empirically resembles a less noisy version of the original data. The effect

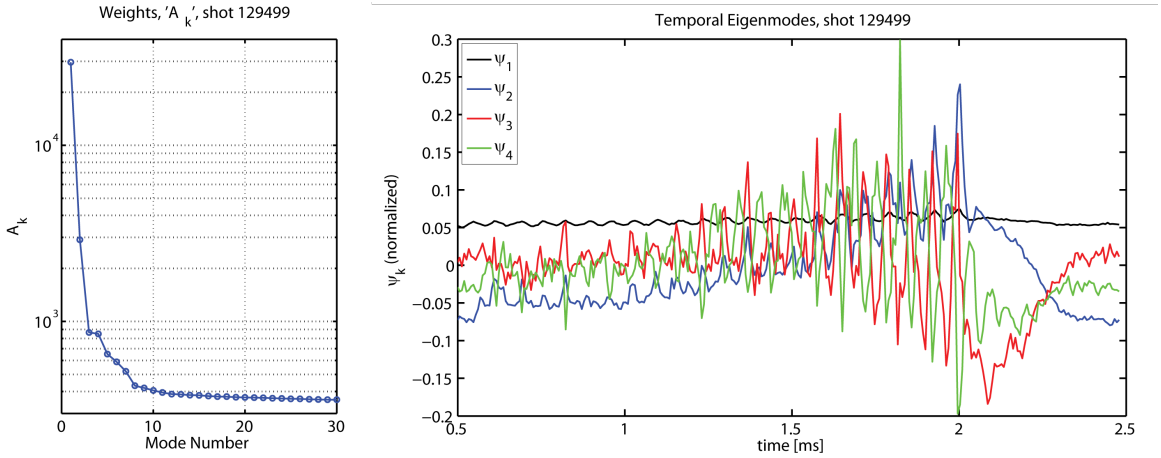

Figure 4.4: Weights of coherent modes (left) and first four chronos (right).

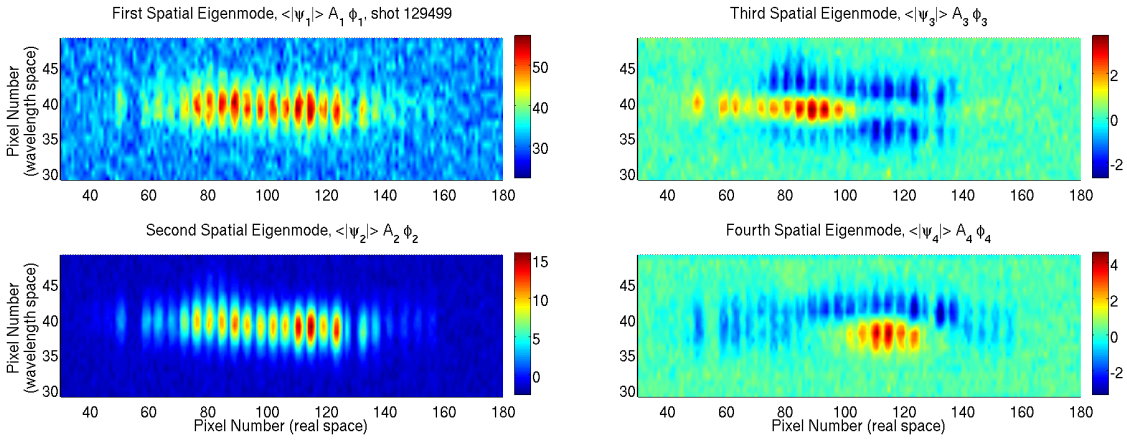

Figure 4.5: The first four *weighted* topos of HIT-SI shot 129499.

of excluding modes higher than 10 is shown in figure 4.6. The effect on fit uncertainty is also examined by comparing the results of filtered and unfiltered data. In figure 4.7 the reduction in uncertainty for temperature and velocity are plotted. The emission is lower early in the shot, hence the larger effect early in time. To ensure that significant data is not discarded, the excluded BD modes are added together to create a movie of "noise". A snapshot of the resulting movie is shown in figure 4.8. There is no discernable pattern and specifically nothing that resembles the data in figure 4.6. The advantage of this technique is that it selects the strongest components of the data regardless of frequency or spatial structure. For example, a strong, high frequency oscillation would not be "smoothed over" as would happen in frequency-based filtering. This filtering tool is used in all the data presented in this document.

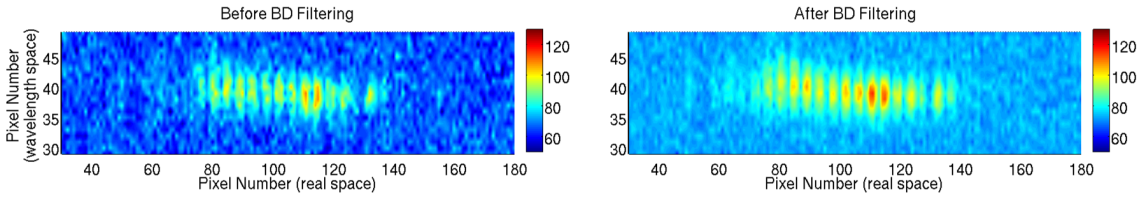

Figure 4.6: A random CCD frame before (left) and after (right) BD filtering. The shot is 129499 and time is 1.437 ms.

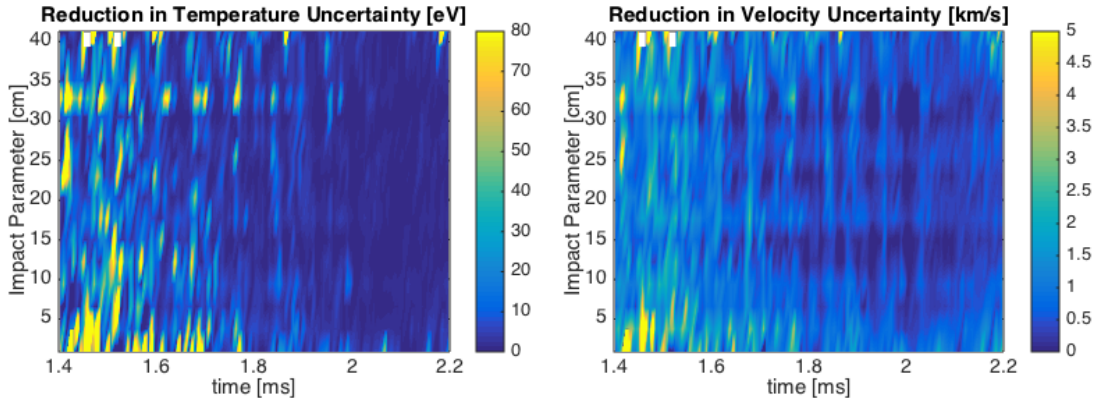

Figure 4.7: The temperature uncertainty of filtered data subtracted from that of unfiltered data (left) and the velocity uncertainty of filtered data subtracted from that of unfiltered data (right).

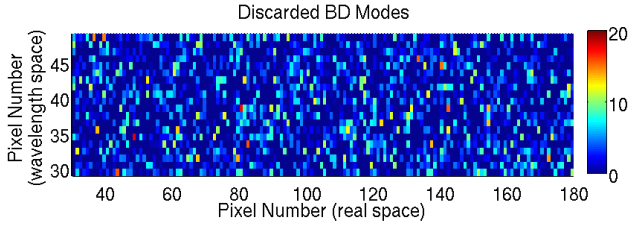

Figure 4.8: A section of the "discarded" movie for shot 129499 and time 1.437 ms. Modes 11 through 291 are included.

Chapter 5

DATA FROM HIGH–CURRENT HIT-SI PLASMAS

The final campaign of HIT-SI at a driving frequency of 14.5 kHz yielded the brightest data recorded and the image intensifier was removed from the optical system. This produced a data set with high temporal resolution of approximately 10, 6.9 μs exposures per injector cycle. Additionally, the tangential, reentrant port was installed (see figure 4.1, right) yielding chords spanning from the geometric axis to impact parameter $R \simeq 41$ cm.

The spatial resolution and span of the system enables measurements of chord-averaged profiles. The intensity profile can be related to density and the temperature and velocity profiles can be directly measured. Since HIT-SI operates with positive or negative toroidal current, the effect of current direction on velocity profiles can be found.

The sub-injector-cycle time resolution enables studying dynamics related to the injector-linking plasma. The phasing of velocity oscillations can be used to separate flows in the injector flux from spheromak motion due to the time-varying injector flux and current.

5.1 Parameters of 80 − 90 kA, 14.5 kHz HIT-SI Shots

To put the IDS measurements in context, a selection of data from shot 129499 are shown in figure 5.1. More data from this shot can be found in appendix B.1. Toroidal current steadily increases to 90 kA as injector currents are ramped up to nearly 30 kA each. Current amplification of 3 is typical for shots like this and no other device has achieved higher amplification (although HIT-SI approached 4 at very high injector frequency). Record power of up to 15 MW was injected thanks to feedback controlled power supplies and efficient, resonant circuits. A filtered photomultiplier tube detecting continuum radiation sees steady growth of emission throughout the shot, similar to bolometric and ion emission signals. The ratio of C IV to C III as measured by the VUV spectrometers increases but then declines. This ratio is related to temperature and indicates that HIT-SI's performance

may be limited by an inability to "burn through" due to radiation losses and over-fueling. Although the interferometer was not able to consistently resolve electron density at these parameters, the density of shots like 129499 increases steadily to $5 - 7 \times 10^{19}$ m^{-3}.

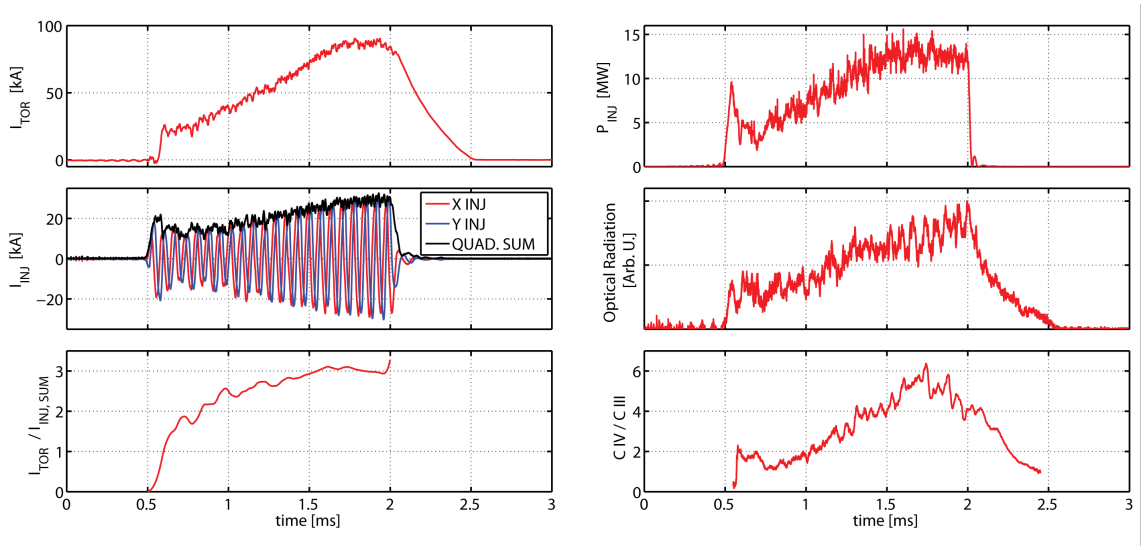

Figure 5.1: Measured quantities for HIT-SI shot 129499. Plotted are toroidal current as measured by surface probe arrays (upper left), X and Y injector currents and their quadrature sum (middle left), and current amplification, a common figure of merit for spheromaks (lower left). The power injected is plotted in the upper right, broadband radiation as measured by a bandpass filter (middle right), and the ratio of C IV to C III (lower right).

5.2 Plasma Spectral Lines

One of the brightest emission lines observed by the SPRED spectrometer (see section 2.4.3) is at ∼465 nm. When the IDS was tuned to 465 nm the line was resolved as three separate lines. The similar brightness and spacing suggests they are the components of a C III triplet (464.742, 465.025, and 465.147 nm). However, their spacing on the CCD does not quite match the calibrated wavelength per pixel and their relative spacing does not match the theoretical values to within fitting uncertainty. After searching for other candidate lines in the vicinity, the best fit for both relative wavelength spacing and wavelength per pixel is obtained if the middle line is O II ($\lambda = 464.913$ nm) and the other two lines are the lower

two C III lines (see figure 5.2). This spectral profile is very similar to that observed in the edge of the COMPASS tokamak with similar plasma conditions [40].

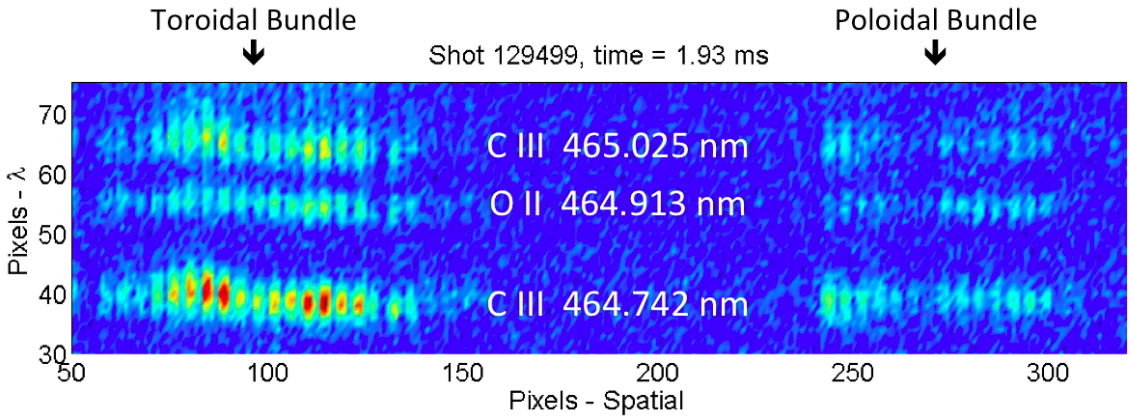

Figure 5.2: CCD frame from near the end of shot 129499 with spectral lines identified. Since only the middle ∼20 fibers from each sub-bundle receive light, the toroidal and poloidal sub-bundles appear separated on the detector. The longest wavelength C III line is weak and just out of the frame.

The C III data shown in the following sections are taken from the shortest wavelength line since it is brightest.

5.3 Error Analysis using C III Lines

The error bars presented in this work are the uncertainty of the fit to the data for the specified parameter (temperature or velocity). To ensure that this method reflects the true uncertainty, it is compared with an empirical error estimate. Since the two C III lines are part of a triplet from the same electronic transition, $1s^2 2s3p \rightarrow 1s^2 2s3s$, they should yield the same plasma parameters. Differences between them reflect the random uncertainty in the entire measurement system. Figure 5.3 shows the difference in temperature (left) and velocity (right) between the two lines as a function of intensity. The time window for the data shown is 1.664 to 1.85 ms, which will be used for comparing to simulations in chapter 6. The temperature differences are less than 10 eV for the majority of data points. For velocity, most of the differences lie below $\Delta v = 2$ km/s.

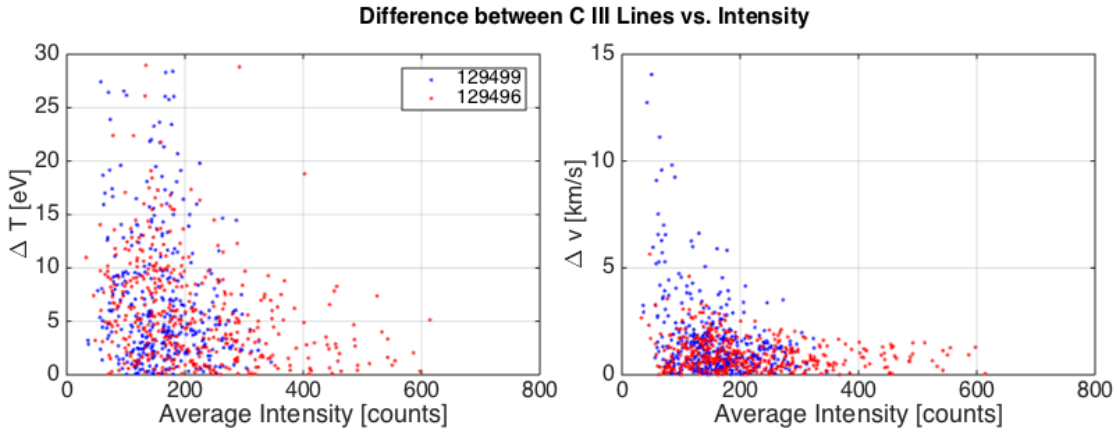

Figure 5.3: Differences in temperature (left) and velocity (right) between C III lines at 465.025 nm and 464.742 nm for two similar shots. The time window for the data is 1.664 to 1.85 ms.

The empirical differences between C III lines are compared with the average fit uncertainties for the same lines in figure 5.4. The mean empirical temperature uncertainty is 8 eV whereas the fit uncertainty is 6 eV. The mean empirical velocity uncertainty is 1.3 km/s but the fit uncertainty is 1.1 km/s.

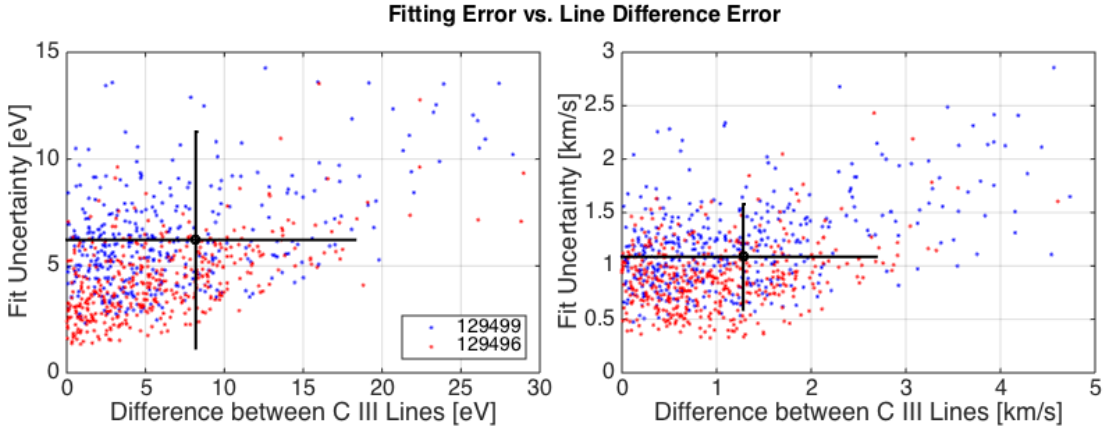

Figure 5.4: Fit uncertainty vs. C III line differences for temperature (left) and velocity (right). Means and standard deviations (black) include all data.

5.3.1 Discussion of Error Analysis

This analysis shows that on average the fit uncertainty is slightly less than the empirical uncertainty. The wider distribution of empirical uncertainties is expected since this method is random (i.e.: the C III temperatures may be exactly the same or anomalously different, but the statistical estimate never has less than 1 eV uncertainty). Since the empirical method is the difference between two independent measurements, it overestimates the error of an individual fit by a factor of $\sqrt{2}$. Therefore, by acounting for this factor, the statistical fit uncertainty is a slightly more conservative estimate of error and is confidently used in this work. Additionally, it should be noted that the empirical method is not viable for individual fits due to its random nature.

5.4 Temperature and Intensity Measurements from Shot 129499

Temperatures of C III and O II over the entire shot are plotted in figures 5.5 and 5.6. C III is consistently hotter than O II. The higher uncertainty early in time is due to low signal intensity. Temperature fluctuations at the injector frequency are visible on most chords, especially at impact parameters $R \simeq 12$ and 9 cm. After the injectors shut off the C III and O II temperatures fall to ∼10 and 5 eV, respectively, for ~ 50 μs. The C III temperature then falls to ∼5 eV, within uncertainty of zero.

To examine equilibrium quantities the data from figures 5.5 and 5.6 are time-averaged during sustainment, from $t = 1.6 - 2.0$ ms, and the chord-averaged profiles are plotted in figure 5.7. Note that the error bars are the certainties of the means (standard deviation divided by the square root of the number of measurements). Near the magnetic axis, at impact parameter $R \simeq 31-33$ cm, the temperatures are relatively low: 19 eV for C III and 8 – 10 eV for O II. Maximum average temperatures are observed at small impact parameters: up to 35 eV for C III and 20 eV for O II.

The intensities for C III and O II from shot 129499 are plotted in figures 5.8 and 5.9. The C III emission is generally brighter than O II and exhibits larger fluctuations, primarily at the injector frequency. Note the increase in intensities during the shot, resulting in high fitting uncertainty for temperatures and velocities early in time. The time and chord-

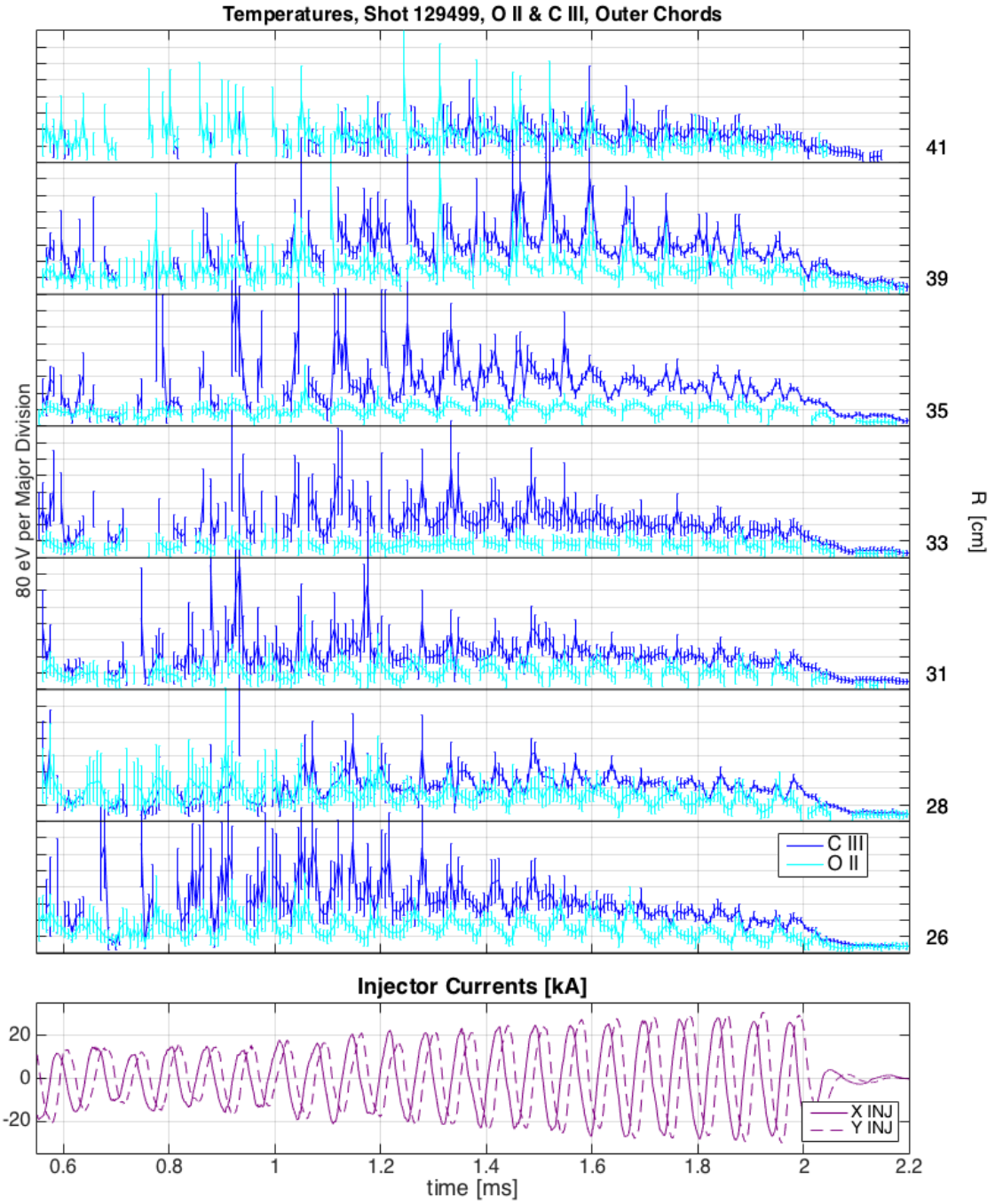

Figure 5.5: Chord-averaged temperatures from O II and C III emission from shot 129499. Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. The fiber corresponding to impact parameter $R \simeq 37$ cm is broken. X and Y injector currents for shot 129499 are shown below.

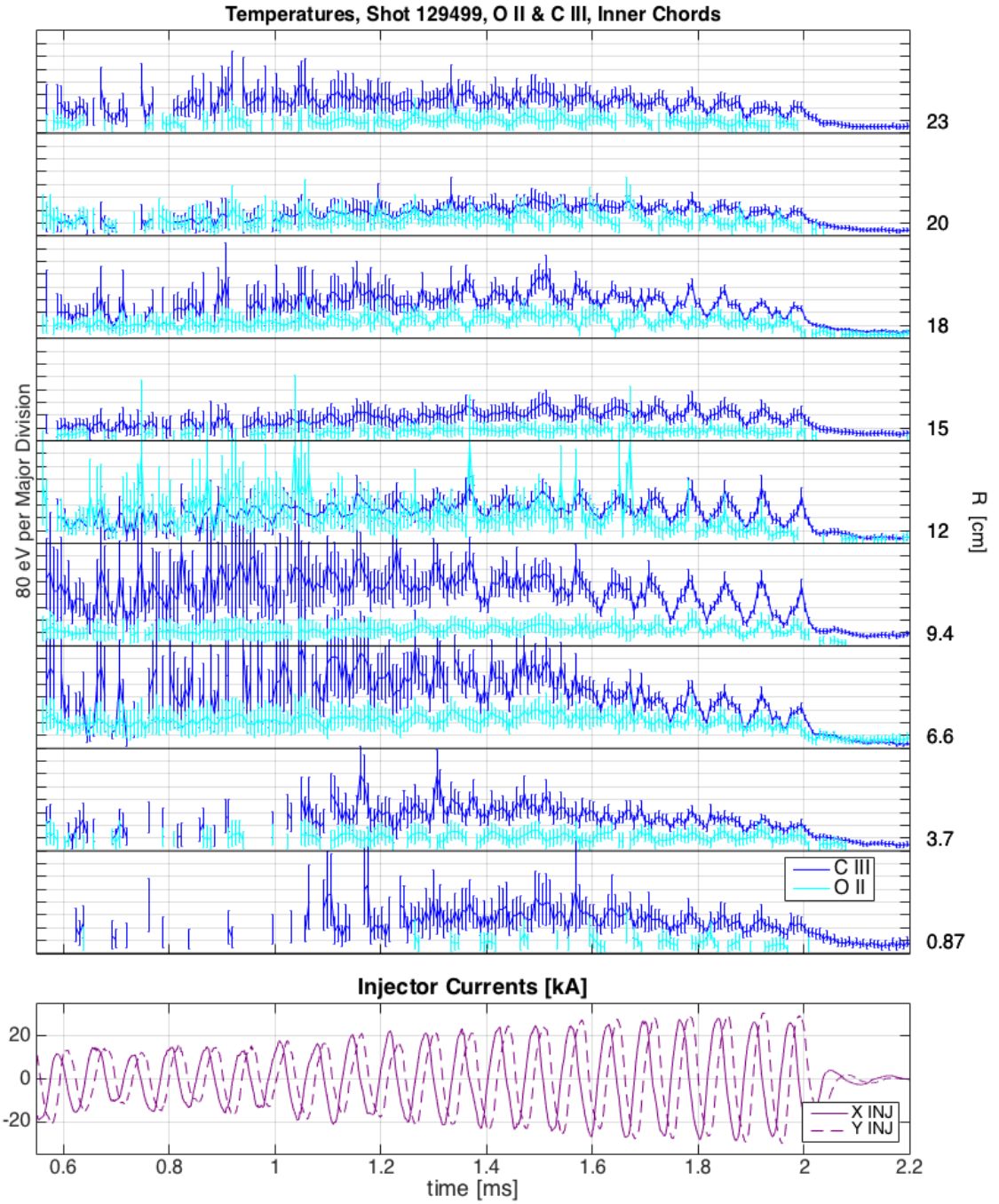

Figure 5.6: Chord-averaged temperatures from O II and C III emission from shot 129499. Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

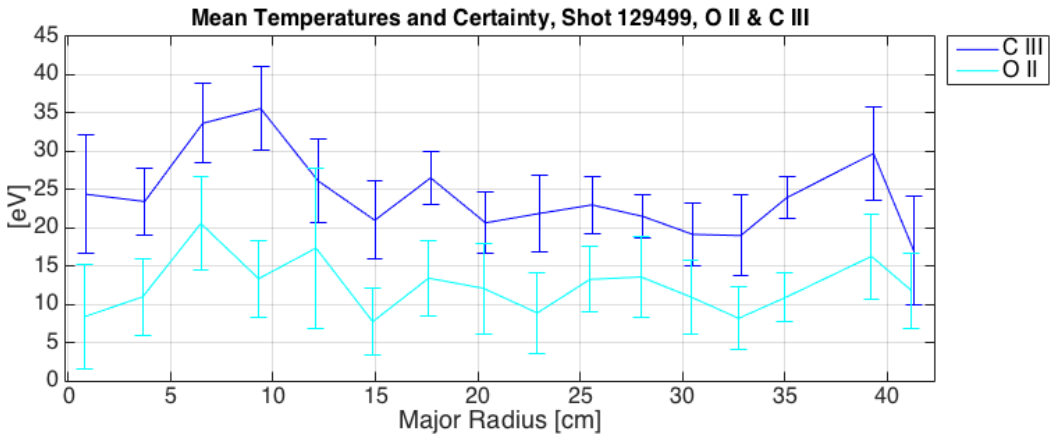

Figure 5.7: Average temperatures and certainties of O II and C III emission from shot 129499. Every available chord from the array is displayed. Statistics are calculated from $t = 1.6 - 2.0$ ms. Note the error bars are certainty $(\sigma/\sqrt{N})$ where $N = 58$ and σ includes fitting uncertainty and the standard deviation of the data.

averaged intensity profiles during sustainment are plotted in figure 5.10. The averages and standard deviations are calculated during sustainment, from $t = 1.6 - 2.0$ ms. The C III fluctuations are clearly larger than O II. Also, both ions exhibit a trend toward higher emissivity farther outboard.

5.5 Velocity Measurements from Positive and Negative Toroidal Current Shots

Velocity measurements were obtained for positive (129499) and negative (129496) toroidal current shots. Chord-averaged velocities from C III emission from shots 129499 and 129496 are shown in figures 5.11 and 5.12. Velocities from O II emission from the same shots and chords are shown in figures 5.13 and 5.14. The sparse data and higher error early in time is due to low-intensity data as shown above. Both O II and C III data show oscillations at the injector frequency outboard of impact parameter $R \sim 18$ cm. The time-averaged means and standard deviations for C III from both shots are shown in figure 5.15. The means and standard deviations for O II are shown in figure 5.16.

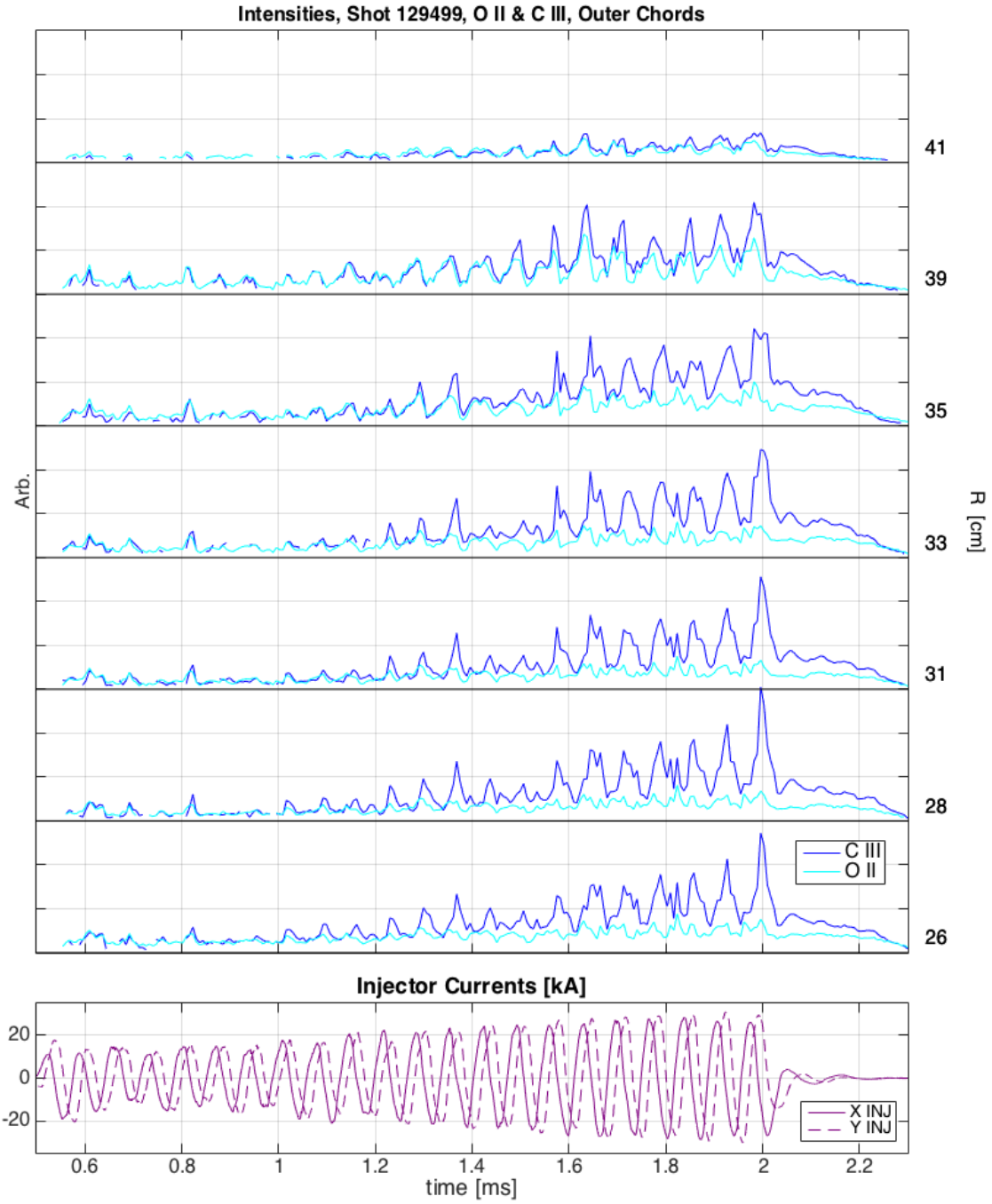

Figure 5.8: Intensities from O II and C III emission from shot 129499. Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars are not plotted because the uncertainty is too small to plot. X and Y injector currents for shot 129499 are shown below.

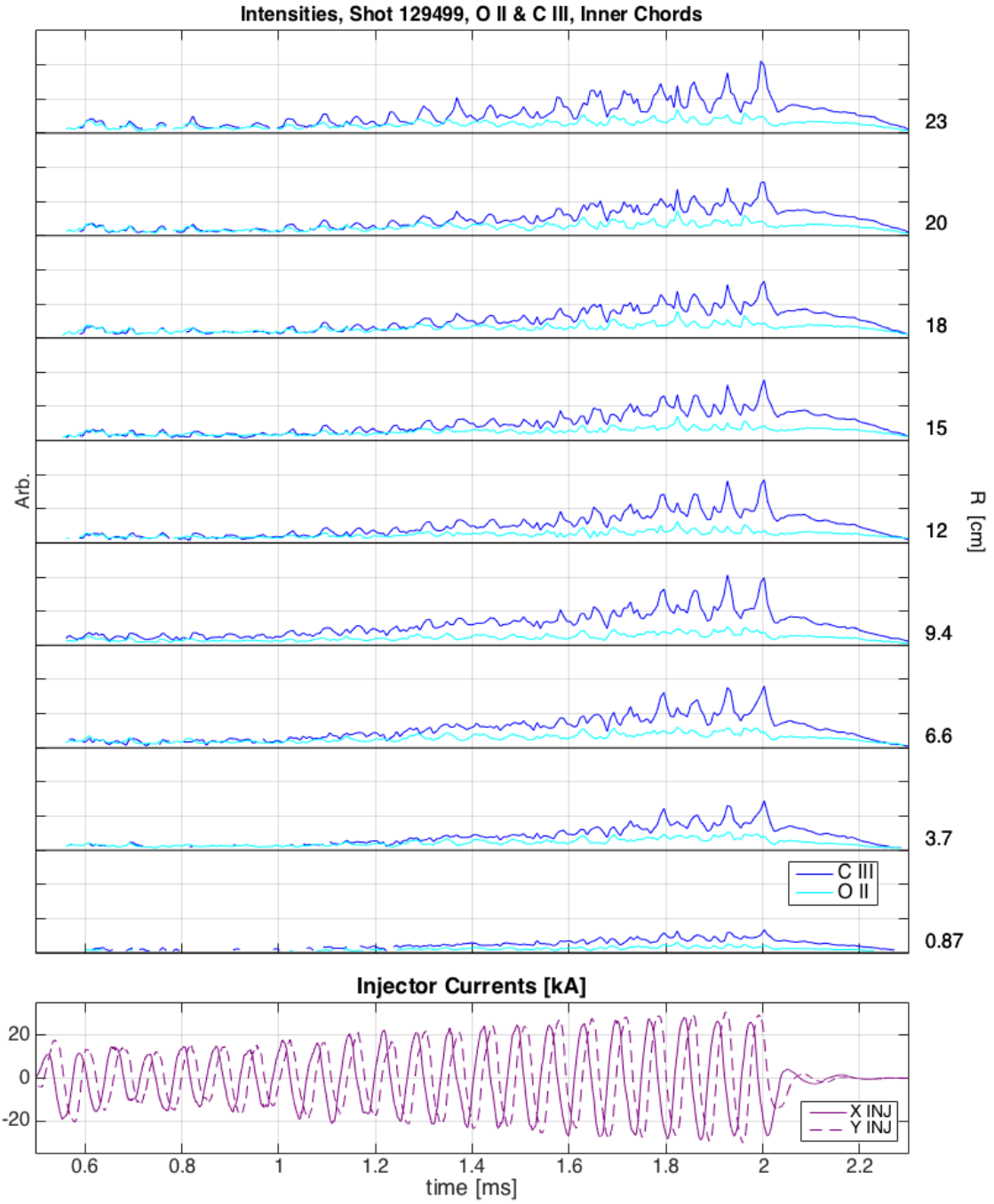

Figure 5.9: Intensities from O II and C III emission from shot 129499. Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars are not plotted because the uncertainty is too small to plot. X and Y injector currents for shot 129499 are shown below.

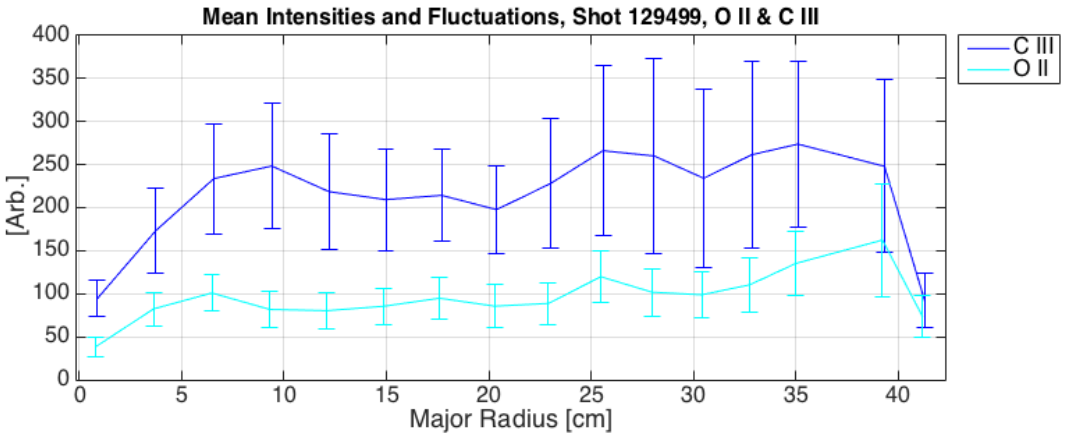

Figure 5.10: Average intensities and fluctuations of O II and C III emission from shot 129499. Every available chord from the array is displayed. Averages and fluctuations are calculated from $t = 1.6 - 2.0$ ms.

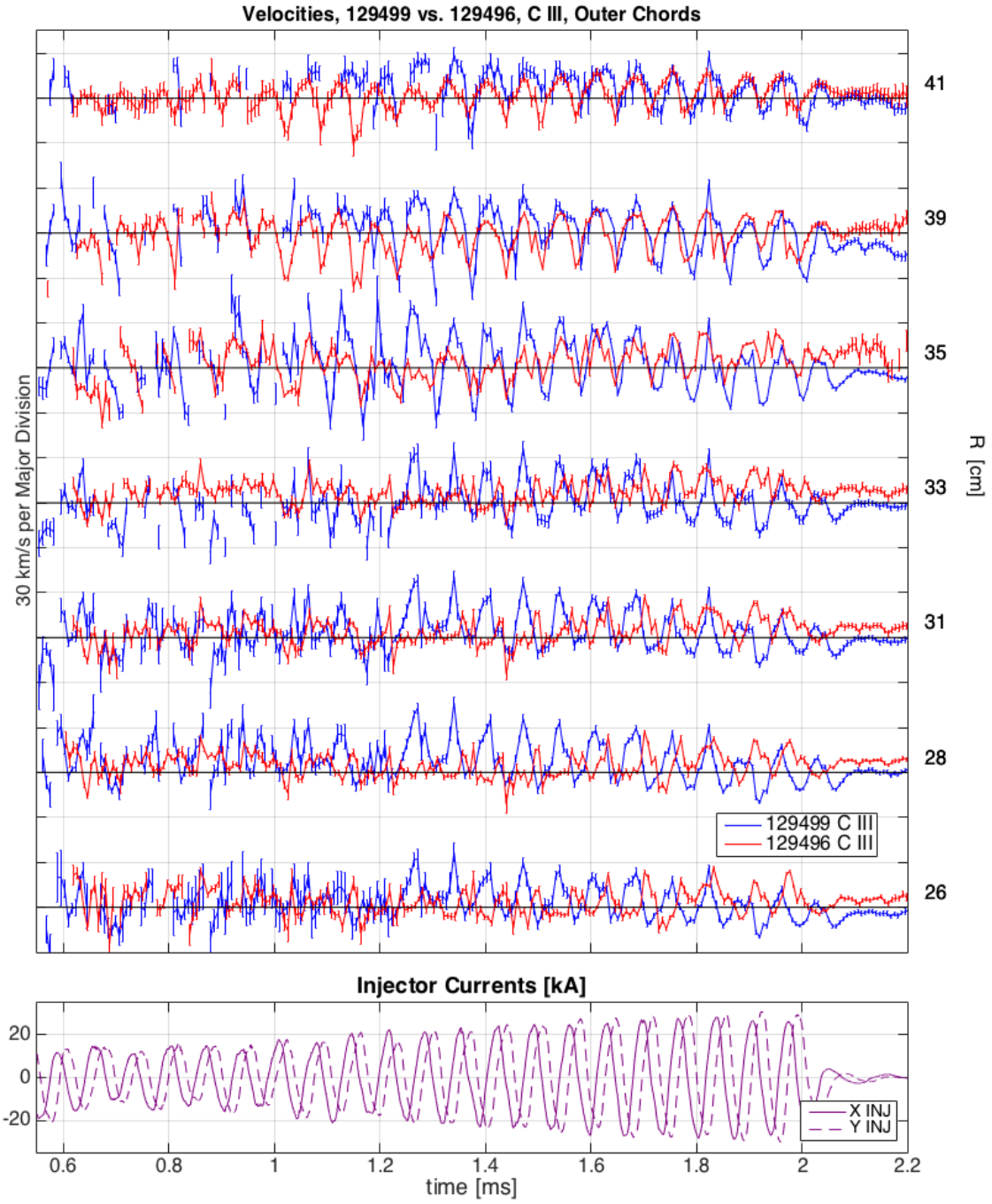

Figure 5.11: Velocities along the outer chords from shot 129499 (blue, positive I_{tor}) and 129496 (red, negative I_{tor}). Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

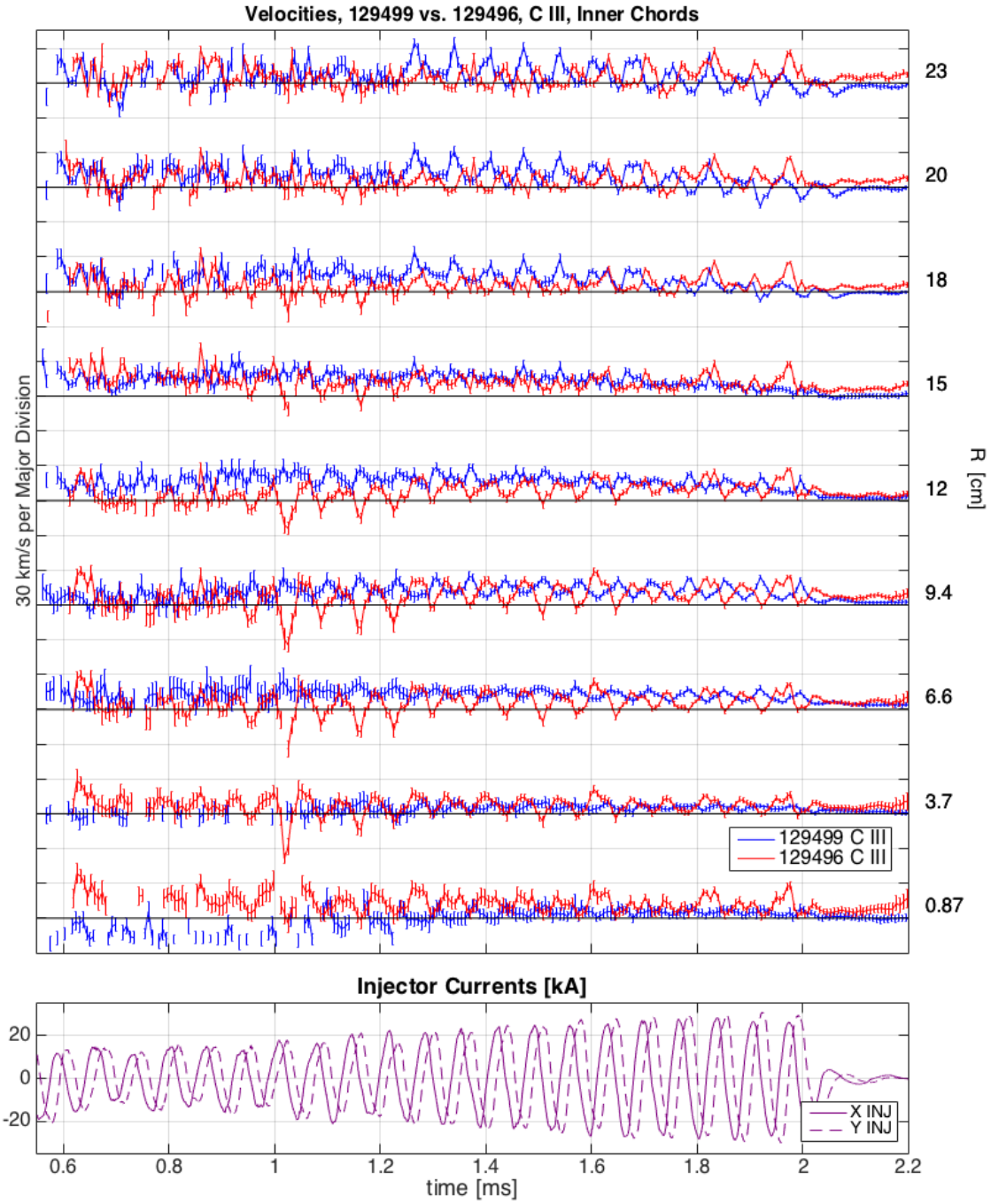

Figure 5.12: Velocities along the inner chords from shot 129499 (blue, positive I_{tor}) and 129496 (red, negative I_{tor}). Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

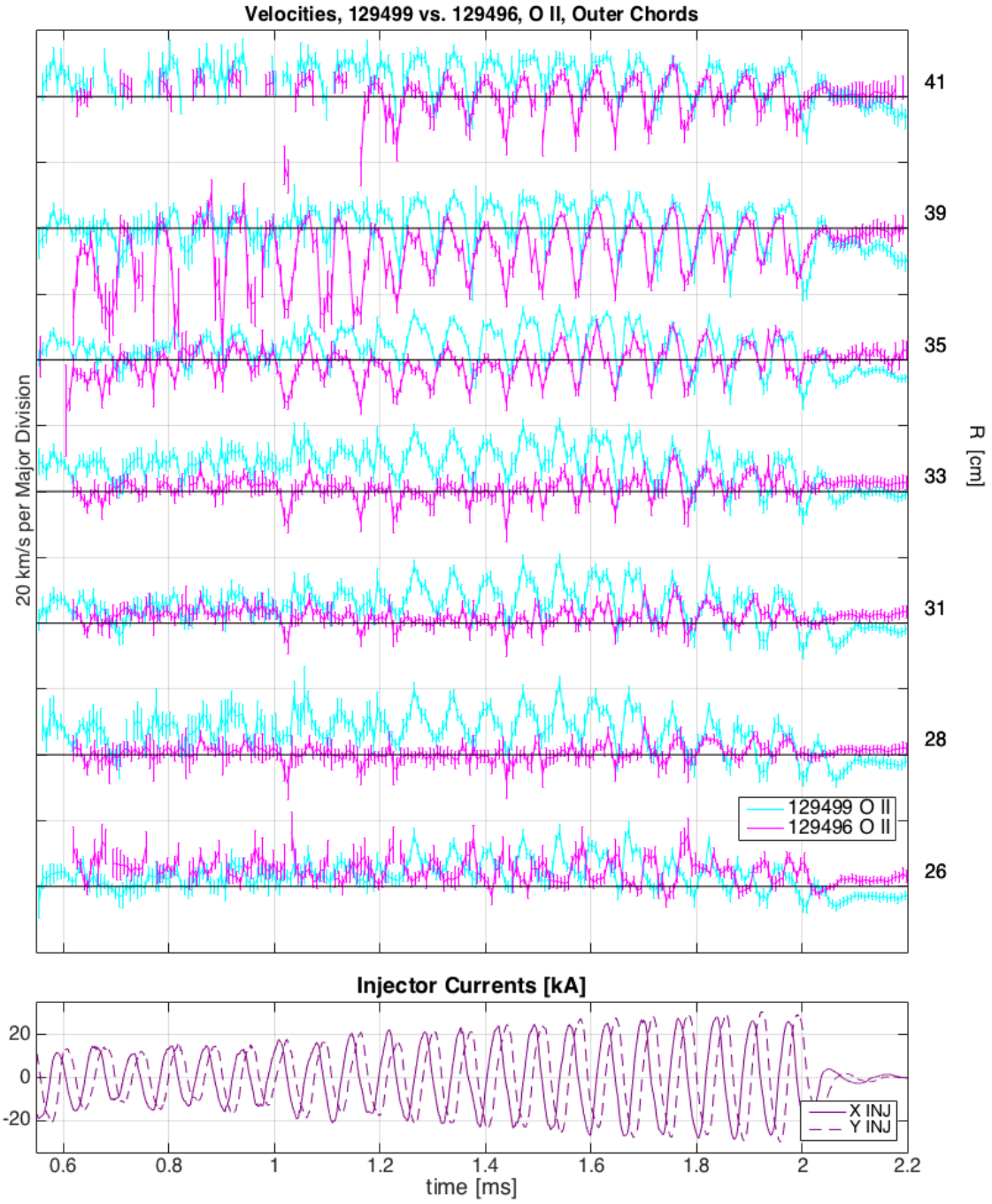

Figure 5.13: Velocities along the outer chords from shot 129499 (cyan, positive I_{tor}) and 129496 (magenta, negative I_{tor}). Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

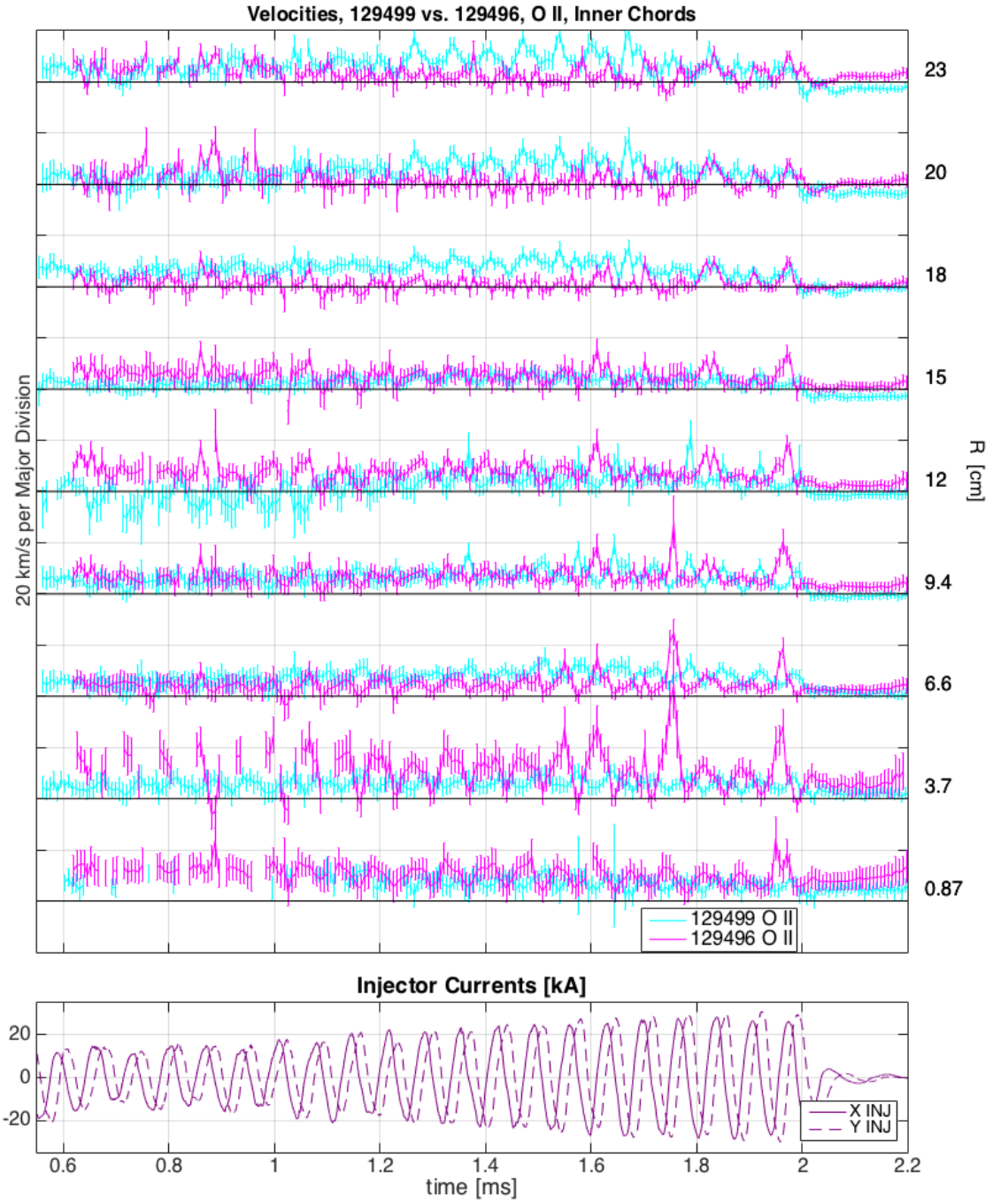

Figure 5.14: Velocities along the inner chords from shot 129499 (cyan, positive I_{tor}) and 129496 (magenta, negative I_{tor}). Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

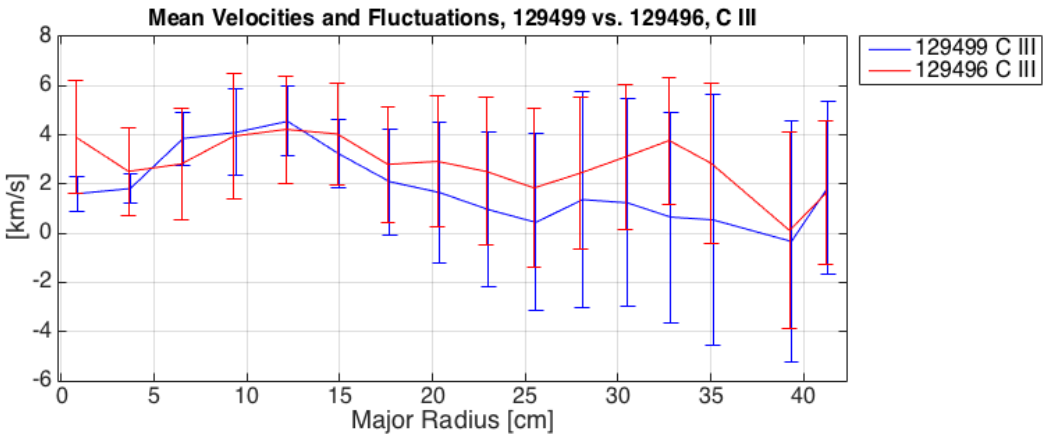

Figure 5.15: Time-averaged velocities for C III from shots 129499 and 129496 from $t = 1.6$ to 2.0 ms. All available chords are displayed from the toroidal fiber sub-bundle: from the geometric axis to $R \simeq 41$ cm. The error bars are the standard deviations of the velocities representing the fluctuation amplitude.

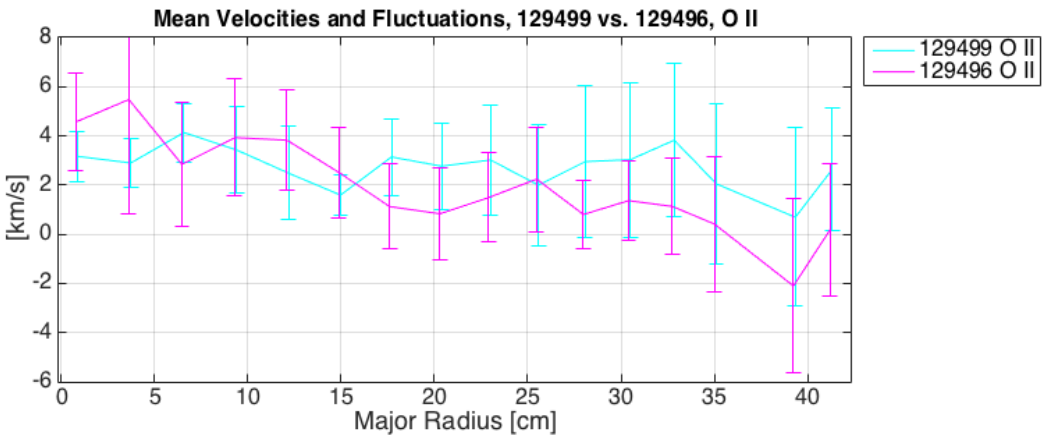

Figure 5.16: Time-averaged velocities for O II from shots 129499 and 129496 from $t = 1.6$ to 2.0 ms. All available chords are displayed from the toroidal fiber sub-bundle: from the geometric axis to $R \simeq 41$ cm. The error bars are the standard deviations of the velocities representing the fluctuation amplitude.

5.6 Additional Velocity Measurements

To show temporal detail and repeatability, the velocities from three similar shots are plotted in figures 5.17 and 5.18. All chords between $R \simeq 1$ and 41 cm are plotted and the time span is less than 3 injector periods.

5.7 Measurements of Magnetic Toroidal Fourier Modes

As mentioned in section 2.4.1, HIT-SI has two arrays of magnetic probes around the toroidal midplane. The following procedure exactly follows the method of Victor *et al.* [54]. For each point in time the magnetic field measured by each array is decomposed into spatial Fourier modes [56] and the results from the two arrays are averaged. The amplitudes of modes 1 through 4 from shot 129499 are time-averaged over an injector cycle and plotted in figure 5.19(a). The amplitude of the $n = 1$ mode is clearly much higher than the others, but the injectors have $n = 1$ symmetry and thus impose most of the $n = 1$ energy. In order to separate the injector-imposed components from those in the spheromak, biorthogonal decomposition [13] is employed because it isolates temporally correlated signals. The surface probe and injector current signals are arranged into a matrix $B_{x,t} = B\left(\left[x_m, CI_{inj,x}, CI_{inj,y}\right], t_n\right)$ where C is a scale factor which forces the injector currents to be the second and third largest BD modes. The matrix is decomposed using singular value decomposition into the form

$$B\left(\left[x_m, CI_{inj,x}, CI_{inj,y}\right], t_n\right) = \sum_{k=1}^{K} A_k \phi_k\left(\left[x_m, CI_{inj,x}, CI_{inj,y}\right]\right) \psi_k\left(t_n\right), \tag{5.1}$$

where k is the empirical mode number, A_k is the weight (eigenvalue), ϕ_k is the spatial eigenvector and ψ_k is the temporal eigenvector. The 1^{st} mode has the most energy and largest weight and the K^{th} mode has the smallest weight. As mentioned above, the X and Y injector currents are enhanced so that they are the 2^{nd} and 3^{rd} modes. Since BD groups temporally coherent structures together, any signal component directly correlated with the injector currents is included in those modes. The second and third modes are then

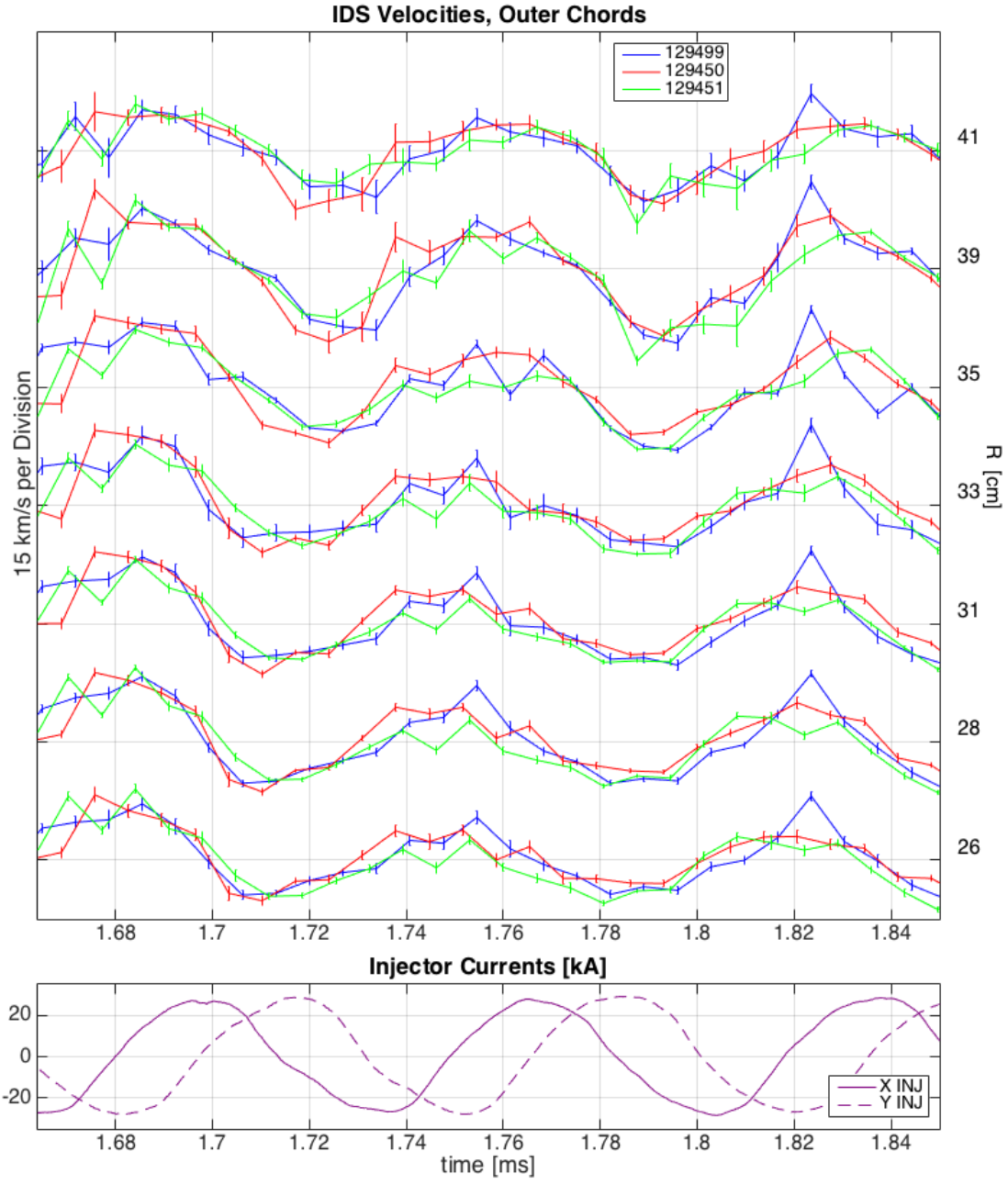

Figure 5.17: Chord-averaged velocities from $R \simeq 26$ to 41 cm for three similar shots. Horizontal lines mark zero-velocity for each chord, labeled on the right. Errorbars are fitting uncertainty. Injector currents plotted below are for shot 129499 only.

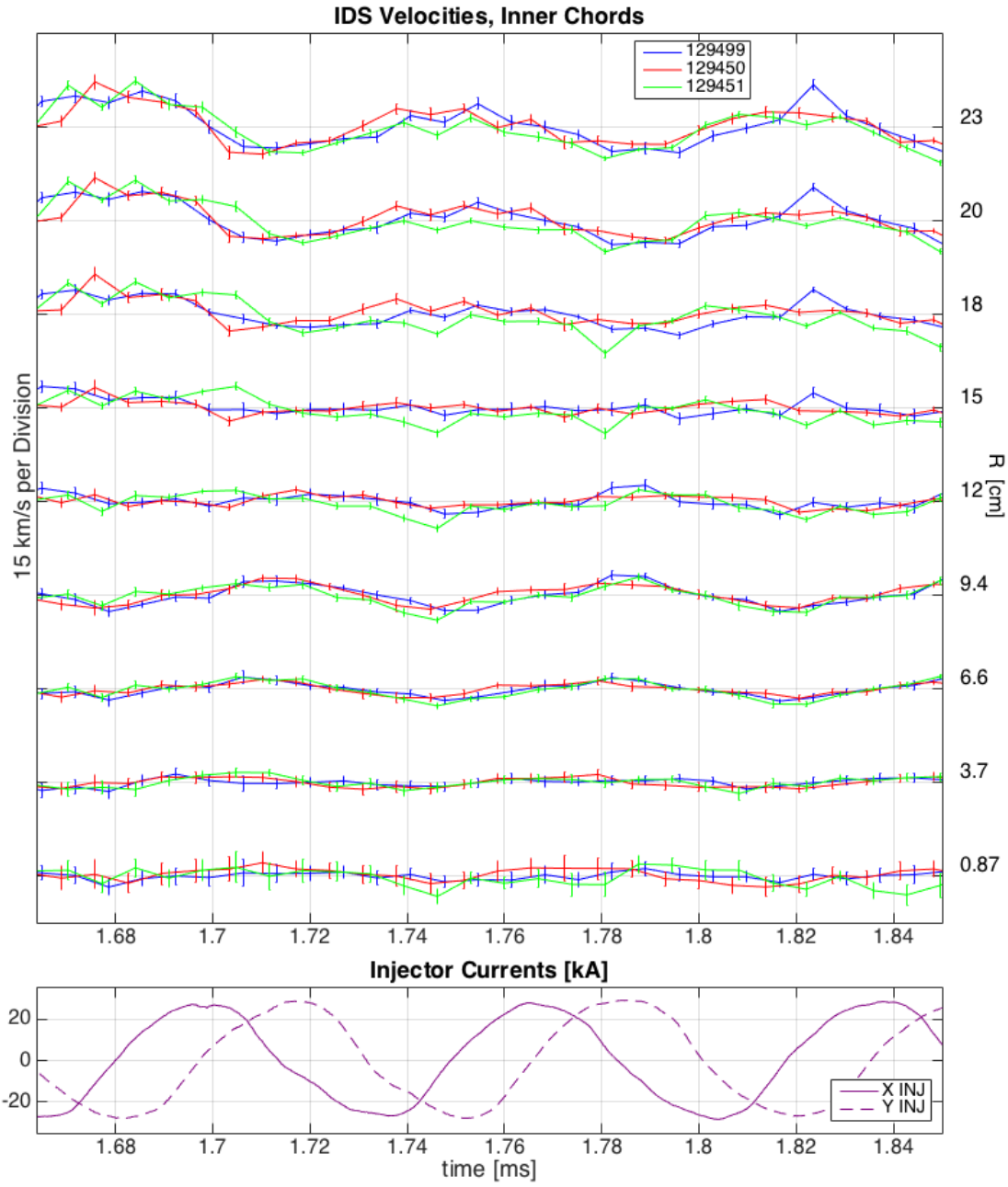

Figure 5.18: Chord-averaged velocities from $R \simeq 1$ to 23 cm for three similar shots. Horizontal lines mark zero-velocity for each chord, labeled on the right. Errorbars are fitting uncertainty. Injector currents plotted below are for shot 129499 only.

subtracted from the data matrix:

$$B_{sub}(x,t) = B(x,t) - A_2\phi_2(x)\,\psi_2(t) - A_3\phi_3(x)\,\psi_3(t)\,. \tag{5.2}$$

The resulting matrix of magnetic probe signals with injector-correlated components subtracted is decomposed into spatial Fourier modes as before. The results are plotted in figure 5.19(b). Note that the $n = 1$ component is now lower than $n = 2, 3$, and 4 after the initial spheromak formation. To further show the correlation between the injector current and $n = 1$ mode energy, the two are scaled so that they overlay in figure 5.19(c). The toroidal current is also plotted in figure 5.19(d).

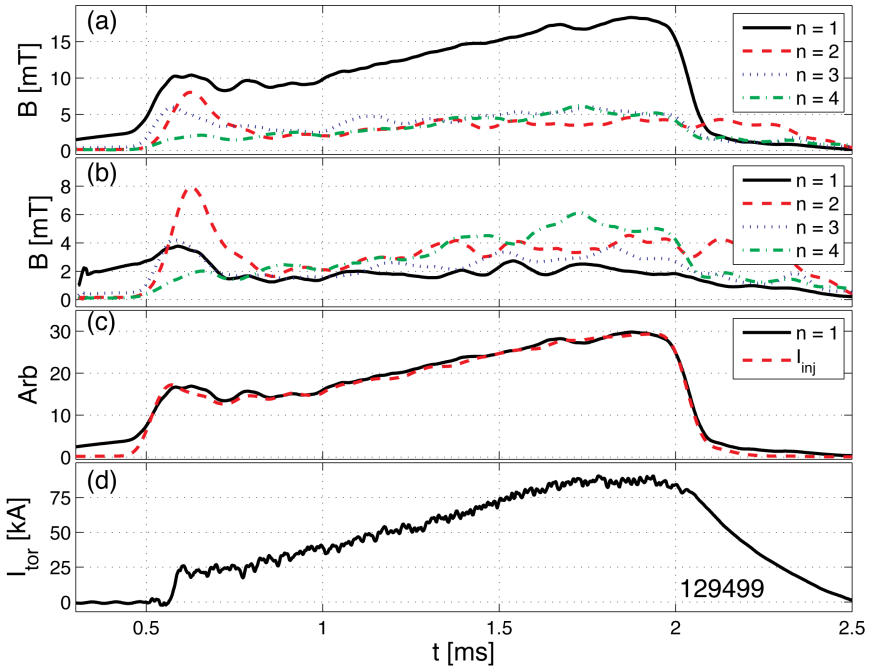

Figure 5.19: (a) Toroidal Fourier modes as measured by the midplane surface magnetic probes time-averaged over an injector cycle. (b) The same modes with injector-correlated components subtracted. (c) The original $n = 1$ mode and quadrature injector current traces overlayed to show correlation. (d) Toroidal current.

5.8 Theoretical Temperature Evolution

The measured temperature profiles are compared with the theoretical evolution of a perfectly confined, Ohmically-heated plasma. The thermal energy of a plasma is related to Ohmic heating by the equation

$$3nk_BT = \eta j^2 \tau_h, \tag{5.3}$$

where $n = n_e = Z_{eff} n_i$, $T \simeq T_e \simeq T_i$, η is the temperature-dependent resistivity, j is the current density, and τ_h is the characteristic heating time. Solving equation 5.3 yields the differential equation

$$\frac{dT}{dt} = \frac{\eta j^2}{3nk_B}. \tag{5.4}$$

For an initial condition, the IDS temperature data from figures 5.5 and 5.6 are time-averaged from $t = 1.0$ to 1.2 ms since data are sparse due to weak signal earlier than $t = 1.0$ ms. The plasma density is estimated to be 6×10^{19} m^{-3}, as mentioned in section 5.1. The current density is estimated by dividing the toroidal current by the minor cross sectional area, 0.2 m^2. The temperatures for O II and C III are evolved using this model and plotted with the measured data in figures 5.20 and 5.21.

To further illustrate the differences between the model and data, radial profiles are plotted for two points in time. The IDS data and model are time-averaged from $t = 1.55$ to 1.65 ms, near the end of the toroidal current ramp, and the profiles are plotted in figure 5.22. Note that the model profiles are typically 2 – 10 eV higher than the measured data.

A similar 100 μs average is taken at the end of the shot, from $t = 1.9$ to 2.0 ms, and the profiles are plotted in figure 5.23. Note that the model temperatures are typically 15 – 25 eV higher than the measured temperatures.

5.9 Analysis of Velocity Oscillations

In order to compare the phase of the velocity oscillations with the injector current, the injector-frequency components of the velocities are isolated with Fourier analysis. The time

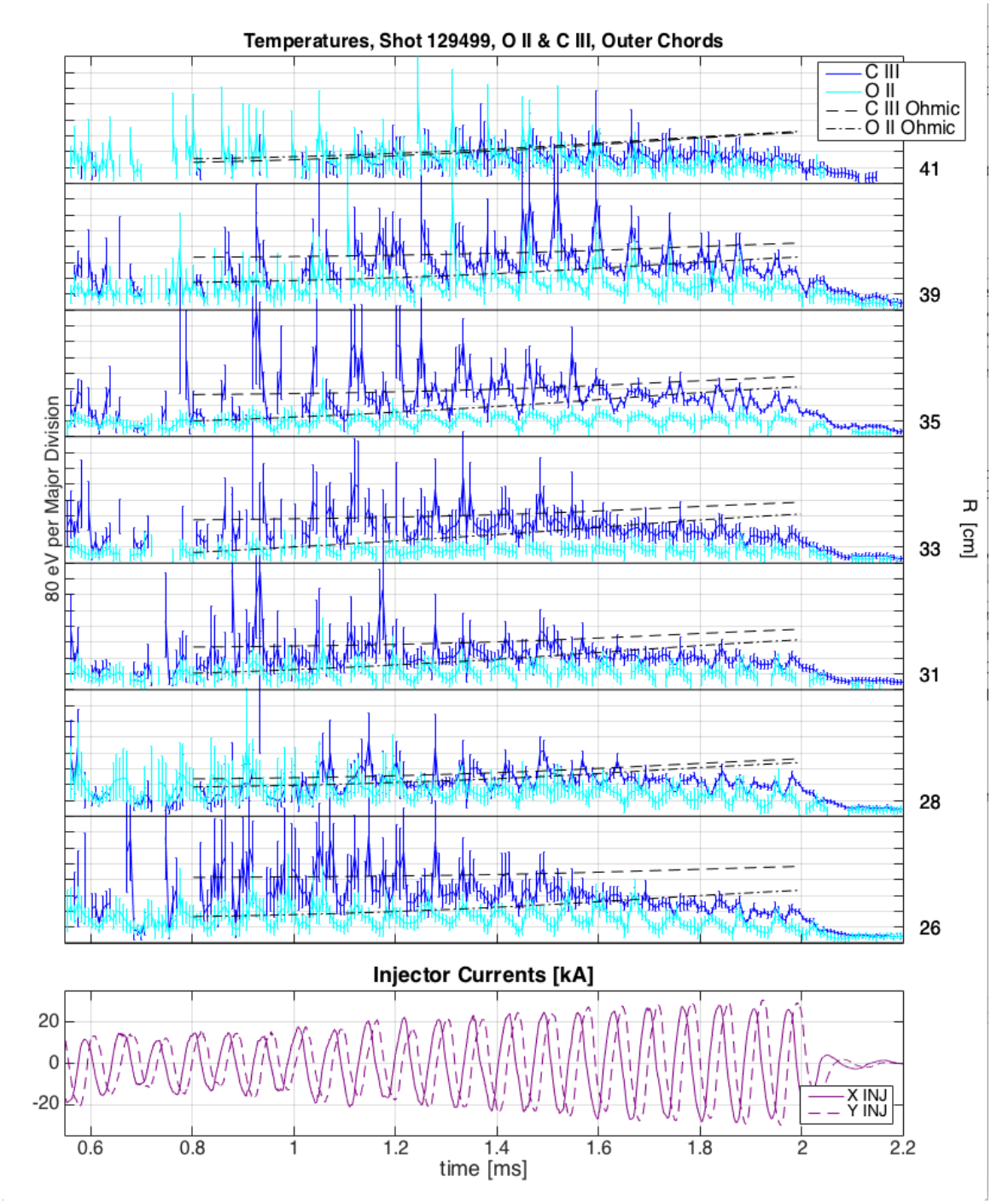

Figure 5.20: Temperatures from O II and C III emission from shot 129499 and theoretical temperatures for each assuming Ohmic heating and perfect confinement. Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

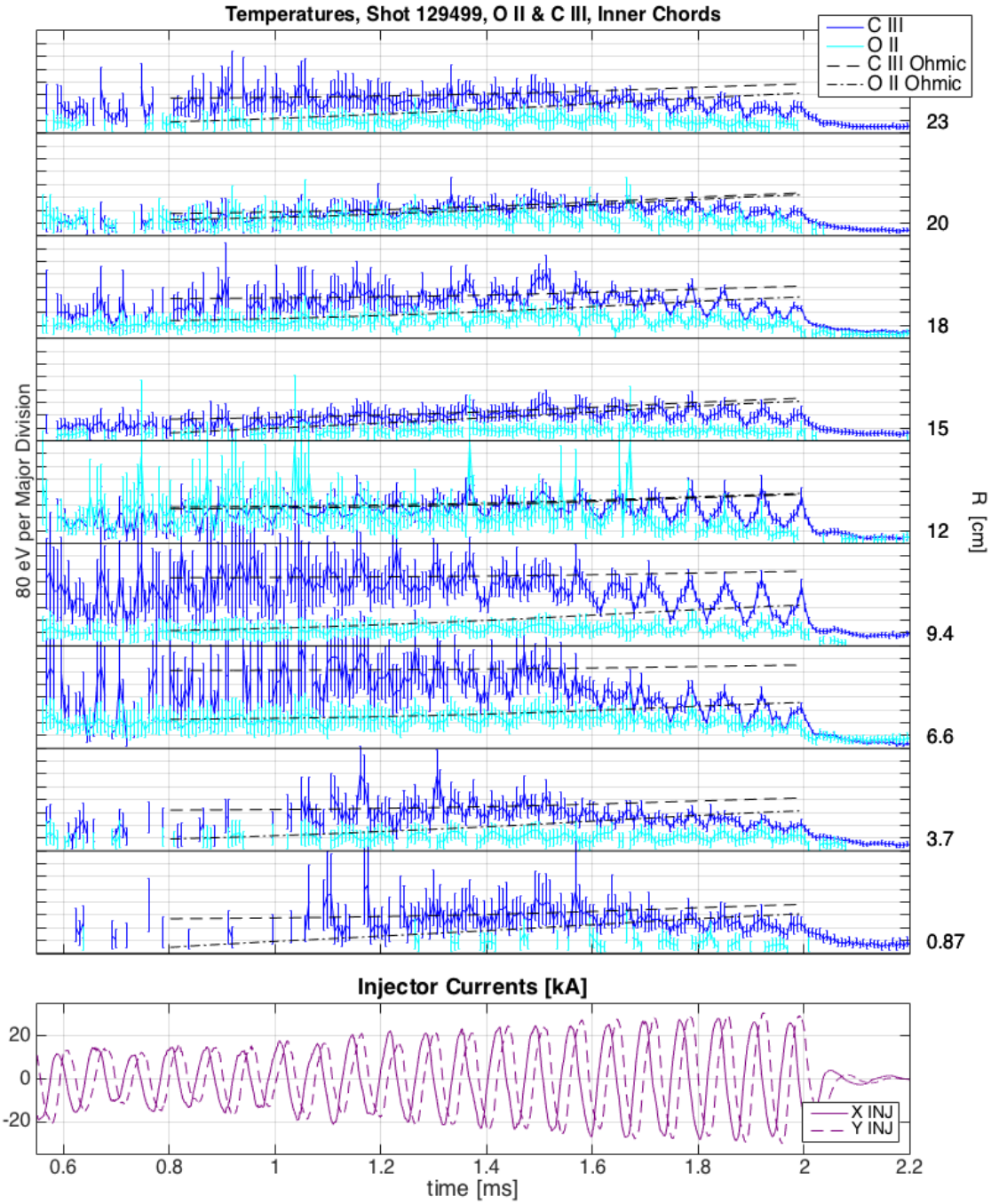

Figure 5.21: Temperatures from O II and C III emission from shot 129499 and theoretical temperatures for each assuming Ohmic heating and perfect confinement. Black, horizontal lines are the zeros for each chord and corresponding impact parameters are displayed on the right. Error bars represent the fitting uncertainty only. X and Y injector currents for shot 129499 are shown below.

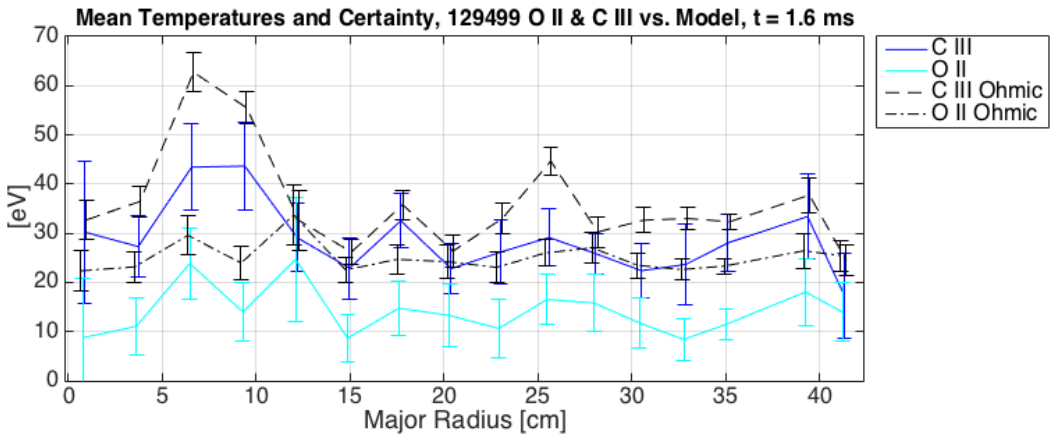

Figure 5.22: Temperatures from O II and C III emission from shot 129499 and theoretical temperatures. All data sets are time-averaged from $t = 1.55$ to 1.65 ms. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 34$ and σ includes fitting uncertainty and the standard deviation of the data.

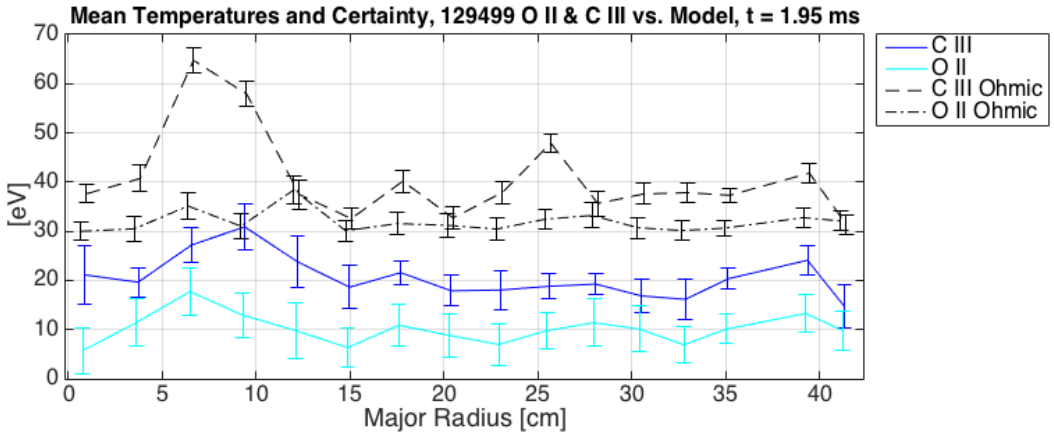

Figure 5.23: Temperatures from O II and C III emission from shot 129499 and theoretical temperatures. All data sets are time-averaged from $t = 1.9$ to 2.0 ms. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 33$ and σ includes fitting uncertainty and the standard deviation of the data.

spans for analysis of each of the three shots are independently selected so that the current amplification is near 3 and the toroidal current is at least ∼80 kA. The full, selected data are shown in appendix A.2. The mean velocity is subtracted since we are only concerned with fluctuating velocities. The data from each chord is then decomposed using Fourier analysis and reconstructed with only the injector frequency, which is typically teh largest amplitude mode. Under the assumption that velocity oscillations are due to plasma motion, the sinusoidal velocities are time-integrated to yield "displacement". The results are plotted in figures 5.24 and 5.25. Note that the oscillations are in phase from $R \simeq 18$ to 35 cm and shift by $\sim 1/8^{\text{th}}$ of an injector cycle farther outboard. For $R < 18$ cm the displacement amplitudes decrease and the phases shift significantly. The maximum positive displacement (i.e.: away from the detector) of ~ 2.5 cm is approximately aligned with the maximum X injector current.

5.10 Theoretical Calculations of Separatrix Position

Since a transition is observed at $R \simeq 18$ cm, this result can be compared with thoeretical calculations of the separatrix between the spheromak and injector-linking plasma. The IDCD step-λ profile is calculated by Chris Hansen for current-amplification of 3 ± 1 and injector-λ of 16 ± 1 m^{-1}. The resulting inboard separatrix is at $R = 9.3 \pm 0.5$ cm.

The separatrix position can also be estimated from Taylor minimum energy theory. "Taylor states" are computed in the HIT-SI geometry for each injector and the spheromak separately, then added together to form "composite Taylor states" [20, 45]. Composite Taylor states can not achieve the experimental injector-λ of 16 ± 1, so their predictions have uncertainty bounded between

$$\text{Flux Amplification} \simeq \text{Current Amplification} \tag{5.5}$$

and

$$\text{Flux Amplification} \simeq \lambda\text{-ratio} \times \text{Current Amplification}. \tag{5.6}$$

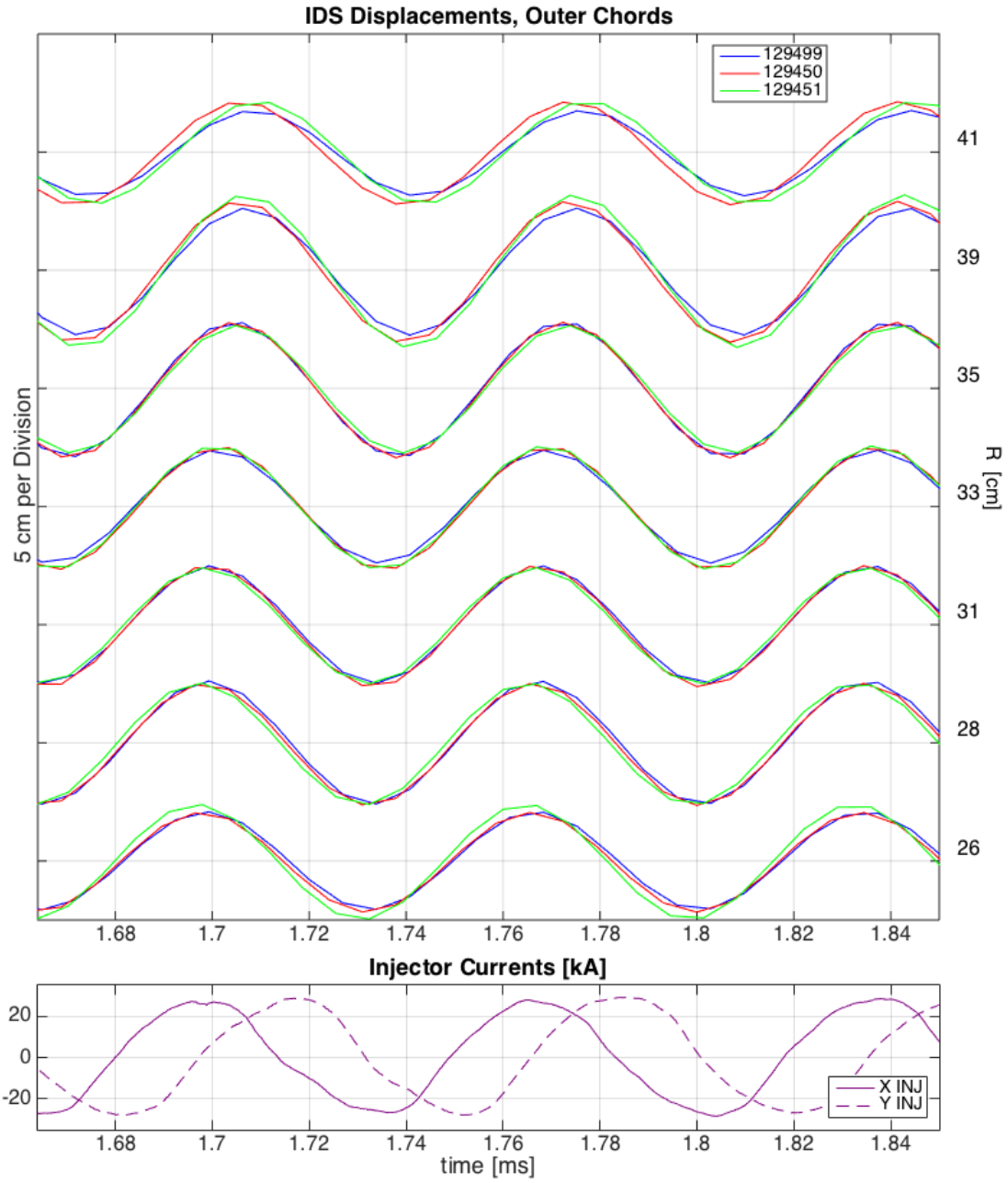

Figure 5.24: Strongest Fourier components of velocities integrated to give displacements. Data from each shot are selected when current amplification is near 3 and reconstructed with only the largest mode. For shots 129450 and 129451 the time span is 1.4 to 2.0 ms and for shot 129499 the time span 1.5 to 2.0 ms.

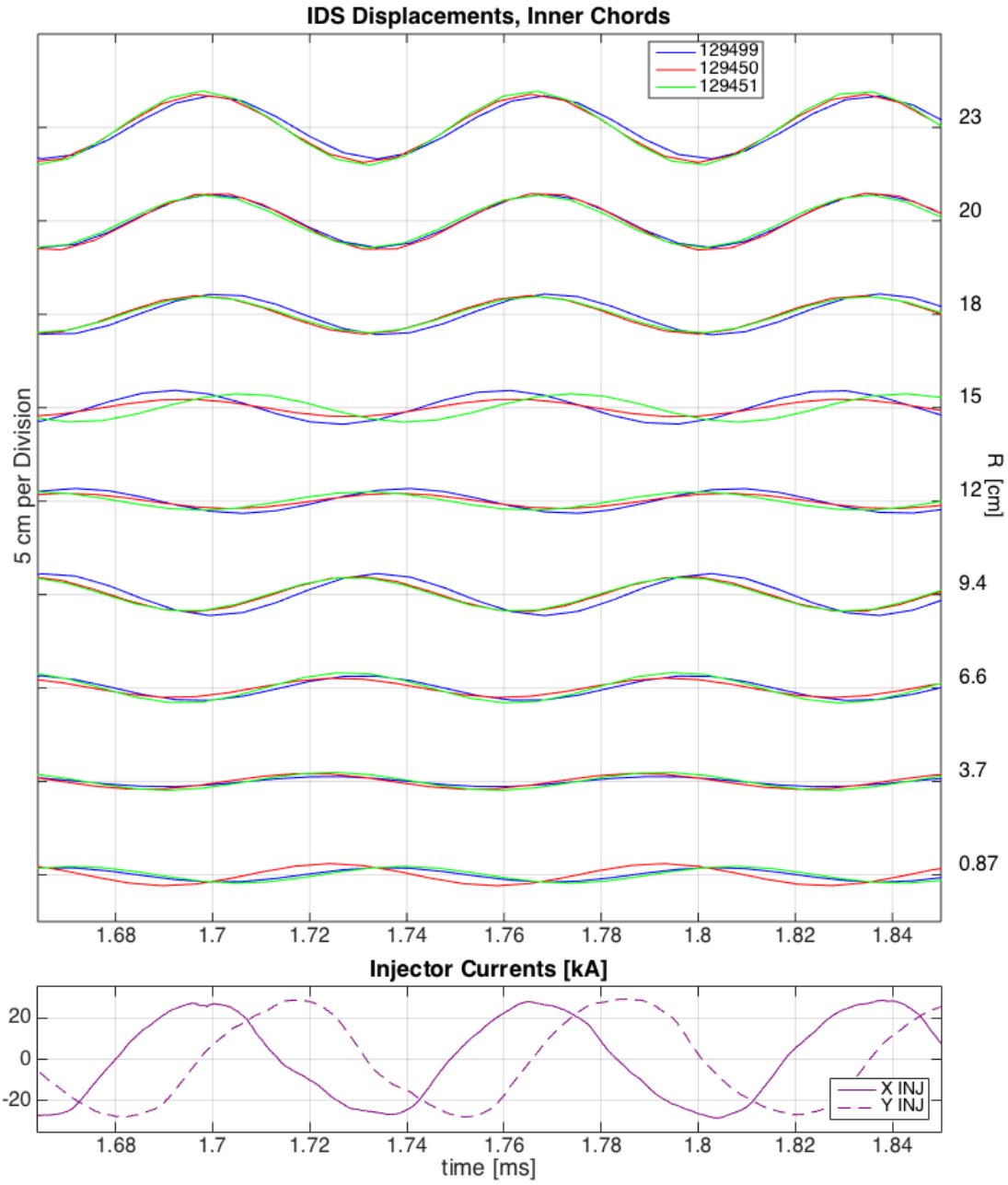

Figure 5.25: Strongest Fourier components of velocities integrated to give displacements. Data from each shot are selected when current amplification is near 3 and reconstructed with only the largest mode. For shots 129450 and 129451 the time span is 1.4 to 2.0 ms and for shot 129499 the time span 1.5 to 2.0 ms.

Puncture plots of the bounding cases are shown in figure 5.26. The inboard separatrix is estimated to be between $R = 21$ and 25 cm. The many punctures just outside the separatrix indicate that the stochastic region within a few centimeters of the separatrix may be nearly closed and strongly coupled to the separate plasma.

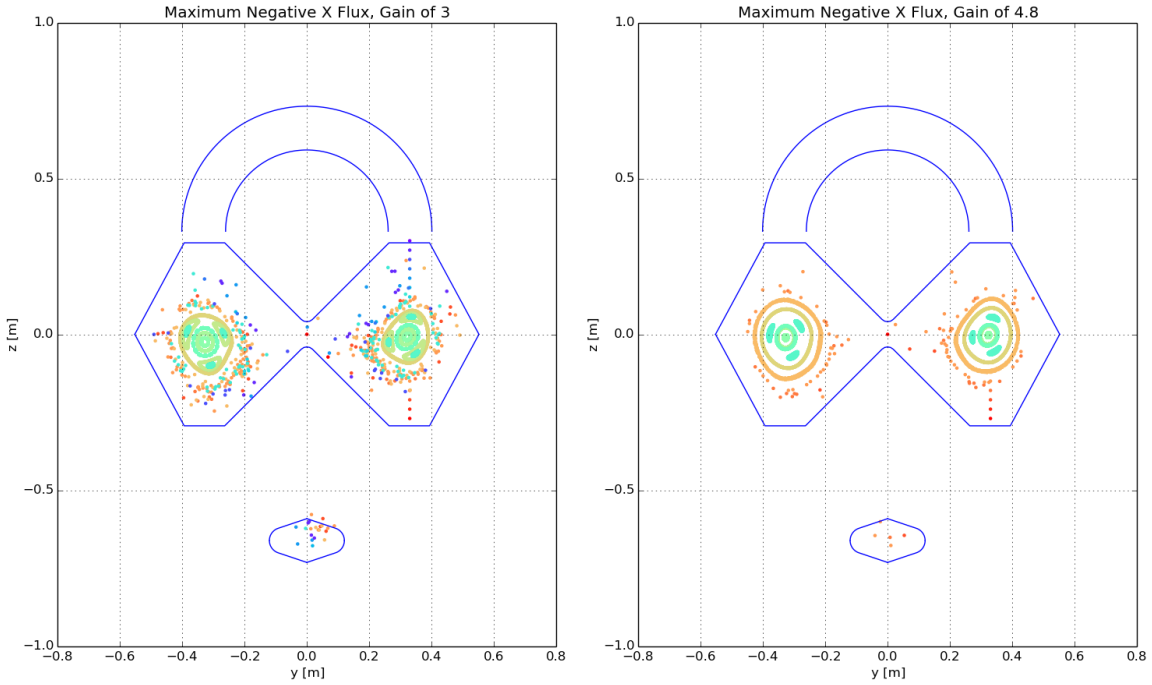

Figure 5.26: Two puncture plots in the Y-Z plane of composite Taylor states for HIT-SI. Flux gains of 3 and 4.8 are bounds for comparison with experimental measurements. The injector phase corresponding to maximum X flux is chosen because the closed-flux region is more poloidally symmetric than at other phases. Figures courtesy of Tom Benedett and Chris Hansen.

5.11 Discussion

A diagram of the spatial relationship between the current and plasma motion is shown in figure 5.27, right. The injector current has been previously shown to "follow" the toroidal current in the toroidal direction to one side of the geometric axis [55], hence the downward arc of the purple arrow. It follows that the injector currents will align with the spheromak

field in the poloidal direction as well. A poloidal transit of the X injector current around the spheromak is illustrated in figure 5.27, left. The injector current and flux impose opposite forces on the spheromak. The flux wrapping around the outside of the spheromak will push it toward the geometric axis due to magnetic pressure, but the injector current parallel to the edge spheromak current yields an attractive force pulling the spheromak outward. As illustrated by the blue arrows, the spheromak is observed to be displaced to the right, away from the detector, when the X injector current is at its maximum value. Thus, the attractive force due to the currents dominates over the pressure from the flux.

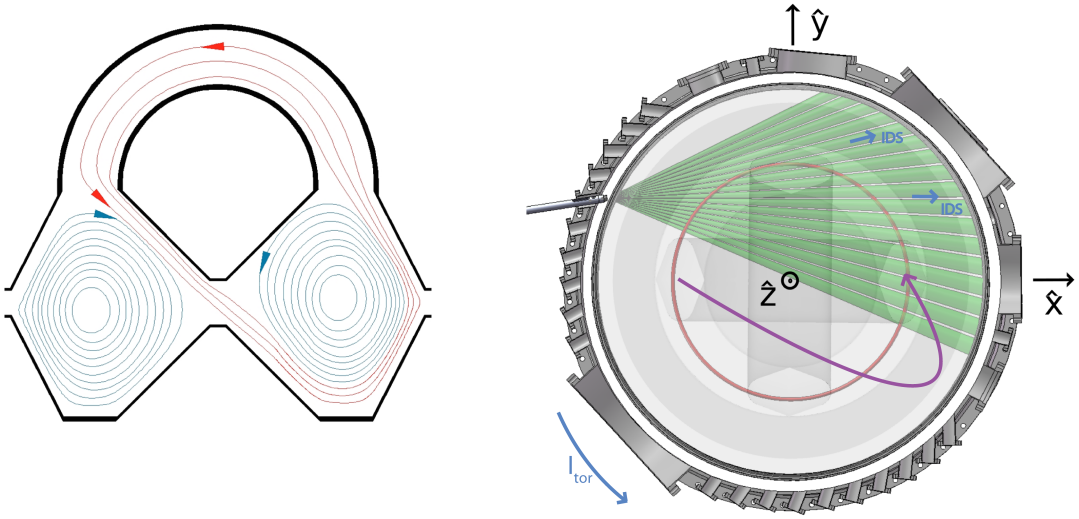

Figure 5.27: Poloidal (left) and toroidal (right) illustrations of the X injector current with respect to the spheromak and IDS chords. In the figure at right, the purple arrow represents the approximate path of the X injector current and flux. The blue arrows indicate that the plasma is displaced to the right. In the left figure the X injector flux is red, shown making a poloidal transit around the spheromak (blue). Poloidal cross section adapted from reference [45].

Since the velocities are in phase and a similar amplitude, the spheromak plasma moves as a coherent object. Coherent motion indicates stability because every flux surface inside the separatrix moves together and responds to the perturbations with a restoring force. Additionally, the amplitudes of magnetic Fourier modes support the claim of stability. The

dominant $n = 1$ mode is almost entirely imposed by the injectors and modes 2 through 4 grow in proportion to the equilibrium with no signs of instability growth. A separate spheromak plasma is predicted by IDCD as a consequence of the step-λ profile and closed flux is predicted by Taylor theory. The observed boundary of $R \simeq 16-18$ cm is between the IDCD prediction of 9 cm and the Taylor closed flux boundary of 21 to 25 cm. The assertion that electron flow is "locked" inside the separatrix of HIT-SI is based on measurements of $\delta B_\perp$ from the internal magnetic probe [33]. If the perturbations just inside the separatrix are not fully distorted there may be a transition region with finite λ-gradient.

The cause of the different temperatures of O II and C III is unknown, but could be due to different spatial distributions. The comparison of measured temperatures with the model of perfect confinement suggests increasing losses throughout the discharge. The measured temperatures show increasing trends over approximately the first half of the discharge and then decreasing during the period of relatively constant toroidal current. The model temperatures continue to rise, diverging significantly from the data by the end of the shot. The reversal of the temperature trend is consistent with the rise and fall of the C IV / C III ratio.

Very little flow is observed for small impact parameter chords and the velocities at larger impact parameters are dominated by oscillations at the injector frequency. There is no significant difference in mean velocity between positive and negative toroidal current shots, and therefore no significant toroidal flow associated with toroidal current. Another consistent trend is that O II velocities are slightly lower in magnitude than C III. The differences could be due to the different charge-to-mass ratio of the impurities, 1/6 vs. 1/16, if flows are driven by electric fields. The impurity velocities are expected to represent the bulk deuterium velocities because the characteristic injector time, ω_{inj}^{-1}, is ~ 11 μs but the ion-impurity collision time is estimated to be less than 100 ns [26].

HIT-SI is inherently dynamic and 3D due to the time-varying injector fluxes, so IDS data cannot be Abel inverted. Simulations can yield more detail by first validating chord-averaged temperatures, intensities, and velocities.

Chapter 6

COMPARISON TO SIMULATIONS

Since chord-averaged data from the IDS cannot be converted to radial profiles, the best method of comparison is to replicate IDS data from the simulations using synthetic diagnostics. Chord-averaged velocities, temperatures, and intensities from the IDS, NIMROD and PSI-TET codes are compared in the following sections.

6.1 Simulation Codes and Synthetic Diagnotics

6.1.1 The NIMROD Code

HIT-SI plasmas are simulated with the NIMROD [46] MHD code. Most plasma parameters match those of the experiment including injector flux and current, density, resistivity, viscosity, and Alfven time. The main exception is electron mass which is enhanced to $1/36^{th}$ of the ion mass for numerical convenience.

The four Hall-MHD equations evolved by NIMROD are the continuity equation, momentum equation, Faraday's law, and the energy equation:

$$\frac{\partial n}{\partial t} + \nabla \cdot \left(n\vec{V} \right) = \nabla \cdot D\nabla n + \nabla \cdot \nabla D_h \nabla^2 n \tag{6.1}$$

$$nm_i \left(\frac{\partial \vec{V}}{\partial t} + \nabla \cdot \left(\vec{V}\vec{V} \right) \right) = \vec{J} \times \vec{B} - \nabla p + \nabla \cdot \Pi \tag{6.2}$$

$$\frac{\partial \vec{B}}{\partial t} = -\nabla \times E + \kappa_{\nabla \cdot B} \nabla \nabla \cdot \vec{B} \tag{6.3}$$

$$\frac{n}{\gamma - 1} \left(\frac{\partial}{\partial t} + \vec{V} \cdot \nabla \right) T = -p_\alpha \nabla \cdot \vec{V} - \Pi : \nabla \vec{V} - \nabla \cdot \vec{q} + \eta J^2 \tag{6.4}$$

Where:

$$\Pi = -nm_i\nu\left(\nabla\vec{V} + \left(\nabla\vec{V}\right)^T - \frac{2}{3}I\nabla\cdot\vec{V}\right) \tag{6.5}$$

$$E = \eta\vec{J} - \vec{V}\times\vec{B} + \frac{\vec{J}\times\vec{B} - \nabla p_e}{ne} + \frac{m_e}{ne^2}\frac{\partial\vec{J}}{\partial t} \tag{6.6}$$

$$\vec{q} = n\left(\chi_{\parallel}\hat{b}\hat{b} + \chi_{\perp}\left(1 - \hat{b}\hat{b}\right)\right)\cdot\nabla T \tag{6.7}$$

$$p = nkT \tag{6.8}$$

Here $\vec{V}$ is the center of mass velocity, $\vec{B}$ is the magnetic field, m_e and m_i are the electron and ion mass, respectively, e is the electron charge, n is the particle density, and η is the electrical resistivity. The diffusivity, D, is set at 1,000 and the hyperdiffusivity, D_h, is 0.1. Viscous dissipation, $\nu = 260$, is calculated [3] from parallel viscosity according to Braginskii. The NIMROD code uses a semi-implicit time advance. The second term in equation 6.3 is used to diffuse $\nabla\cdot\vec{B}$ errors.

Any computational domain in NIMROD must be toroidally symmetric, which necessitates modeling the helicity injectors as boundary conditions of the confinement volume. The bow tie flux conserver is discretized into 24 $\times$ 24 finite elements of polynomial degree 4 in the poloidal plane and 11 Fourier modes in the toroidal direction. The boundary conditions are $\vec{E}\times\hat{n} = \delta\vec{B}\cdot\hat{n} = 0$ and $\vec{V} = 0$ except where the helicity injector boundary conditions are imposed. The injector magnetic flux is simulated with a time-varying normal magnetic field and the current is induced by a specified voltage drop between injector mouths.

One other detail of note is the simulation of HIT-SI's insulating coating by a highly resistive edge layer in NIMROD. The outer 1 mm of the poloidal computational domain has resistivity 3.7×10^4 times greater than the plasma.

Zero-β simulations of HIT-SI have been validated against experimental data as shown in Cihan Akcay's thesis [3] and publication [4]. Quantitative agreement has been demonstrated

in global quantities such as formation time and toroidal current, as well as surface and internal magnetic fields. The simulations evolving density and temperature shown here are recent work by Kyle Morgan.

6.1.2 The PSI-TET Code

The PSI-TET [20] code is a new, fully 3D, non-linear, reduced MHD code that has been developed in conjunction with the PSI-Center. This code uses a tetrahedral spatial discretization that enables the modeling of devices with complex boundaries. Thus, the helicity injectors are fully modeled, unlike the NIMROD code where they are boundary conditions on the axisymmetric volume. This should yield more realistic plasma flow since plasma can flow through the PSI-TET injectors. In addition to modeling the full HIT-SI geometry, the PSI-TET code also utilizes a new boundary condition that captures the effects of the insulating wall coating without a resistive layer. $j_n = 0$ is set as a boundary condition by forcing $\left(\nabla \times \vec{B}\right)_n$ to zero.

6.1.3 Synthetic Diagnostics

In order to compare IDS data with simulation output, synthetic diagnostics were developed. First, output files from the simulation containing all three fluid velocity components, density, and temperature are loaded into the VisIt [10] program. Data are extracted every millimeter along lines equivalent to the IDS chords. A 2D Gaussian is evaluated for every point along the chords with the IDS instrument broadening and *local* density and temperature. The "volume" (i.e.: intensity) of each Gaussian is scaled by n_e^2, a simple assumption which neglects the temperature and density dependent atomic physics of emission. By summing all the functions along each chord an image is generated which is equivalent to the output from the IDS detector. Additionally, since the simulations have higher time resolution than the IDS system, the chord-summed Gaussians can be summed in time as well, imitating the effect of the IDS exposure time. By making a data set which is functionally identical to the "movie" from the IDS detector, the simulation data can be processed with the same analysis codes.

6.1.4 Alignment for Comparison

To show temporal patterns with sufficient detail, a time window of three injector periods is used for the comparison. The simulation timebases are shifted so that the ends of the simulated shots line up with the end of the experimental toroidal current ramp, which coincides with the brightest IDS data (see figure 6.1). The primary reason for this is that the IDS data are dimmer, resulting in higher error, when the toroidal current is of comparable amplitude to the simulated currents. This can be partially justified because the current amplifications at the chosen time of alignment are similar; note the simulated injector currents are lower than the experimental injector current. The injector current phases are aligned to within $\sim$1 μs.

The primary difference between the experimental and simulated toroidal currents is the second derivative. The rate at which the simulations build up current is accelerating, whereas the experimental toroidal current is saturating. As mentioned in section 5.1, the experimental electron temperature is probably declining due to losses not modeled in the simulations. The only loss mechanism modeled is conduction to the boundary which is held at a fixed, low temperature.

6.2 Velocity Comparison

Chord and time-averaged velocities from the simulations are plotted with C III and O II measurements from shot 129499 in figures 6.2 and 6.3. Time-averaged mean and fluctuating velocity profiles are plotted in figure 6.4. For small impact parameters, i.e.: 3.7, 12, and 20 cm, the velocities are relatively low, of order 5 km/s or less. NIMROD and PSI-TET show as much difference between themselves as with the experimental data. Near the magnetic axis, for $R = 28$ and 35 cm, fluctuations at the injector frequency are seen in all four data sets at approximately the same phase. The C III data are in good agreement with the simulations, but the O II fluctuation amplitudes appear slightly lower.

The most prominent point of disagreement is seen on the outermost chord, at $R = 41$ cm. The simulations, especially PSI-TET, show much stronger fluctuations than observed and an offset of $\sim$7 – 8 km/s.

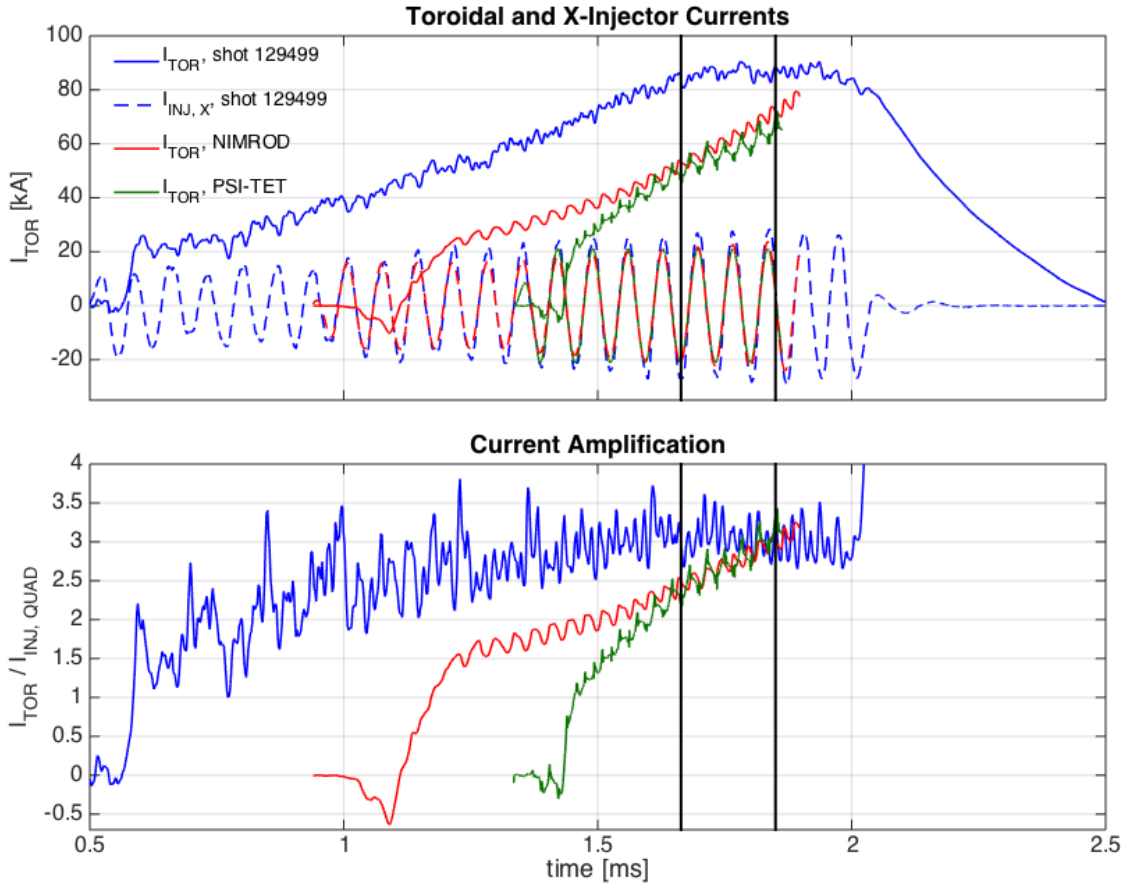

Figure 6.1: Toroidal and X injector currents plotted from HIT-SI shot 129499, NIMROD, and PSI-TET (top). Current amplification (toroidal current divided by injector currents added in quadrature) is plotted below. The time window for comparison is between the black, vertical bars (HIT-SI $t = 1.664 - 1.850$ ms).

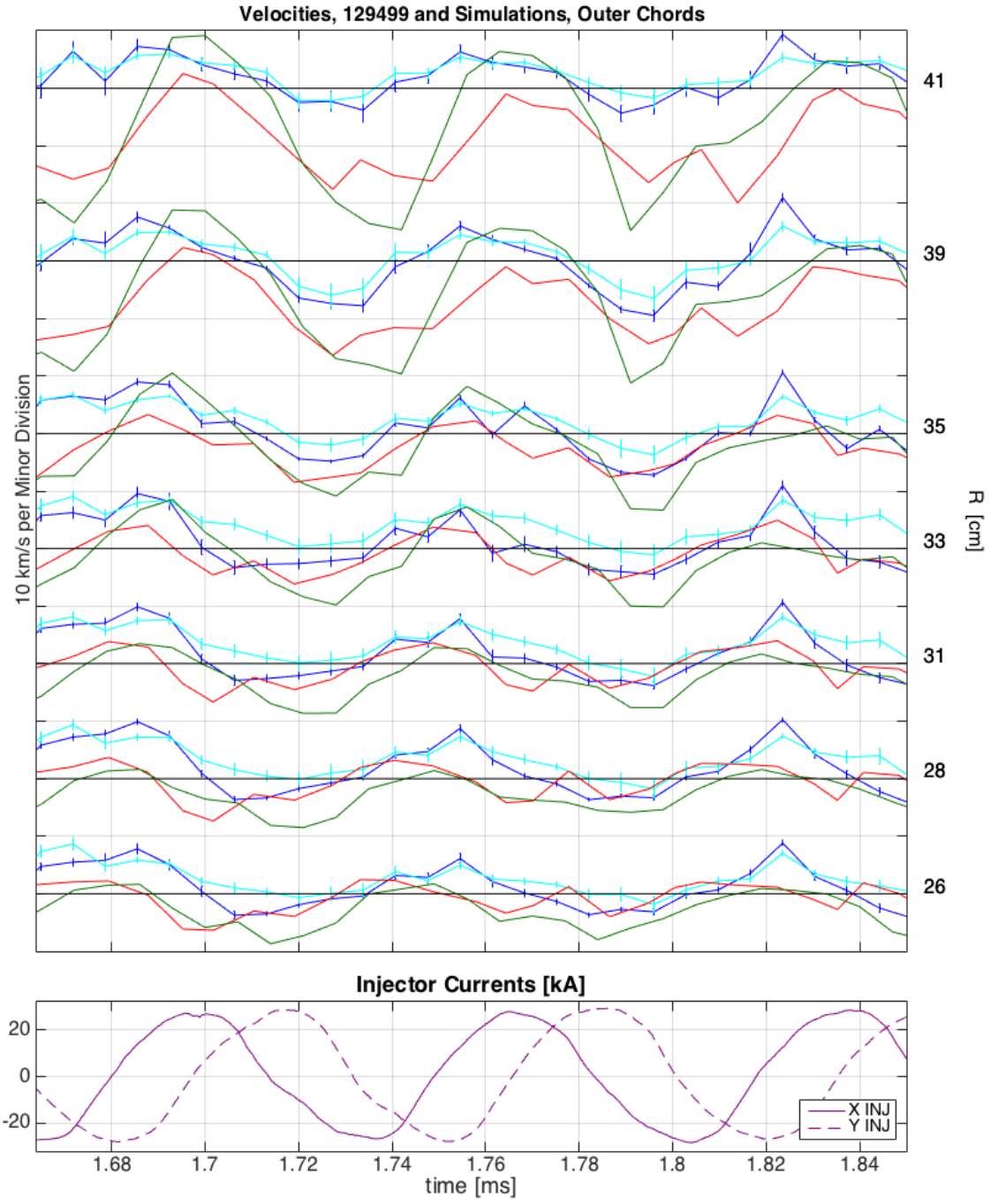

Figure 6.2: IDS O II (cyan) and C III (blue), NIMROD (red), and PSI-TET (green) velocities for outer chords. Error bars on experimental data represent fitting uncertainty. The solid, horizontal lines mark zero-velocity for each chord. Injector currents shown are from shot 129499.

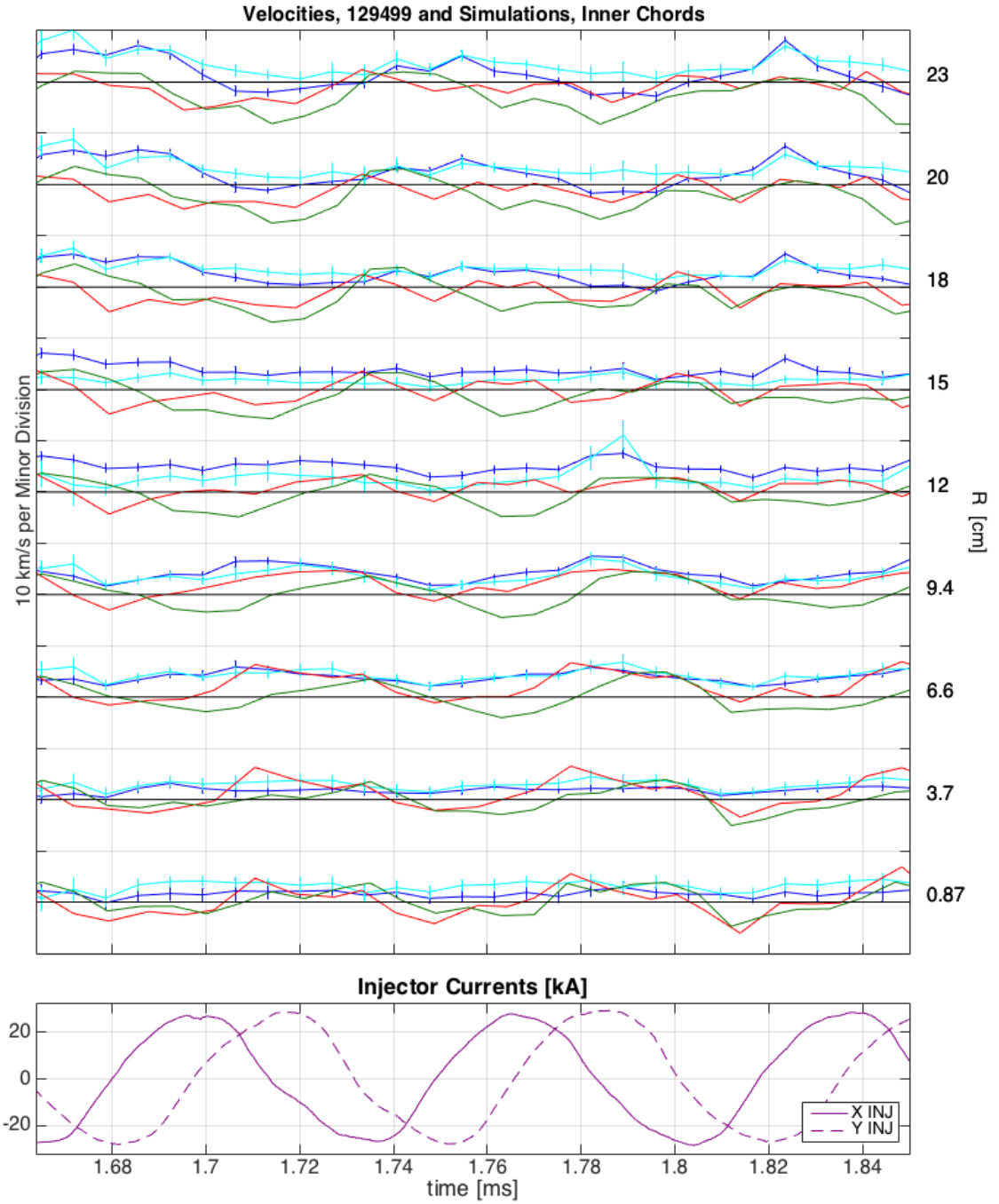

Figure 6.3: IDS O II (cyan) and C III (blue), NIMROD (red), and PSI-TET (green) velocities for inner chords. Error bars on experimental data represent fitting uncertainty. The solid, horizontal lines mark zero-velocity for each chord. Injector currents shown are from shot 129499.

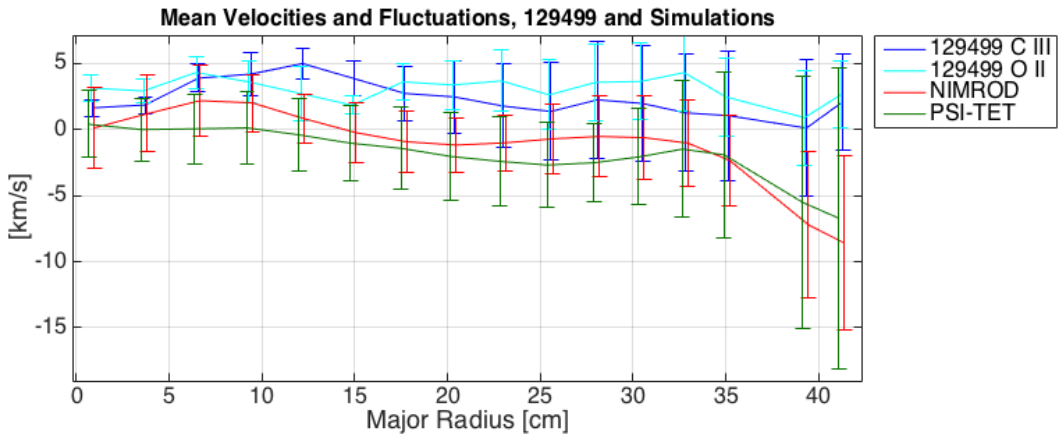

Figure 6.4: IDS O II and C III, NIMROD, and PSI-TET mean and fluctuating velocity profiles for all toroidal midplane chords. Velocities are averaged over the time window shown in figures 6.2 and 6.3 and error bars are the standard deviations of velocities.

6.2.1 Analysis of Velocities from Simulations

In section 5.9 the velocities measured by the IDS were shown to be in phase, demonstrating that the spheromak plasma moves as a rigid object. The same analysis is performed on equivalent synthetic data from NIMROD and PSI-TET and compared with measured data. The velocities from figures 6.2 and 6.3 first have the mean subtracted, then are filtered using Fourier analysis, excluding all but the largest amplitude mode near the injector frequency. The resulting sinusoidal velocities are time-integrated yielding the plasma displacement as a function of time, shown in figures 6.5 and 6.6.

The chord-averaged velocities from simulations do not show the same phase locking and rigid motion as the IDS data. The NIMROD displacements show a slight phase shift with respect to the IDS data from $R \simeq 20$ to 35 cm and a significant shift on the outer two chords. The PSI-TET displacements show a more significant phase shift from $R \simeq 20$ to 35 cm and a similarly large shift from the IDS data on the outer chords.

In order to see the underlying flows that generate the chord-averaged signals, the velocity field from either simulation can be visualized using the VisIt [10] program. The velocity

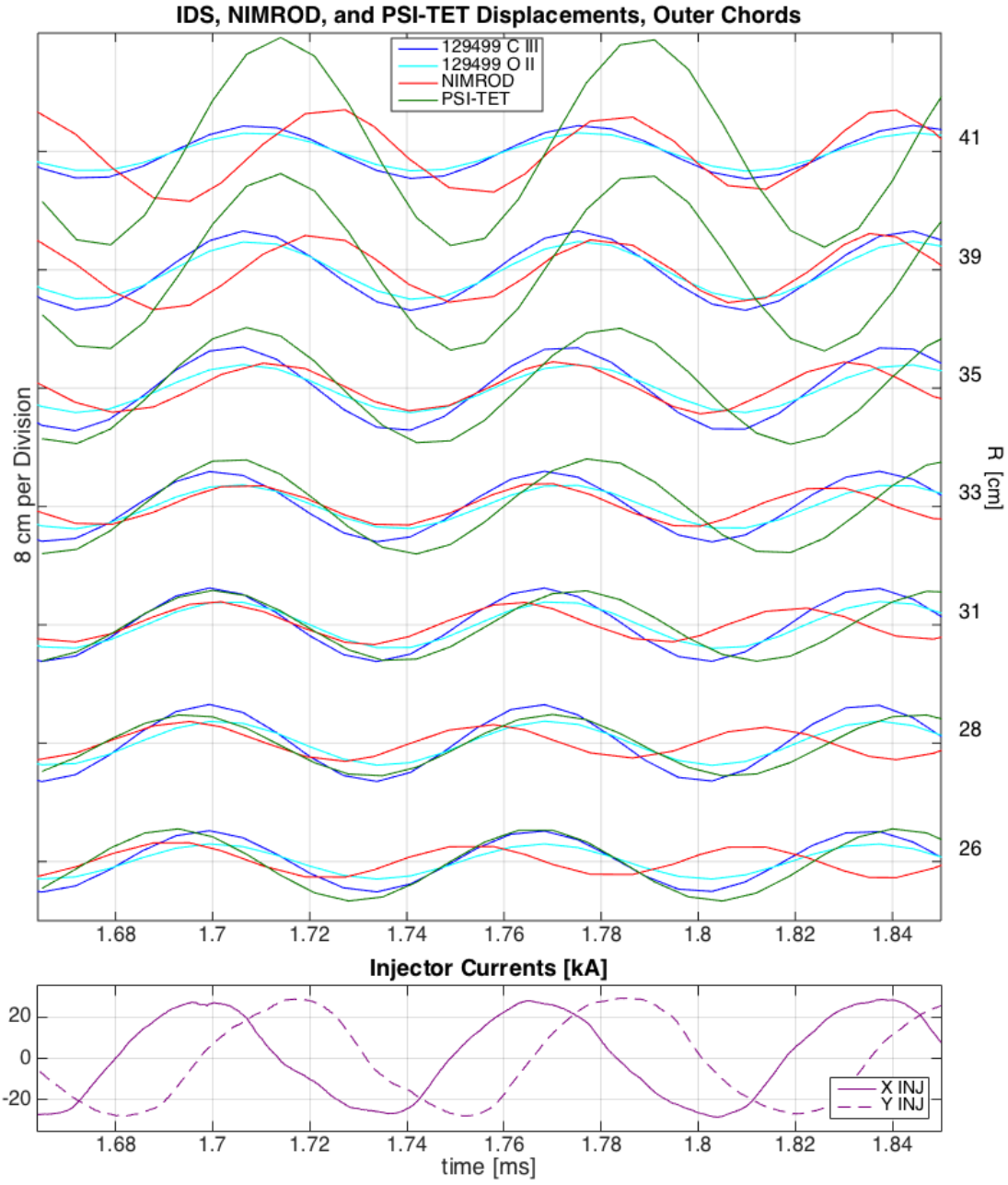

Figure 6.5: Strongest Fourier components of velocities integrated to give displacements. The time span for shot 129499 is 1.5 to 2.0 ms, whereas the standard comparison window is used for the synthetic data: 1.664 to 1.85 ms.

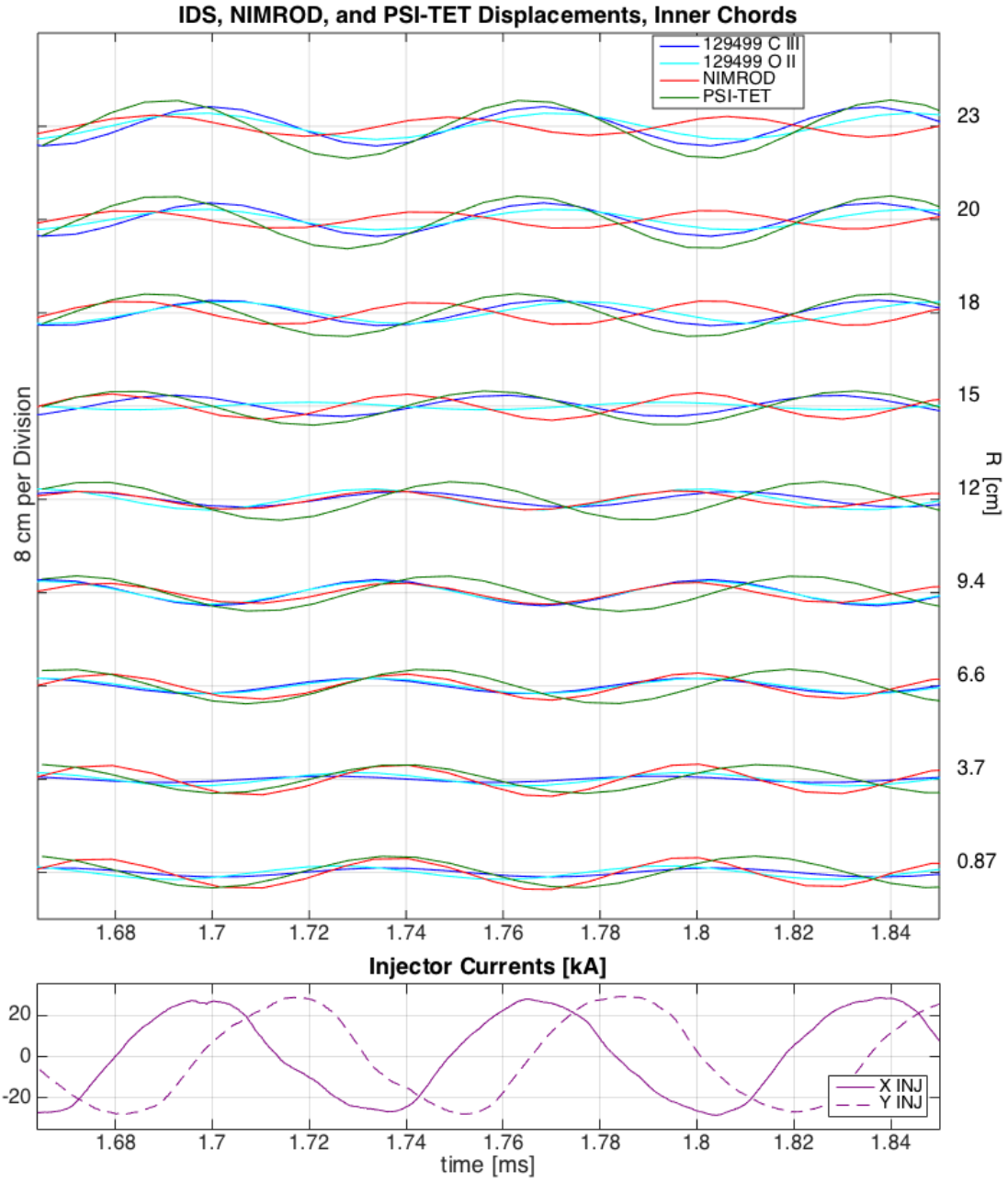

Figure 6.6: Strongest Fourier components of velocities integrated to give displacements. The time span for shot 129499 is 1.5 to 2.0 ms, whereas the standard comparison window is used for the synthetic data: 1.664 to 1.85 ms.

field in the toroidal midplane of the NIMROD simulation is shown in figure 6.7 at one time point. The chord-averaged NIMROD and IDS data are also shown as vectors at the ends of their chords. The flow field is non-axisymmetric and dominated by flows related to the injector. Note that chords passing near the geometric axis, for example, pass through at least 3 different vortical flows. This is more easily seen by plotting the local, chord-aligned velocities along each of the chords. These values are shown in figures 6.8 and 6.9 for the same point in time.

Another observation from this figure is that all the helical flows are *left-handed*. The HIT-SI experiment and NIMROD inject *positive*, i.e.: *right-handed* magnetic helicity. To illustrate this, the current in the midplane of the NIMROD simulation is shown in figure 6.10. Note that all the helical structures in this snapshot are right-handed, as expected. Comparing between figures 6.7 and 6.10, the "locally axial" current and velocity of injector channels are not always aligned.

In order to determine if the left-handed helical flows are a global phenomenon, Kyle Morgan calculated the time evolution of the three components of canonical helicity [57],

$$K_{rel} = m_i^2 \int \vec{u} \cdot (\nabla \times \vec{u})\, dV + 2m_i e \int \left(\vec{u} \cdot \vec{B}\right) dV + e^2 \int \left(\vec{A} \cdot \vec{B}\right) dV. \tag{6.9}$$

The first term is the *kinetic helicity*, the second term is *cross helicity*, and the third term is the magnetic helicity. Note that the derivation in reference [57] is for individual species, but NIMROD is a single-fluid MHD code. A good measure of the vector potential $\vec{A}$ is not available in NIMROD, so the magnetic heliciy could not be calculated directly. The best available approximation is to use the known relationships between currents and Taylor-state helicity content. The simulated plasma is not in a Taylor-state and NIMROD does not capture the helicity content in the injector ducts, but only an order-of-magnitude estimate is needed in this context. Following the method of reference [56] and converting to canonical units, the resulting magnetic helicity is plotted on the right of figure 6.11. The kinetic and cross helicities are plotted on the left.

The negative kinetic helicity is indeed a global phenomenon, but weaker than magnetic helicity by a factor of ~ 500. Helicity in spheromaks is always dominated by the magnetic

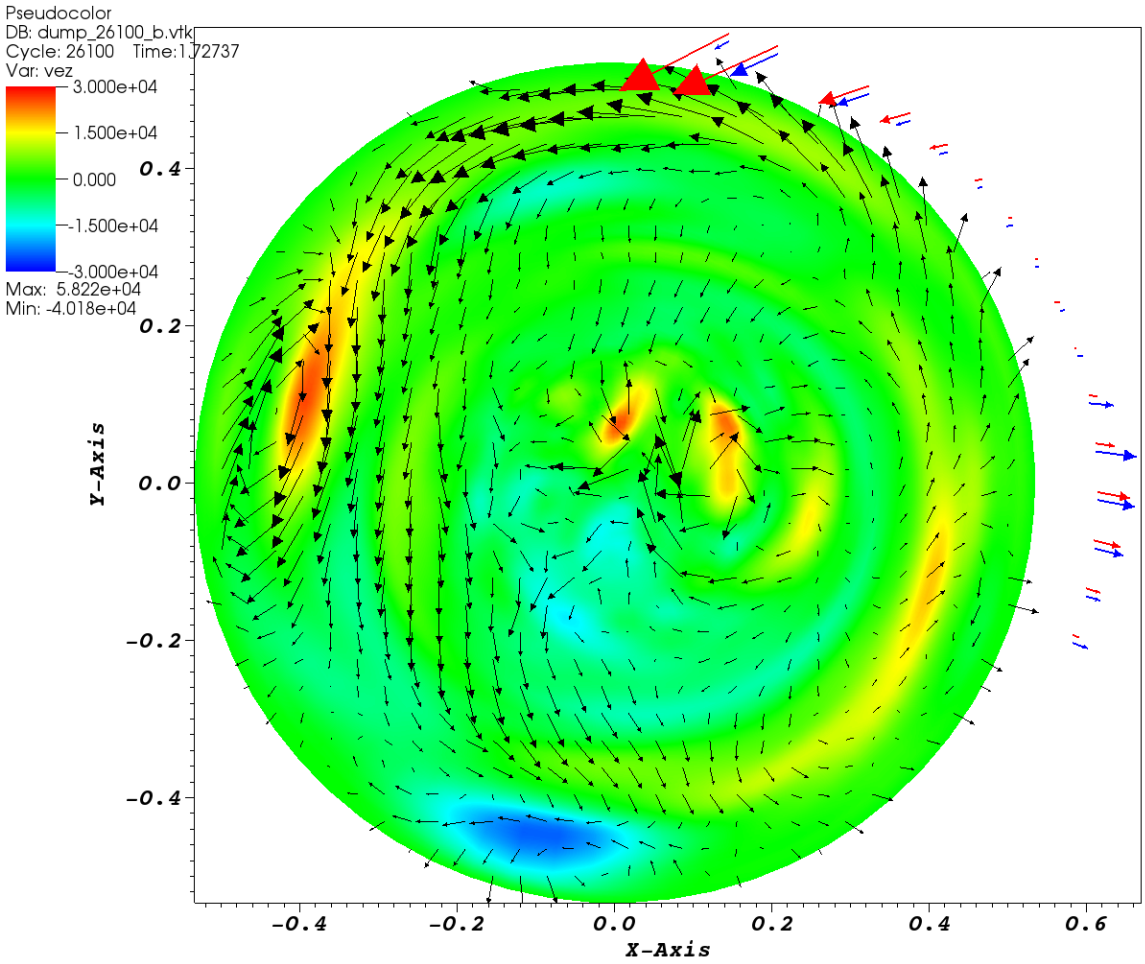

Figure 6.7: Slice through the toroidal midplane of the NIMROD simulation at $t = 1.727$ ms. Black vectors are the in-plane NIMROD velocity and the pseudocolor is the out-of-plane ($\hat{z}$) velocity with colorbar spanning ± 30 km/s. Chord-averaged velocity measurements from the IDS (blue) and the synthetic diagnostic from NIMROD (red) are plotted outside the boundary beyond the ends of their respective chords. The chord-averaged vectors are scaled 3 times larger than the black vectors for visibility. The axes are aligned with those in figure 4.1.

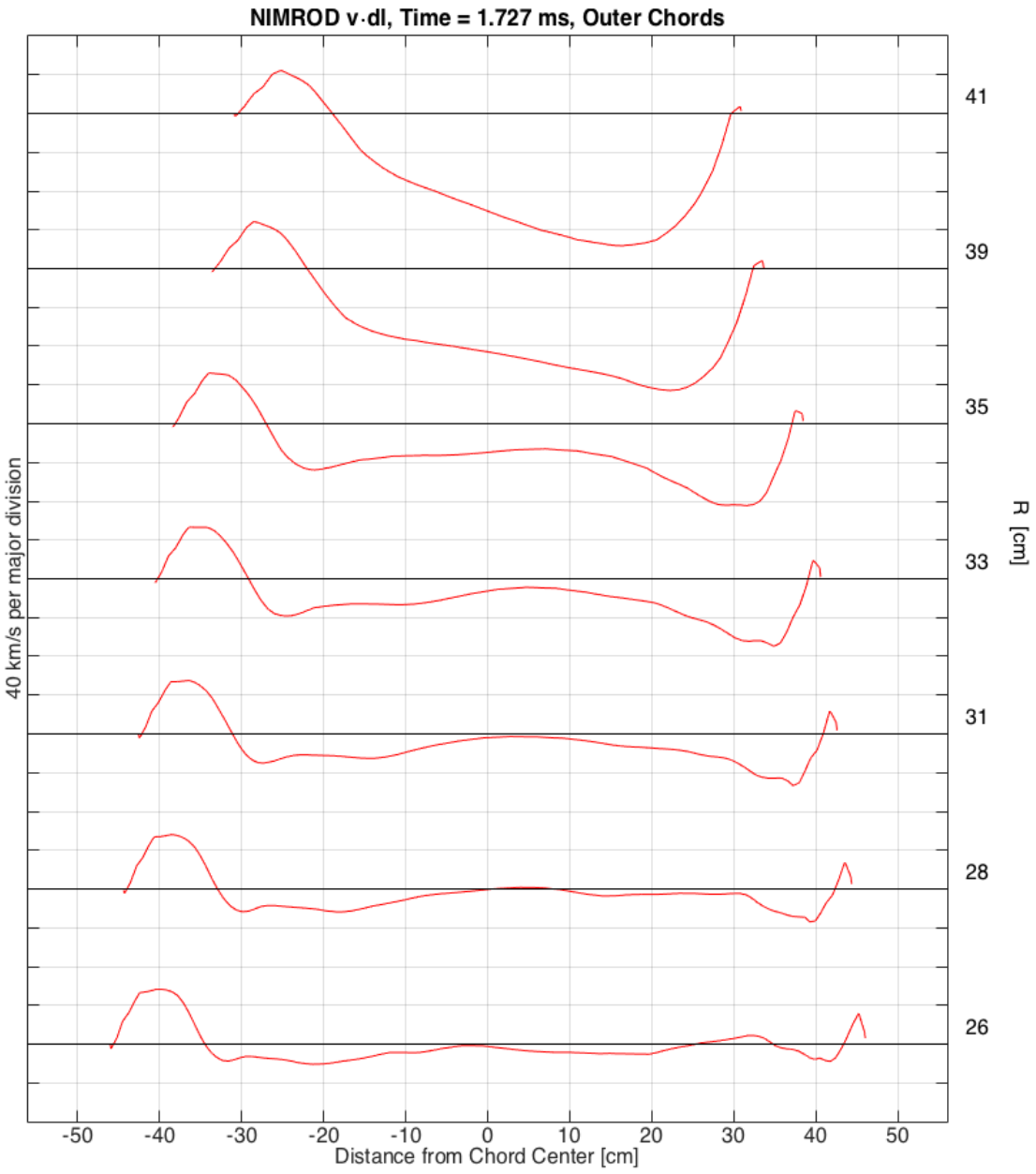

Figure 6.8: Chord-aligned velocities from the NIMROD simulation plotted as a function of distance from the chord center (in order to make the plot symmetric). Positive velocity is defined as pointing away from the detector/chord origin.

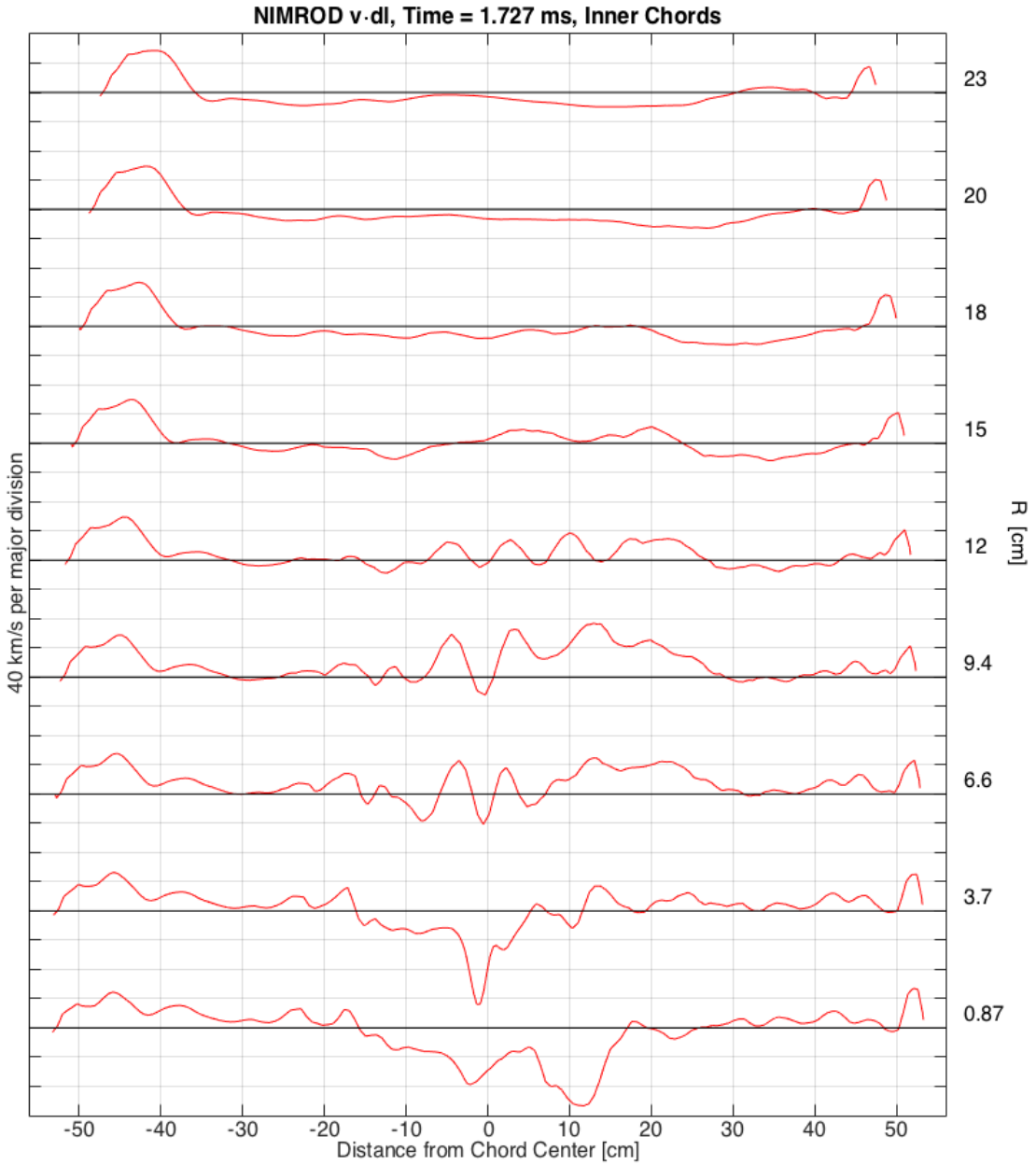

Figure 6.9: Chord-aligned velocities from the NIMROD simulation plotted as a function of distance from the chord center (in order to make the plot symmetric). Positive velocity is defined as pointing away from the detector/chord origin.

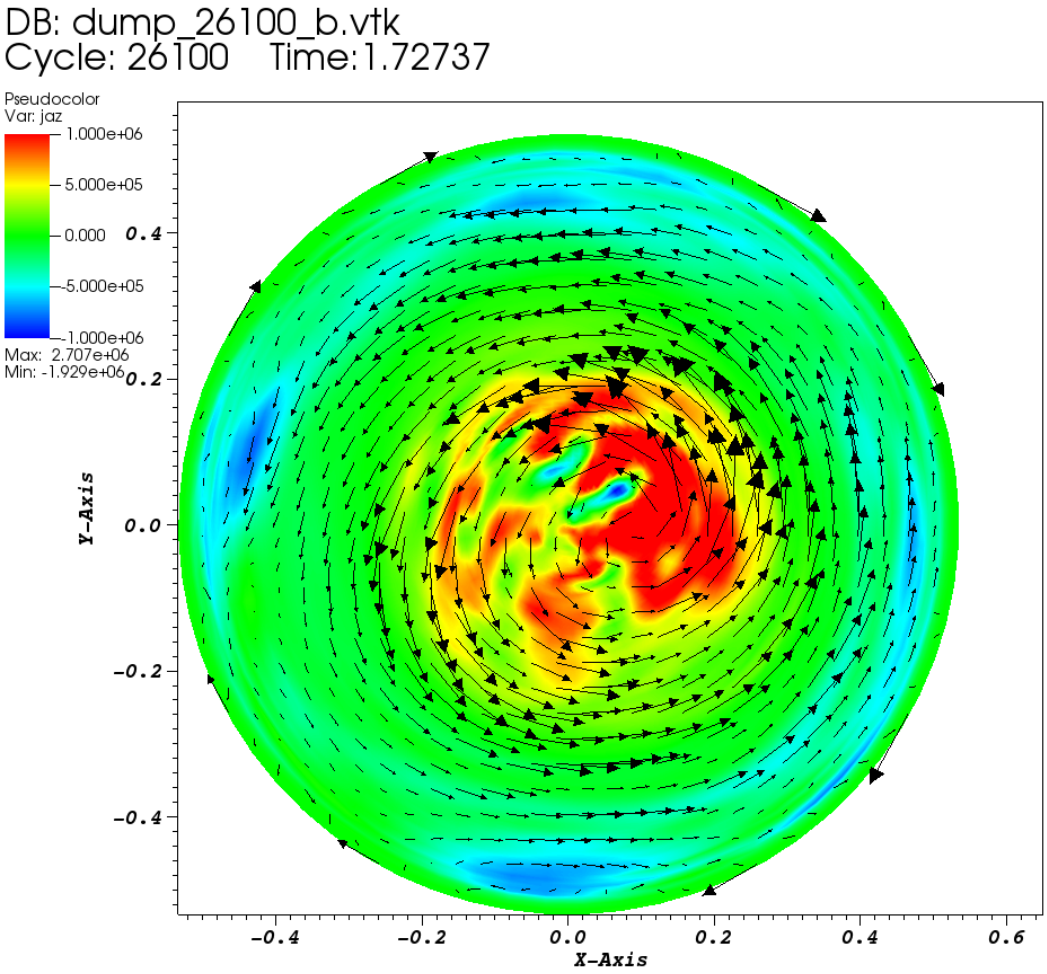

Figure 6.10: Slice through the toroidal midplane of NIMROD simulation at the same time as in figure 6.7. Black arrows represent in-plane current and pseudocolor is out-of-plane ($\hat{z}$) current with colorbar spanning ± 1 MA/m^2.

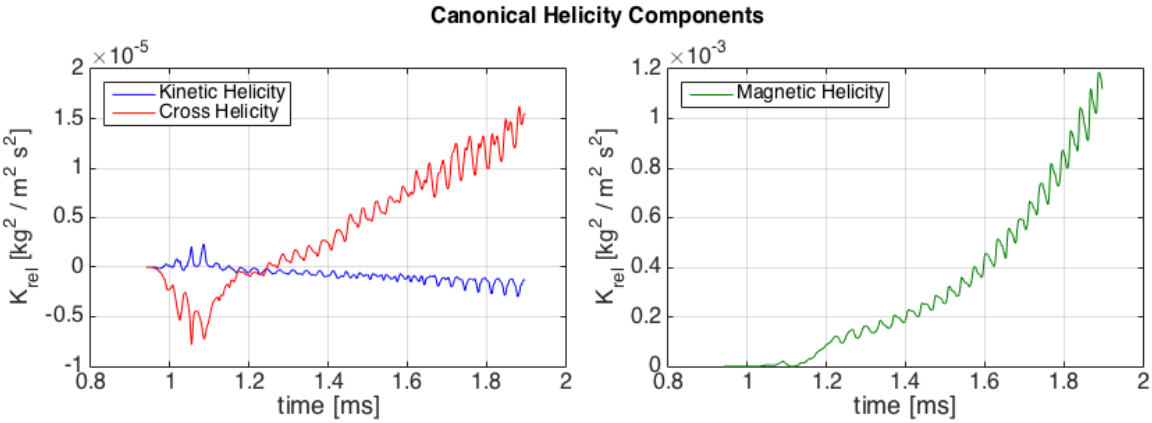

Figure 6.11: Components of canonical helicity in NIMROD simulation, plotted separately due to different scales.

component [35].

6.3 Temperature Comparison

The temperatures of C III, O II, and the simulations are shown in figures 6.12 and 6.13. The time-averaged profiles from all chords are shown in figure 6.14. O II temperatures are approximately half that of C III, as shown in figure 6.14. The C III temperatures lie between the two codes' temperatures for most chords greater than $R \simeq 12$ cm. The main point of disagreement is the larger experimental temperatures and fluctuations at $R = 7$ and 9 cm.

6.3.1 Analysis of Simulated Temperatures

A feature in the simulated temperatures missing from the data is the periodic temperature spikes. For synthetic temperatures, the temporal, differential velocity, and thermal contributions to chord-averaged temperatures can be separated. This is done for the NIMROD temperature data in figures 6.15 and 6.16 and for PSI-TET in figures 6.17 and 6.18. These demonstrate that the temperature spikes from simulations are entirely due to differential velocities. When the differential velocity components of temperature are subtracted from the

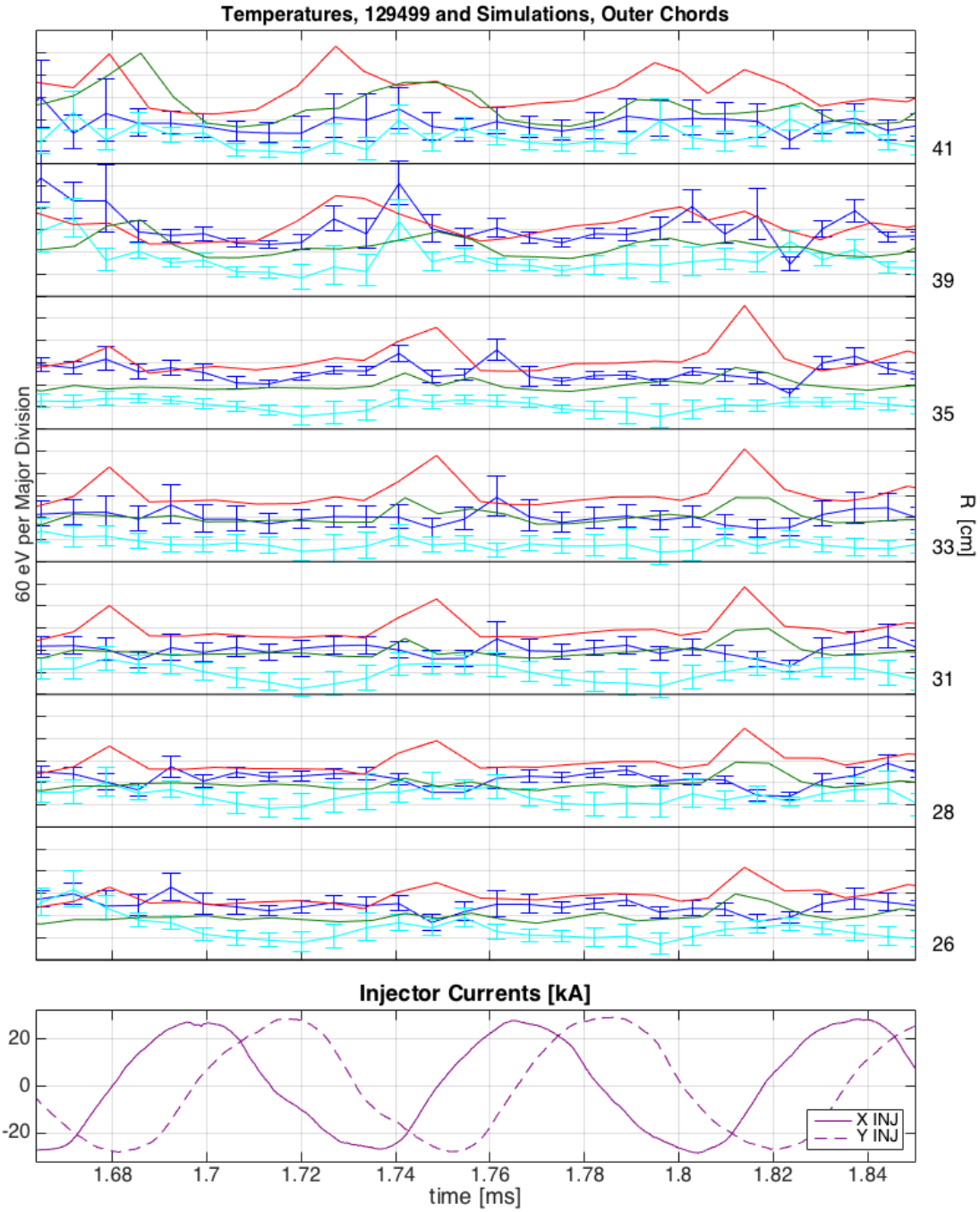

Figure 6.12: IDS O II (cyan) and C III (blue), NIMROD (red), and PSI-TET (green) temperatures for outer chords. The solid, horizontal lines mark zero-temperature for each chord. Injector currents shown are from shot 129499.

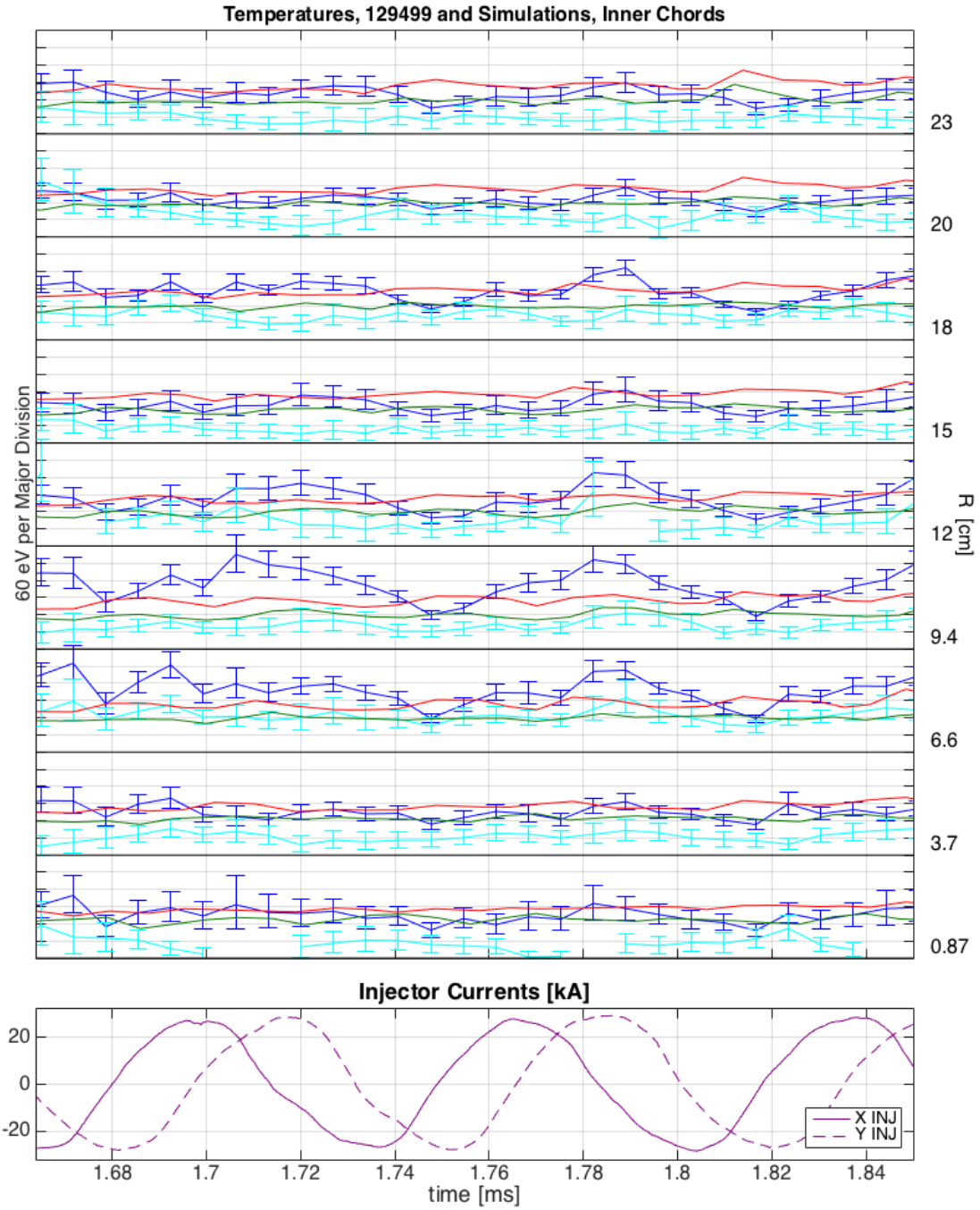

Figure 6.13: IDS O II (cyan) and C III (blue), NIMROD (red), and PSI-TET (green) temperatures for inner chords. The solid, horizontal lines mark zero-temperature for each chord. Injector currents shown are from shot 129499.

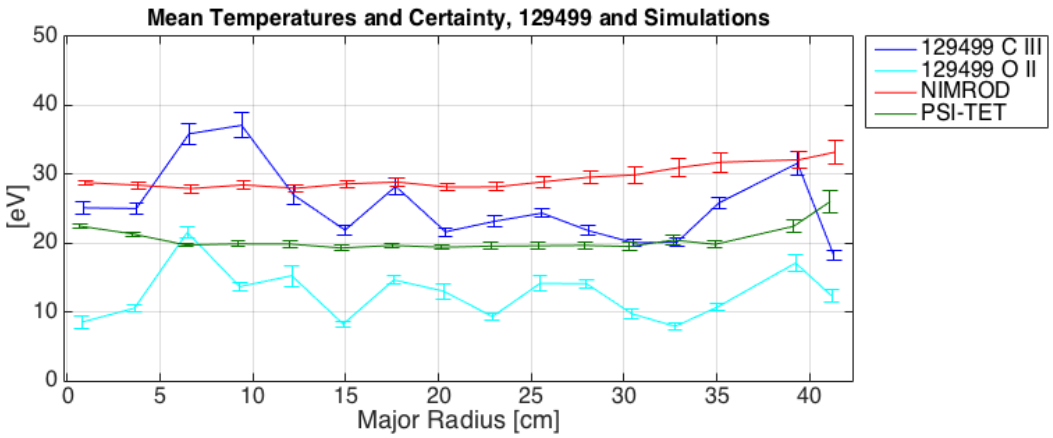

Figure 6.14: IDS O II and C III, NIMROD, and PSI-TET mean temperature profiles and certainties for all toroidal midplane chords as a function of impact parameter. Velocities are averaged over the time window shown in figures 6.12 and 6.13 and error bars are the standard deviations of velocities.

total temperature, much flatter temporal profiles remain. Additionally, these plots demonstrate that the time-integration is not significantly enhancing the temperature. This means the plasma in the simulation is not *accelerating* enough on the timescale of the exposure to cause different velocities at different times to broaden the "spectral line".

The time-averged components of the synthetic temperatures from NIMROD are plotted in figure 6.19 and from PSI-TET in figure 6.20. This shows that the increased synthetic temperatures from chords outboard of $R \simeq 23$ cm are entirely due to increased differential velocities— the thermal temperature is very flat, but decreases slightly. These also show that differential velocities enhance the temperature on most chords by approximately 5 eV. These plots also confirm that time-integration has a negligible effect on temperature since the "NO Time Avg." lines are nearly identical to time-averaged (labeled "Temp. Broadened").

Figures 6.19 and 6.20 show that the "velocity-subtracted" (i.e.: thermal temperature) chord-averaged profiles are flat: 24 to 25.5 eV for NIMROD and 16 eV for PSI-TET. The radial (non-chord-averaged) temperature, however, is peaked toward the magnetic axis. The chord-averaged, thermal temperature profile from NIMROD is reproduced in figure 6.21

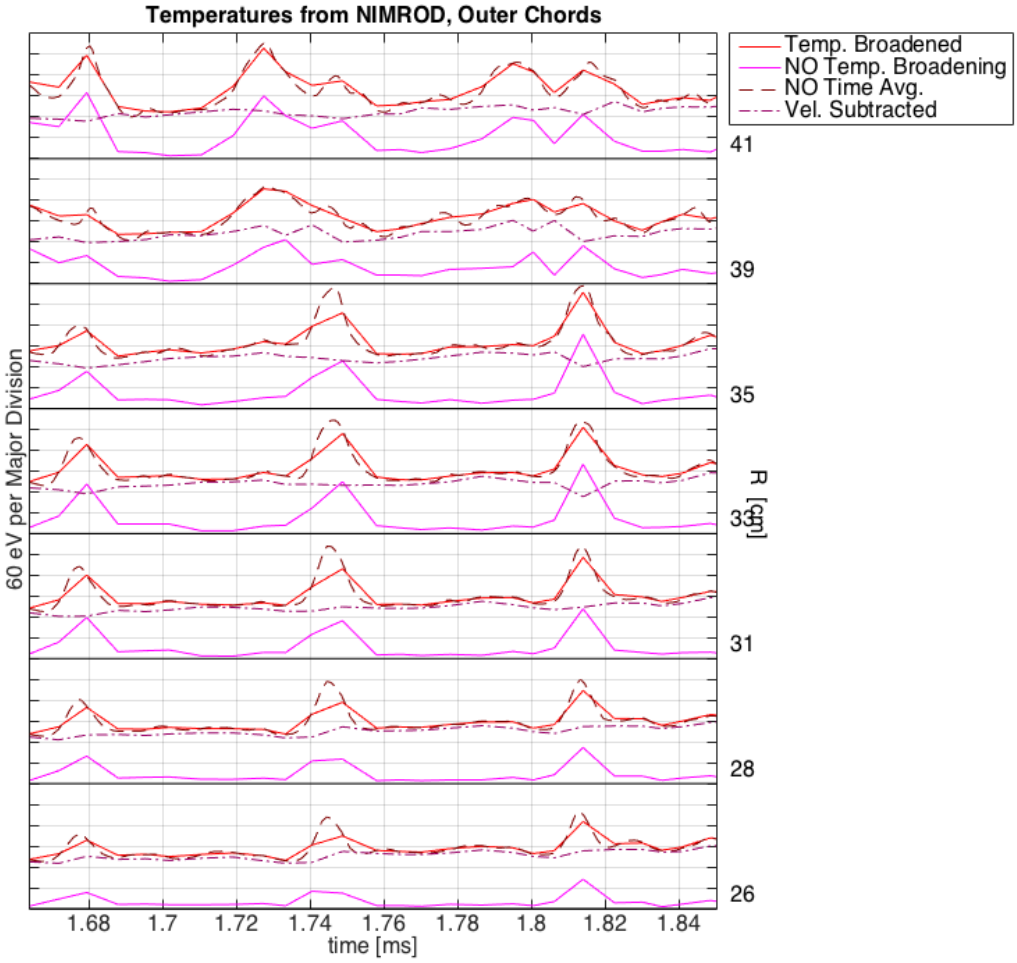

Figure 6.15: Individual components of the NIMROD temperature as calculated through the synthetic diagnostic. The red line, copied from figure 6.12, includes temporal and spatial integration. The dashed, dark red line is not time-integrated; it is calculated from individual output files, i.e.: "snapshots" every ~ 1 μs. The pink line neglects the *thermal* component of temperature, thus this temperature is entirely due to different $\vec{v} \cdot \vec{dl}$ values along the chords. The dashed, purple line is the pink line subtracted from the red line, yielding only the thermal component of temperature.

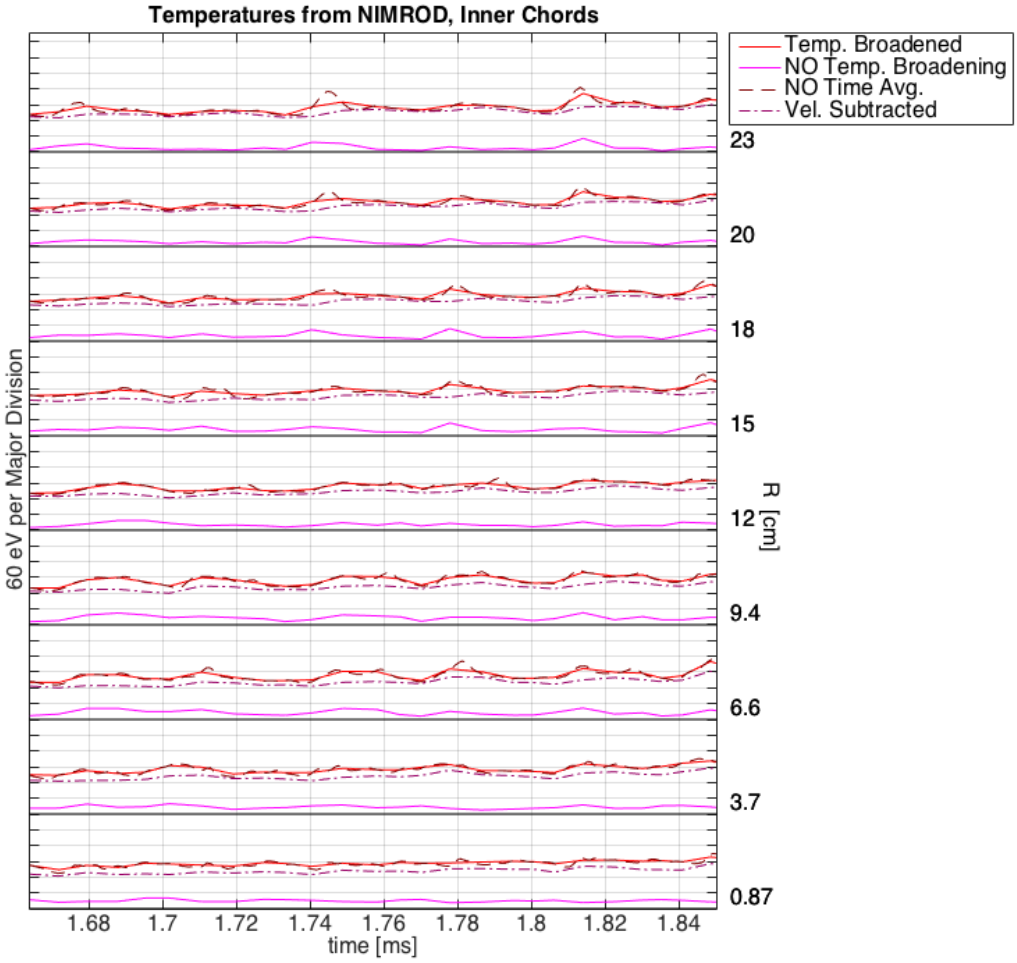

Figure 6.16: Individual components of the NIMROD temperature as calculated through the synthetic diagnostic. The red line, copied from figure 6.13, includes temporal and spatial integration. The dashed, dark red line is not time-integrated; it is calculated from individual output files, i.e.: "snapshots" every ~ 1 μs. The pink line neglects the *thermal* component of temperature, thus this temperature is entirely due to different $\vec{v} \cdot \vec{dl}$ values along the chords. The dashed, purple line is the pink line subtracted from the red line, yielding only the thermal component of temperature.

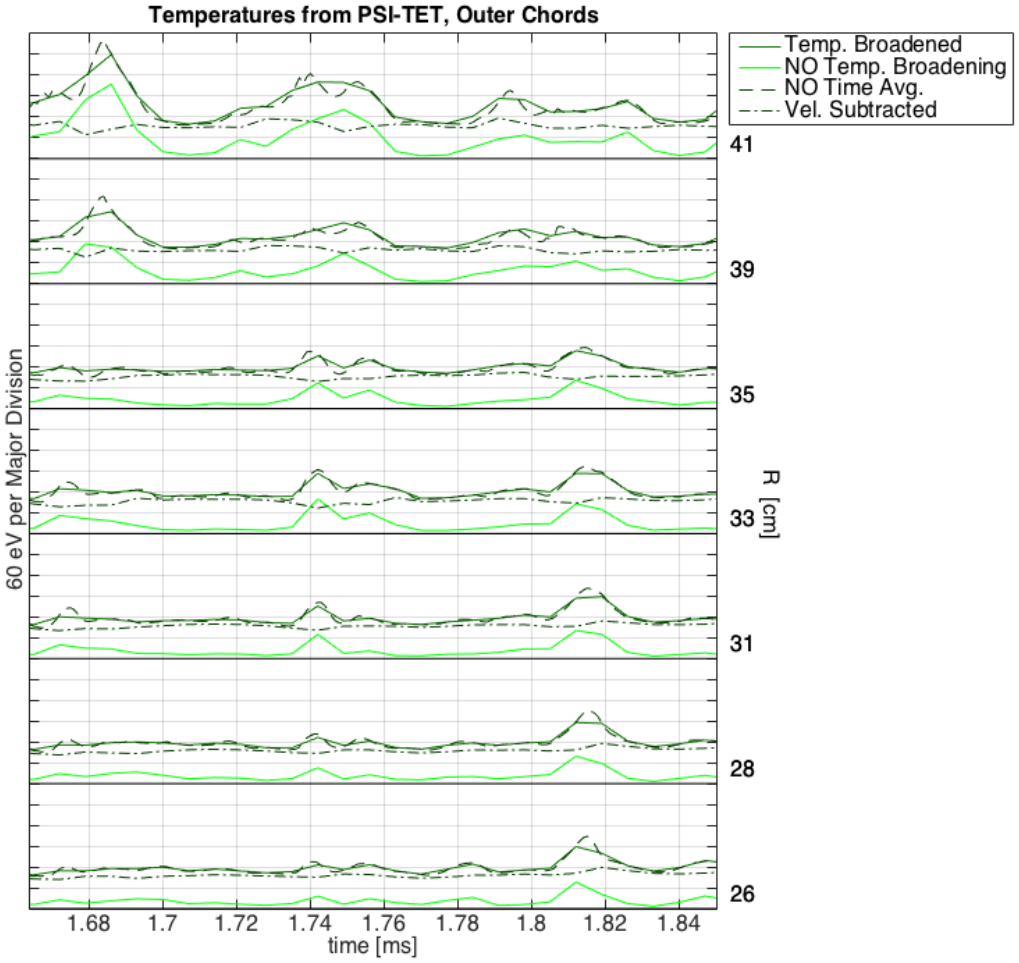

Figure 6.17: Individual components of the PSI-TET temperature as calculated through the synthetic diagnostic. The forest green line, copied from figure 6.12, includes temporal and spatial integration. The dashed, dark green line is not time-integrated; it is calculated from individual output files, i.e.: "snapshots" every ~ 1 μs. The neon green line neglects the *thermal* component of temperature, thus this temperature is entirely due to different $\vec{v} \cdot \vec{dl}$ values along the chords. The dot-dashed line is the neon line subtracted from the forest green line, yielding only the thermal component of temperature.

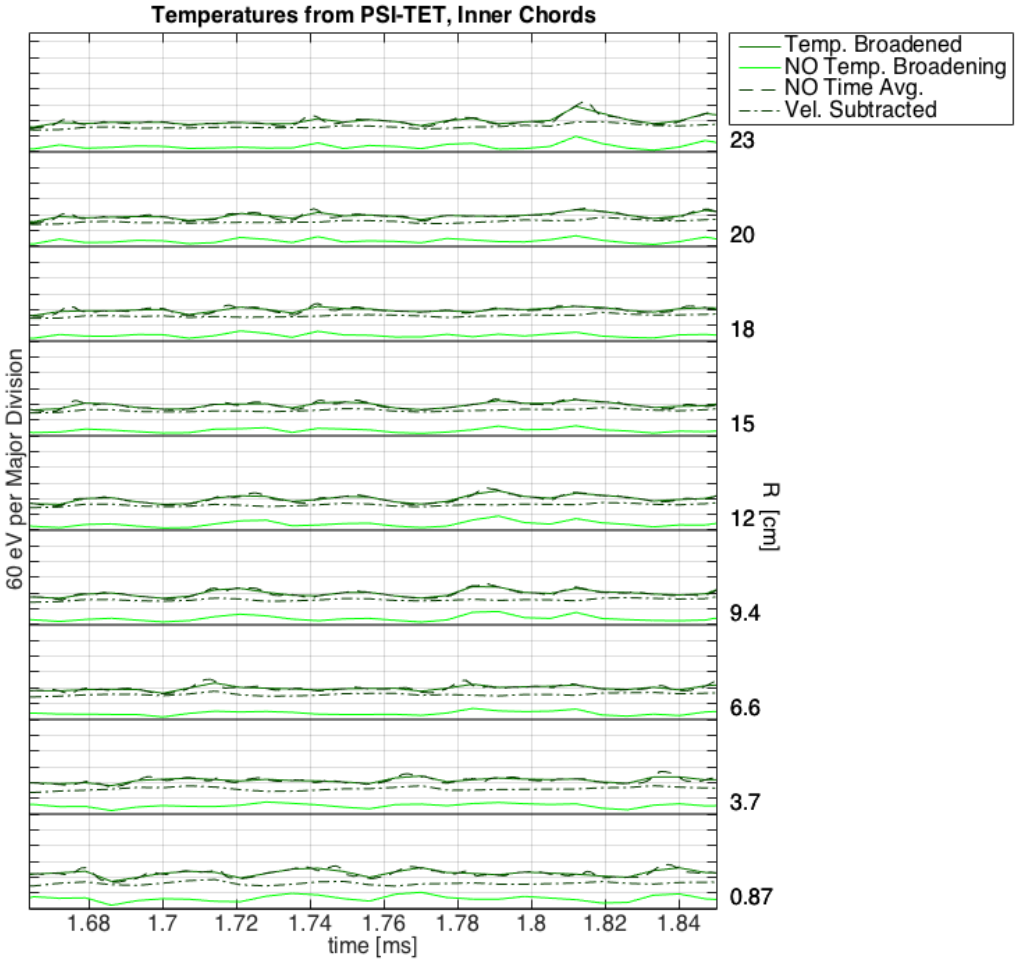

Figure 6.18: Individual components of the PSI-TET temperature as calculated through the synthetic diagnostic. The forest green line, copied from figure 6.13, includes temporal and spatial integration. The dashed, dark green line is not time-integrated; it is calculated from individual output files, i.e.: "snapshots" every ~ 1 μs. The neon green line neglects the *thermal* component of temperature, thus this temperature is entirely due to different $\vec{v} \cdot \vec{dl}$ values along the chords. The dot-dashed line is the neon line subtracted from the forest green line, yielding only the thermal component of temperature.

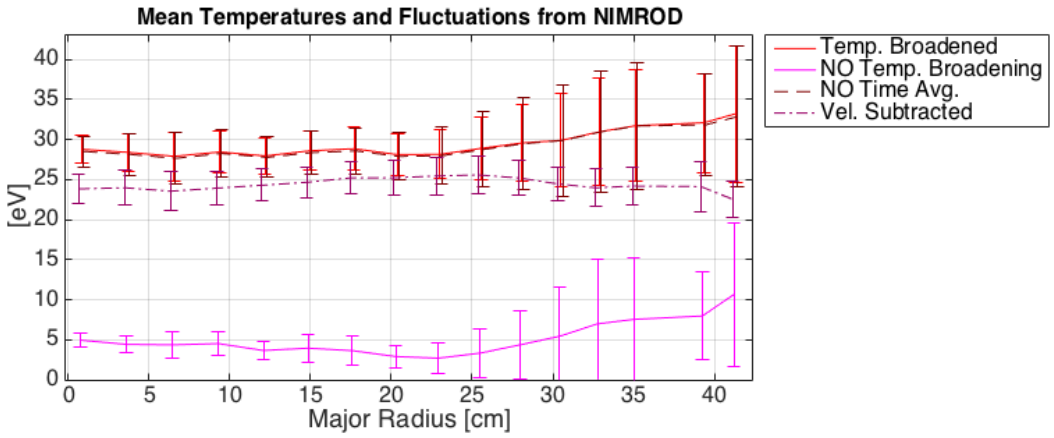

Figure 6.19: Time-averged profiles of temperature components from the NIMROD simulation as a function of impact parameter. Detailed description is in the captions of figures 6.15 and 6.16.

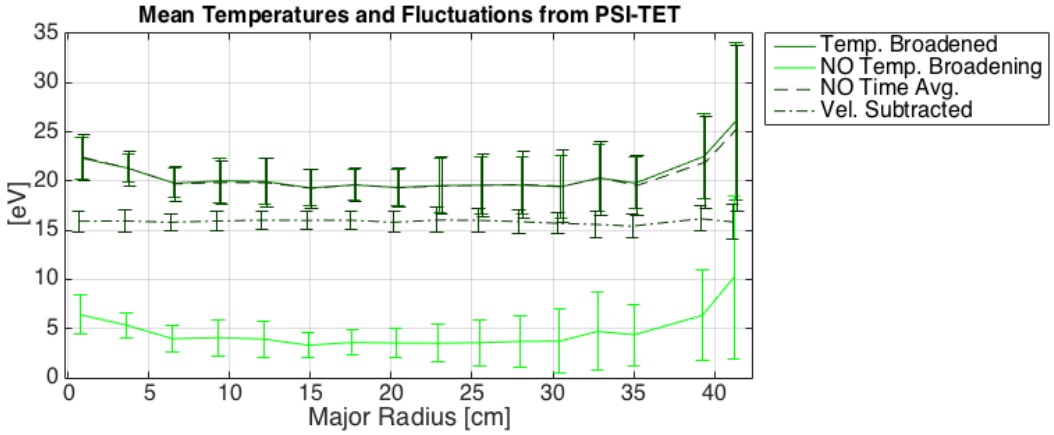

Figure 6.20: Time-averged profiles of temperature components from the PSI-TET simulation as a function of impact parameter. Detailed description is in the captions of figures 6.17 and 6.18.

along with the radial temperature profile derived from the same simulation. Within the range spanned by the chords the radial temperature variation is 14 to 28 eV.

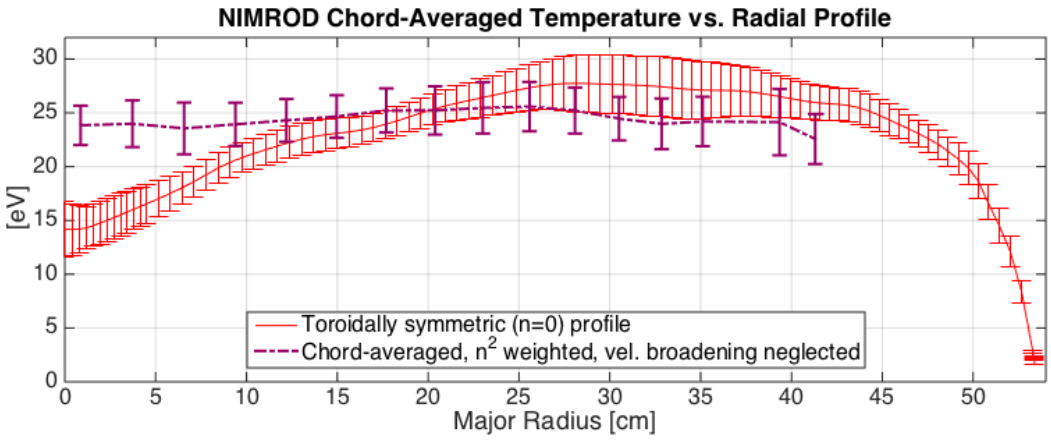

Figure 6.21: NIMROD toroidally symmetric temperature profile and chord-averaged temperature profile from figure 6.19. Data are time-averaged from 1.664 to 1.85 ms. Error bars represent fluctuations in time.

6.4 Intensity Comparison

The intensities from C III and O II emission are plotted in figures 6.22 and 6.23 along with the intensities from simulations, which are proportional to chord-integrated n^2. The IDS is not absolutely calibrated so the data sets have been scaled to fit on the same plot for comparison of trends. Near the magnetic axis, for impact parameter $R \simeq 28$ and 35 cm, the C III emission fluctuates by ∼100% but the O II emission varies much less, and approximately at the opposite phase. A point of agreement is that the simulated intensities oscillate in phase with the C III emission.

The time-averaged intensities and fluctuations are plotted as a function of impact parameter in figure 6.24. The larger amplitude and fluctuation levels of C III are evident, as are the different spatial profiles. The measured intensities increase with impact parameter even though the chords are shorter (the data have not been scaled for chord length). The

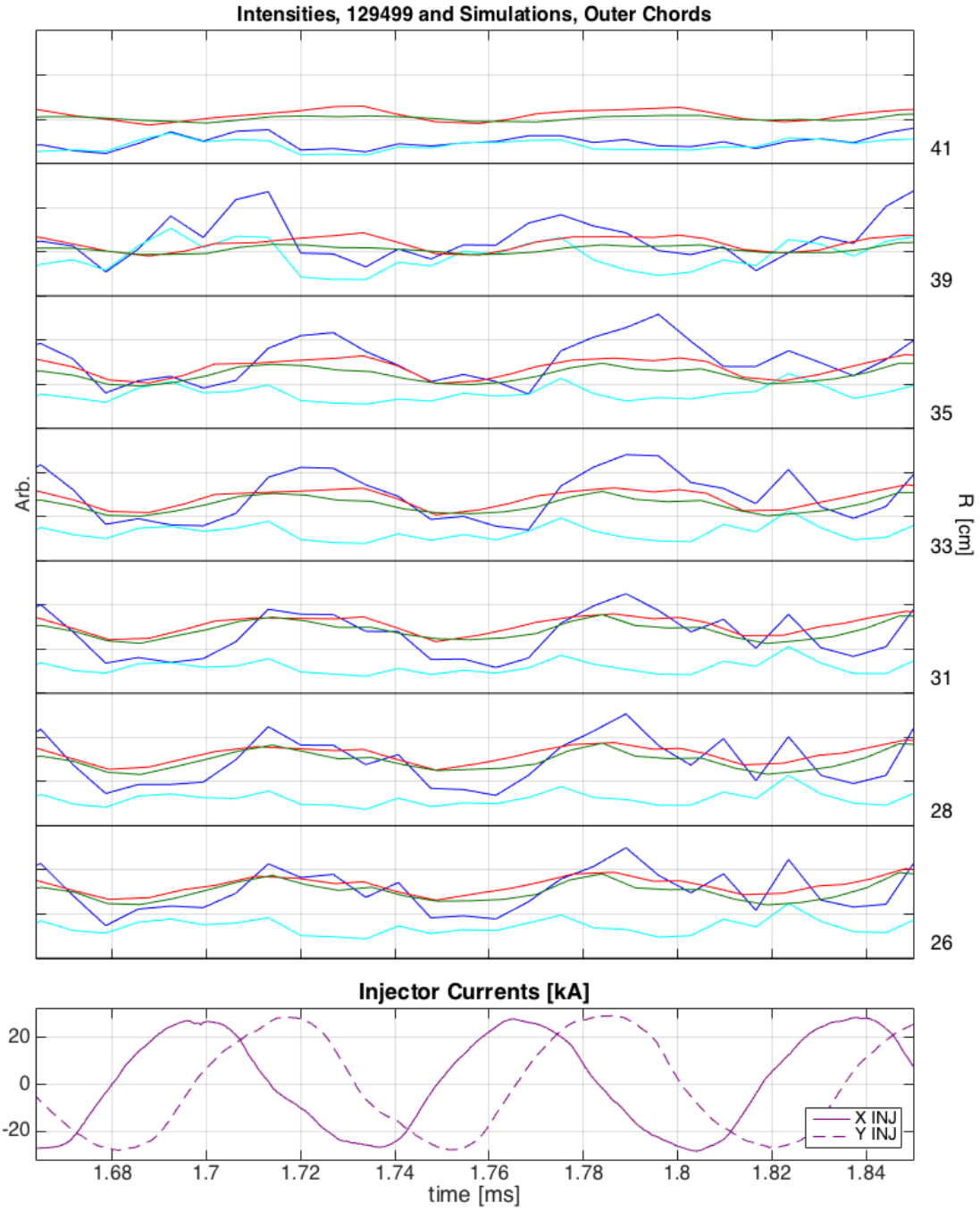

Figure 6.22: IDS O II (cyan) and C III (blue), NIMROD (red), and PSI-TET (green) intensities for outer chords. The solid, horizontal lines mark the zero level of each chord. The NIMROD and PSI-TET data are scaled identically as are the O II and C III data, but the scaling between expermental and synthetic data is arbitrary. Injector currents shown are from shot 129499.

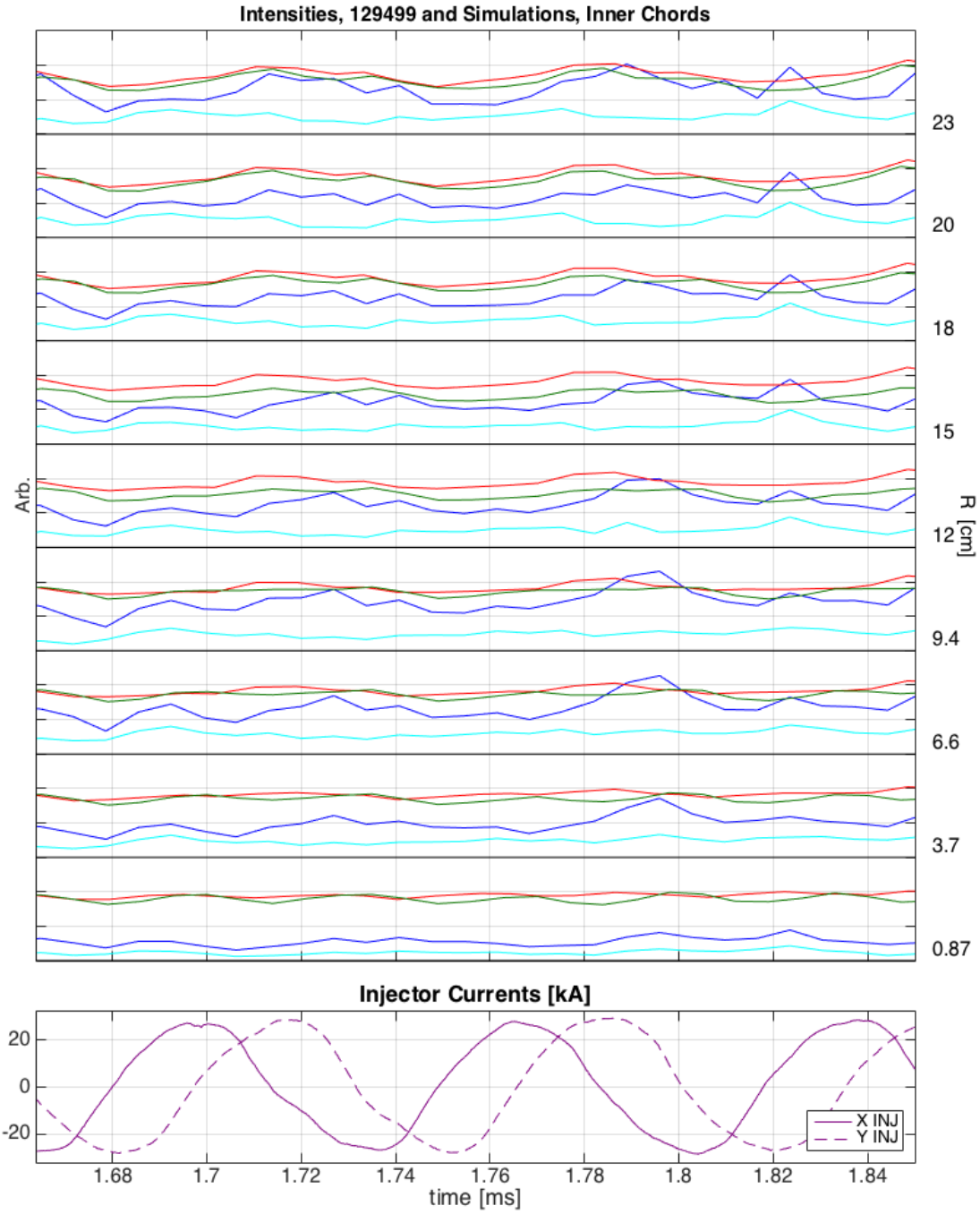

Figure 6.23: IDS O II (cyan) and C III (blue), NIMROD (red), and PSI-TET (green) intensities for inner chords. The solid, horizontal lines mark the zero level of each chord. The NIMROD and PSI-TET data are scaled identically as are the O II and C III data, but the scaling between expermental and synthetic data is arbitrary. Injector currents shown are from shot 129499.

intensities from simulations show smooth trends toward lower intensities at larger R. This is due to the lower integrated density over shorter chord lengths.

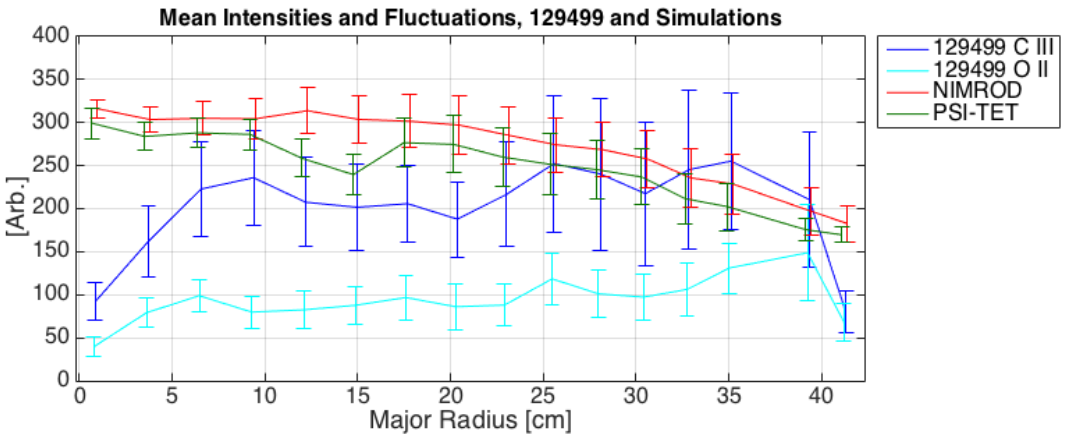

Figure 6.24: IDS O II and C III, NIMROD, and PSI-TET mean and fluctuating intensity profiles for all toroidal midplane chords as a function of impact parameter. Velocities are averaged over the time window shown in figures 6.22 and 6.23 and error bars are the standard deviations of velocities.

6.4.1 Analysis of Intensities

Midplane slices of the densities from NIMROD are plotted in figure 6.25. The two images are from opposite injector-cycle phases to illustrate the temporal nature of the oscillations. The images show that the density fluctuations at the injector frequency have predominantly $n = 1$ structure. This is driven by increased density near injector mouths with flux entering the volume and decreased density near mouths with flux leaving the volume [37]. A similar effect is also seen in the PSI-TET simulation [19].

6.5 Discussion

Both simulation codes qualitatively capture key features of the measured velocities: low velocities for small impact parameters and fluctuations at the injector frequency on chords

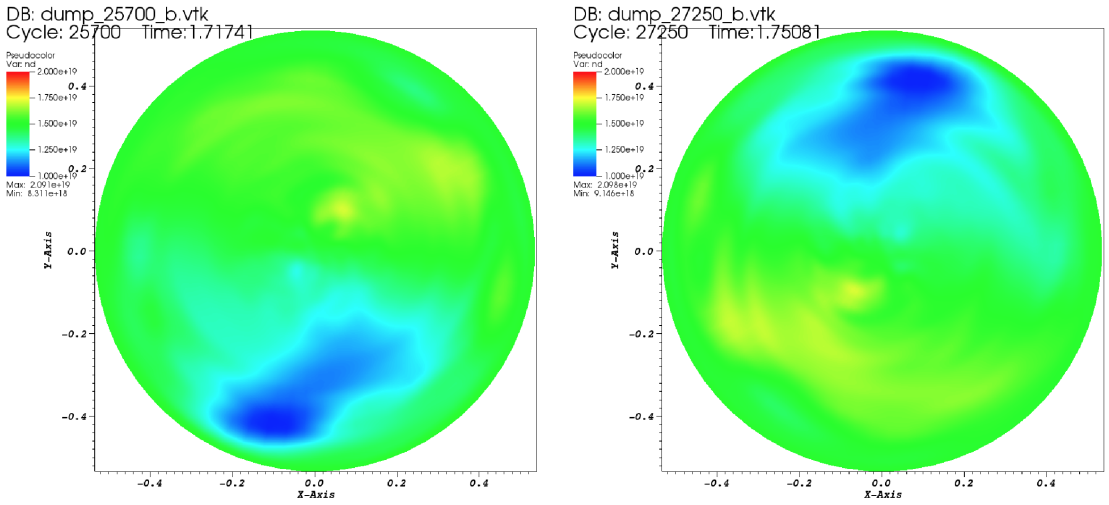

Figure 6.25: Pseudocolor plots of density in the midplane. The left plot is at an approximate maximum of chord-averaged density and the right plot is one half-cycle later. Color scale range is from 1 to 2×10^{19} m^{-3}.

near the magnetic axis. The simulations do not show the velocity phase-locking observed with the IDS. Additionally, the simulations show significantly larger velocities on the outer two chords.

One candidate for the disagreement is the lack of neutral drag in the models. Neutral particle drag in the experiment, especially in the edge, may prevent strong flows. Another possible cause is low simulated viscosity. Cihan Akcay varied viscosity for a zero-β, low temperature simulation and found magnetic energy and toroidal current to be relatively insensitive (see reference [3], section 4.1.2). Kinetic energy, however, varied greatly. At the time, the relatively low viscosity of 260 m^2/s was selected as a standard setting because the magnetic energy and current matched the experiment more closely.

The observation of negative kinetic helicity in NIMROD is of interest in a purely computational and theoretical context, but further study is outside the scope of this work.

The measured C III temperatures are between those from PSI-TET and NIMROD, indicating qualitative agreement between all three data sets but also the need for verification between the two codes. Both codes show "temperature" spikes due to differential velocities

that are not observed in the experimental measurements. This suggests that the simulations undergo reconnection and relaxation events which do not reflect the reality of HIT-SI dynamics. This result also supports the need for a new viscosity scan. Comparing the radial temperature profile to the chord-averaged profile shows how chord-averaging can "hide" a temperature gradient. Thus, a minor-radial temperature gradient may exist in HIT-SI but the IDS is insensitive to it.

The measured intensities agree poorly with the simulations in both fluctuation levels and profiles. Presently, the simulations cannot be run with realistic density variations for numerical stability reasons, so unrealistically high diffusivity is used. Thus there is a practical limit to the agreement achievable with density comparisons.

Chapter 7

MEASUREMENTS AT SMALL R INCLUDING HIGHER INJECTOR FREQUENCIES

The injector driving frequency of HIT-SI was incremenally raised to 36.8, 53.5, and 68.5 kHz. For operations at 53.5 and 68.5 kHz an outward shift in the magnetic profile was observed [54]. Grad-Shafranov fitting produces β values between 21% and 29% for 68.5 kHz shot 129175, whereas 14.5 kHz shot 122385 has $\beta < 7\%$. Optimistically this indicates that HIT-SI confines plasma pressure at higher injector frequency, but alternative explanations are considered. Toroidal plasmas can shift outward due to the high inductance of a peaked λ profile. However, a peaked λ profile has never been observed in sustained spheromaks and a peaked profile does not match the data. Another mechanism that could cause a shifted profile is a strong toroidal mass (ion) flow. IDS measurements of toroidal flow can be used to calculate the outward shift due to rotation.

IDS temperature measurements can be combined with other data to directly calculate β. By combining the T_i from IDS with n_e from the FIR interferometer and $< B >_{wall}$ from surface magnetic probes, the plasma $< \beta >_{wall}$ can be estimated according to the equation

$$\beta_{wall} = \frac{2\mu_0 p}{B^2_{wall}}. \tag{7.1}$$

This β can be compared with the β from Grad-Shafranov fitting to the internal magnetic profile.

Another result from reference [54] shows that density oscillations decrease with higher injector frequency, but that result is based on one interferometer chord. IDS intensity data can be used to estimate intensity fluctuations at smaller major radii by assuming $I \propto n^2$.

IDS data were collected through the radial midplane reentrant port (figure 4.1, left) during high and low frequency HIT-SI operations. Due to power supply and circuit limitations the power coupled to the plasma decreased with increasing frequency. Consequently the

toroidal currents also decreased and the plasma density and fueling decreased to maintain a similar j/n for optimal performance. Since plasma emission scales roughly as n^2, the image intensifier was required for injector frequencies higher than 14.5 kHz. Unfortunately the intensifier, which uses a microchannel plate, is prone to saturating. This was not fully characterized until after operations at 36.8 kHz were complete, so all data at that frequency is suspect.

In the following sections data are presented from 53.5, 68.5, and 14.5 kHz operations. From these data, toroidal roation, plasma βs, and density fluctuation scalings are calculated.

7.1 53.5 kHz Operations

7.1.1 Operating Parameters

Selected data from a typical shot, 128580, are shown in figure 7.1. Additional data are shown in appendix B.3. Note the toroidal current reaches a lower value than at 14.5 kHz, 26 kA vs. 90 kA, but is flatter and maintains current gain near 3 for the majority of the shot. Injected power is lower, 4 MW vs. ∼13 MW, and the continuum radiation and carbon ratio are not as dynamic as at 14.5 kHz. The steady currents, power, and radiation suggest that shot 128580 is closer to steady-state than 14.5 kHz shots.

7.1.2 Plasma Spectral Lines

Since shots at 53.5 kHz have lower density and injected power than 14.5 kHz the spectral lines were not able to be resolved by the high speed camera on the timescale of the injector period. The Quantum Leap image intensifier by Stanford Computer Optics was added to the system. In order to ensure the data were not saturated it was operated at ∼6% duty cycle. The intensifier and camera would expose for 4.67 μs, 1/4 of an injector period, at a rate of 13.375 kHz, 1/4 of the injector frequency. Thus every four injector cycles the system captures the spectra, always at the same injector phase. The camera begins recording before the discharge is triggered, so the exposure time relative to the injector phase is random for any given shot. Therefore, by combing spectral data at different injector phases from similar shots an overall picture of the dynamics emerges. The exposure times in relation to injector

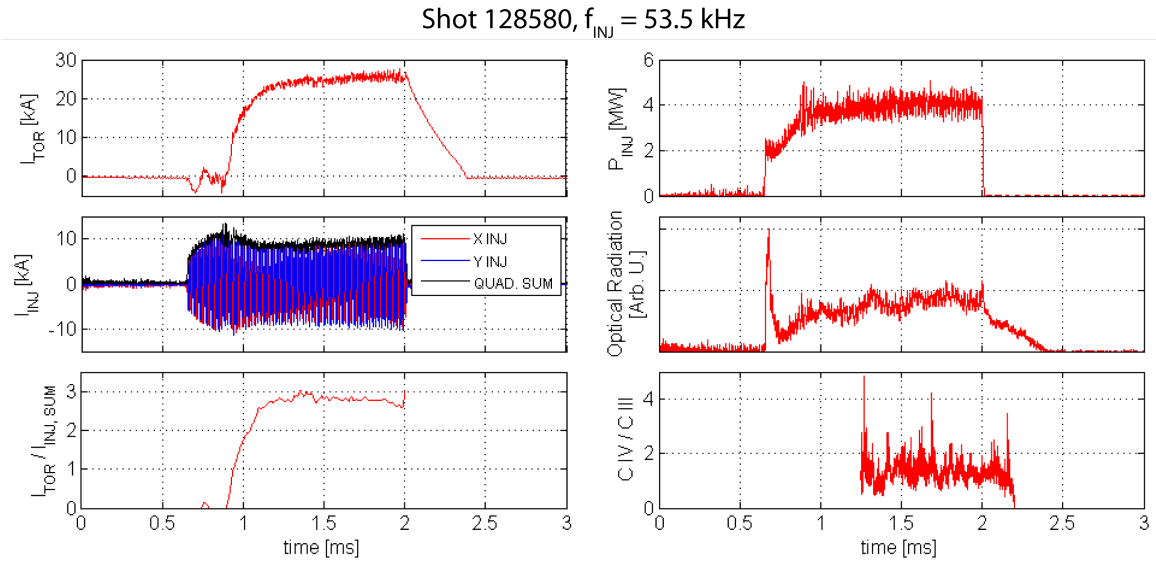

Figure 7.1: Measured quantities for HIT-SI shot 128580 in the same format as figure 5.1. Plotted are toroidal current, X and Y injector currents and their quadrature sum (middle left), current amplification (lower left), power injected (upper right), broadband radiation (middle right), and the ratio of C IV to C III (lower right).

phase are illustrated in figure 7.2 for a set of eight similar shots.

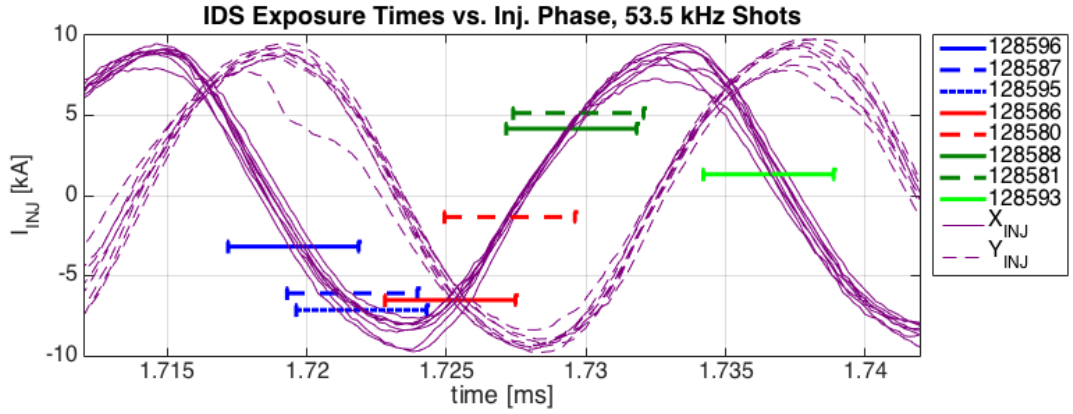

Figure 7.2: IDS exposure times of various shots in relation to the injector currents. Shots have been color-coded by 1/4 injector period. All X and Y injector currents are plotted to illustrate random shot-to-shot variation of ~ 1 μs.

The O II emission at this frequency is brighter than C III, opposite of the observed spectra at 14.5 kHz (see figure 5.2). Figure 7.3 shows a wavelength profile obtained by summing the pixels over an entire sub-bundle of the fiber optic. The brightest line, O II, is the only usable line from these shots; the data from individual fibers for other emission lines is too weak to use.

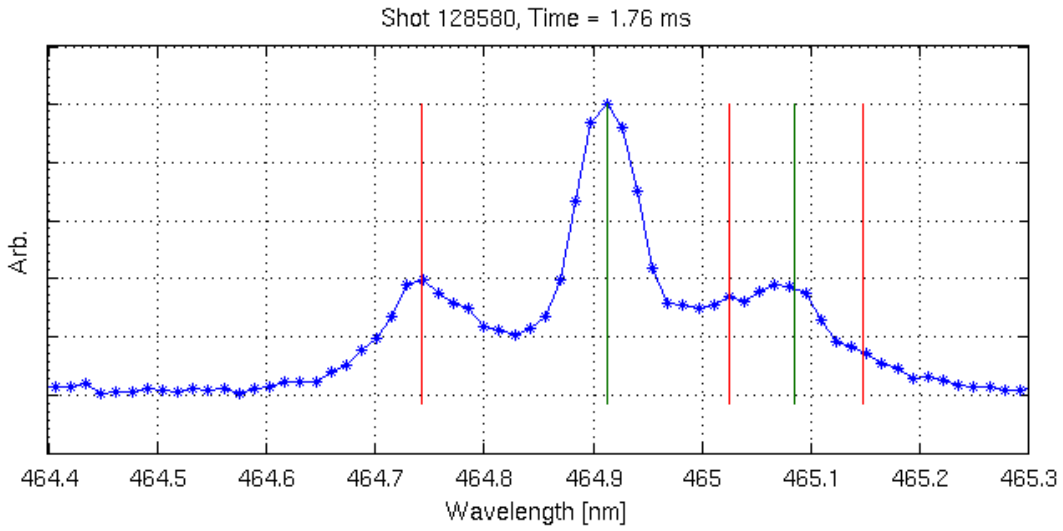

Figure 7.3: CCD intensity as a function of wavelength for a shot at 53.5 kHz injector frequency. Pixels are summed in the spatial direction to obtain the profile. The red lines indicate C III emission wavelengths 464.742, 465.025, and 465.147 nm. The green lines indicate O II emission at 464.913 and 465.084 nm.

7.1.3 Temperature and Intensity Profiles

The mean, chord-averaged temperatures and their certainties during discharges are shown in figure 7.4. For clarity the data are separated by injector 1/4 period and color coded to match figure 7.2. The error bars plotted are the certainties of the mean temperatures from 1.5 to 2.0 ms. Since most other signals (currents, injected power, radiation, carbon ratio) indicate the plasma is in a steady-state, the error bars are indicative of measurement uncertainty. At small impact parameters ($-10 < R < 10$ cm) the average temperature and variation over all shots is 29 ± 10 eV. Farther out (impact parameters > 20 cm) the

temperature is reduced to 17 ± 6 eV. Over the entire data set the average temperature is 25 ± 8 eV.

The intensities for the same data are shown in figure 7.5, separated and color coded identically to figure 7.4. The standard deviation of intensity for impact parameters > 20 cm is 20%, and there is a slight trend toward lower intensity near the geometric axis.

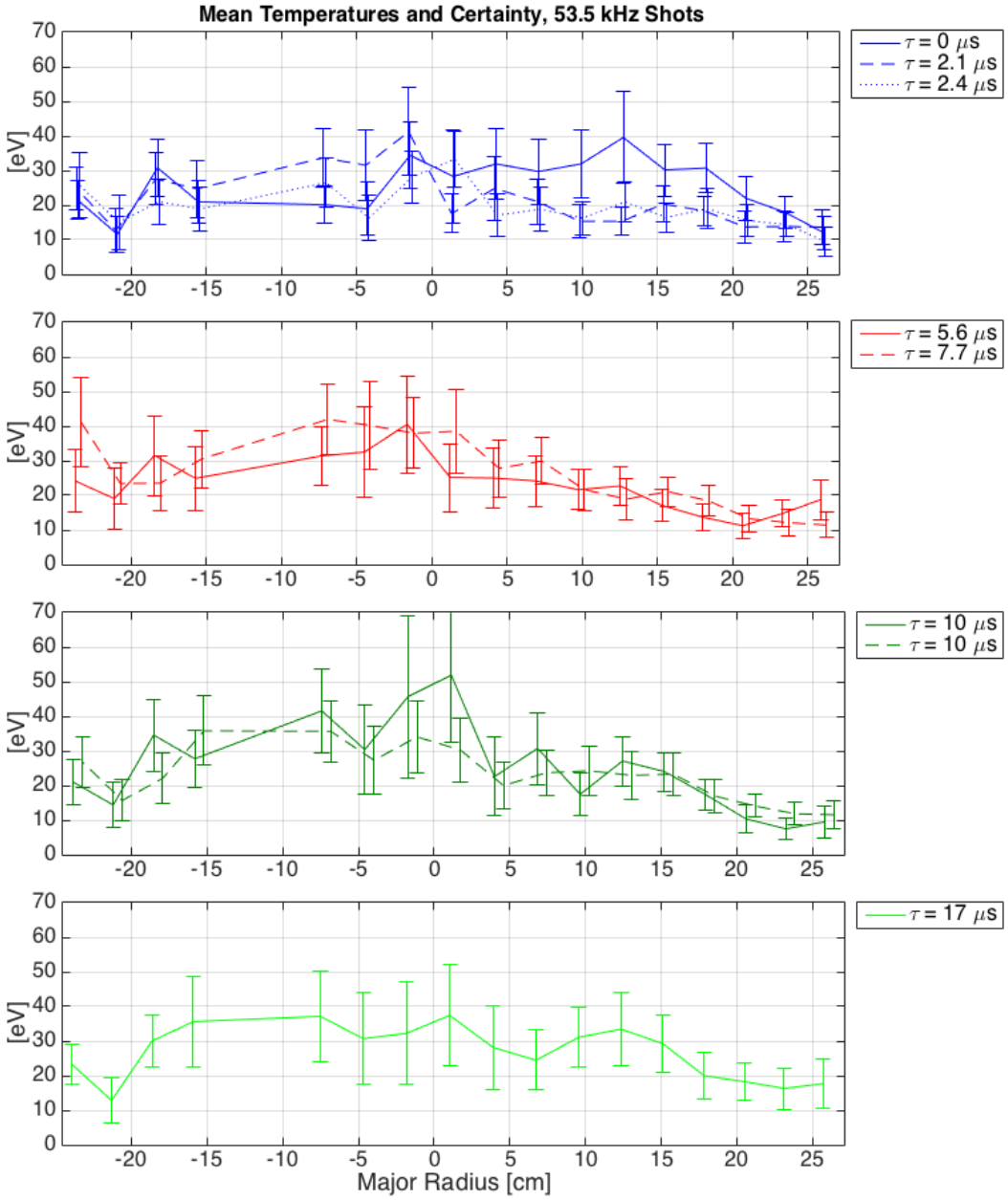

Figure 7.4: Mean, chord-averaged temperatures and certainties for eight similar shots, separated by 1/4 injector cycle and color coded to match figure 7.2. The means and certainties are taken during sustainment from $t = 1.5$ to 2.0 ms. Note the error bars are certainty $(\sigma/\sqrt{N})$ where $N = 8.5 \pm 0.6$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

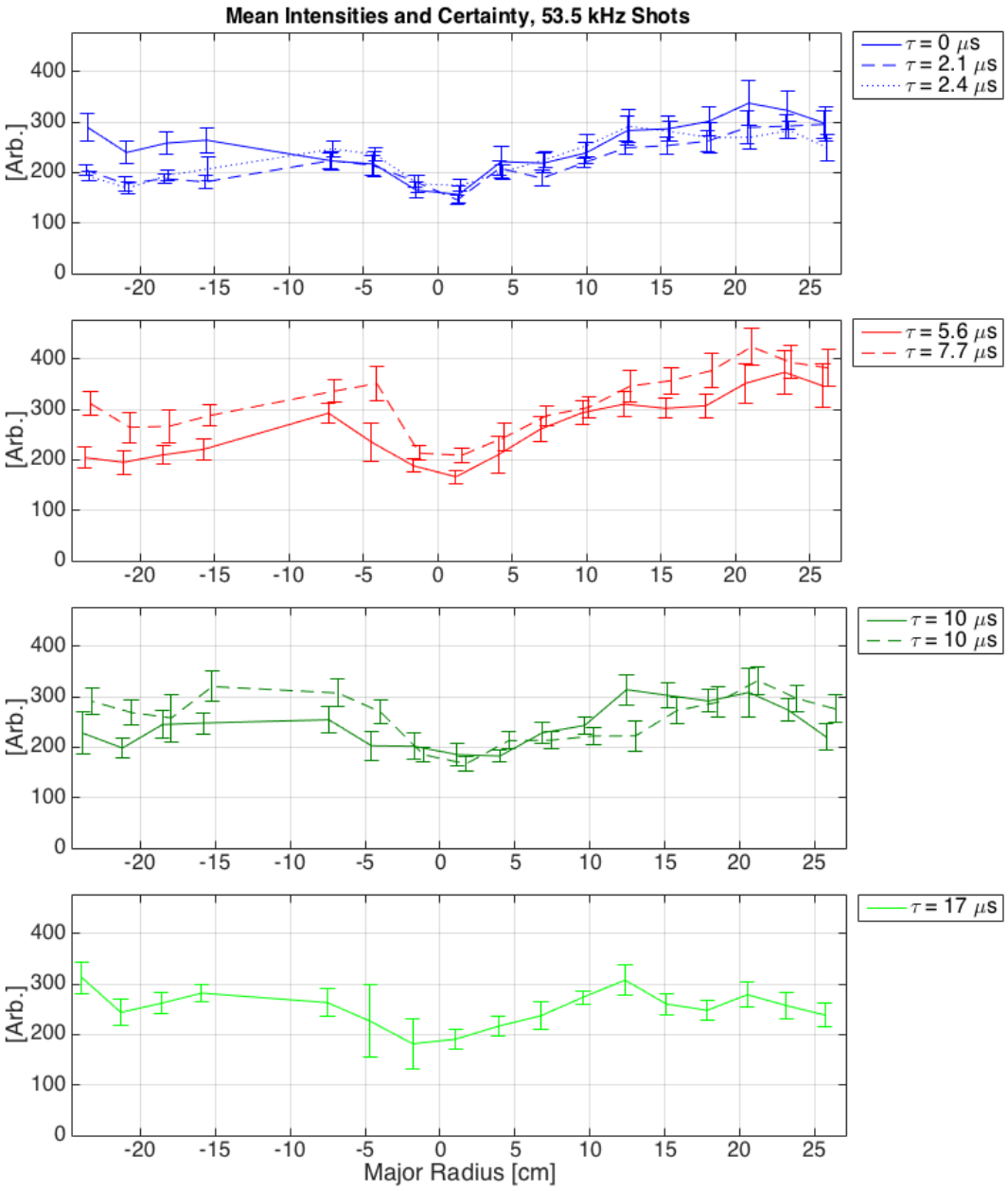

Figure 7.5: Mean, chord-averaged intensities and certainties for same shots and time period as figure 7.4. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 8.6 \pm 0.5$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

7.2 68.5 kHz Operations

7.2.1 Operating Parameters

The record current amplification of 3.9 for HIT-SI and spheromaks was achieved at 68.5 kHz [54]. IDS was not operating optimally for the best shot, 129175, but data were collected for four similar shots. Parameters for one of the shots with current amplification of nearly 3 is shown in figure 7.6, with additional data found in appendix B.4. Note the absolute value of toroidal and injector currents was further reduced from 53.5 kHz operations. Power injected was ramped with a greater slope than 53.5 kHz but reached a slightly lower value. The carbon ratio was more dynamic and achieved higher values than at 53.5 kHz.

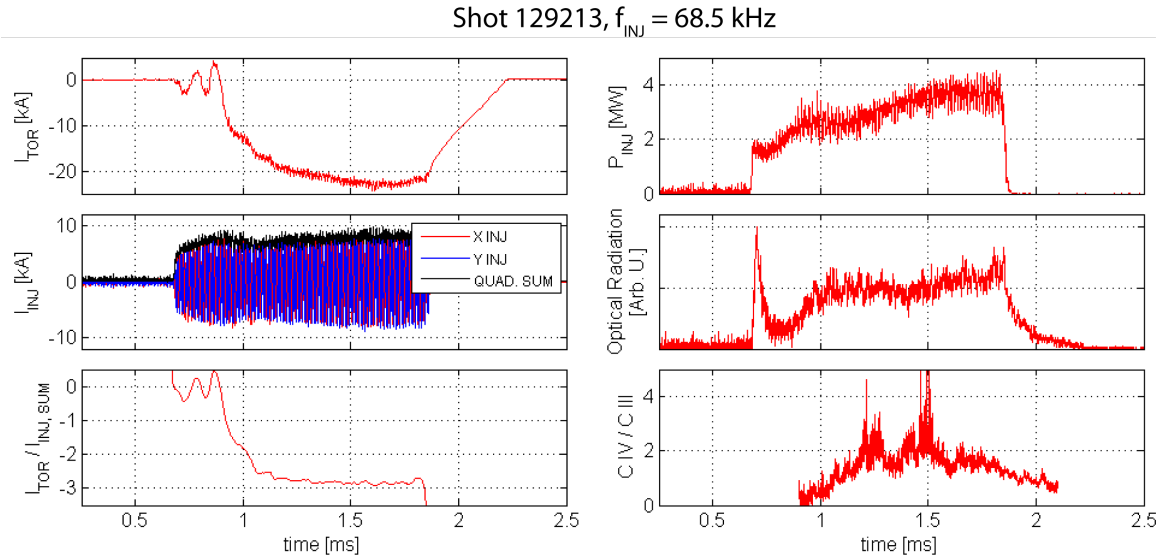

Figure 7.6: Measured quantities for HIT-SI shot 129213 in the same format as figure 5.1. Plotted are toroidal current, X and Y injector currents and their quadrature sum (middle left), current amplification (lower left), power injected (upper right), broadband radiation (middle right), and the ratio of C IV to C III (lower right).

7.2.2 Plasma Spectral Lines

The detector configuration for 68.5 kHz data is identical to that of 53.5 kHz. The plasma line shapes are also similar, with the brightest O II line being the only useable line. The

camera was operated with a frame rate of 22.384 kHz, 1/3 of the injector frequency. The exposure was 7.3 μs which is half of an injector period. Thus the camera exposes for 1/2 of an injector period every three cycles. The four shots shown in the following sections are aligned with different, randomly selected phases of the injector cycle, as illustrated in figure 7.7.

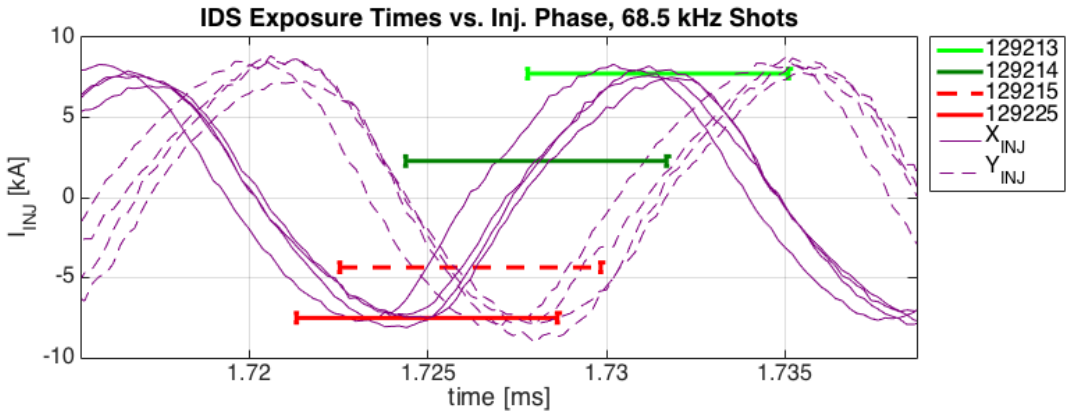

Figure 7.7: IDS Exposure times of various shots in relation to the injector currents. Shots have been color-coded by 1/4 injector period in the same way as in figure 7.2. All X and Y injector currents are plotted to illustrate random shot-to-shot variation which is < 1 μs except for 129215.

The IDS was collecting data for three shots with current gain above 3. The detector was exposing for an entire injector period, 14.6 μs, and the image intensifier became saturated toward the end of the discharge. Some data are usable from earlier in the discharge, however, and the current gain is $\sim 3.5 \pm 0.3$.

7.2.3 Temperature and Intensity Profiles

The chord-averaged temperature profiles for the 68.5 kHz shots with current gain of 3 are plotted in figure 7.8. For chords in the central region, $-10 < R < 10$ cm, the average temperature is 28 ± 6 eV and in the outer region, $|R| > 20$ cm, the temperature is 20 ± 6 eV. The average temperature and variation for all chords is 26 ± 6 eV.

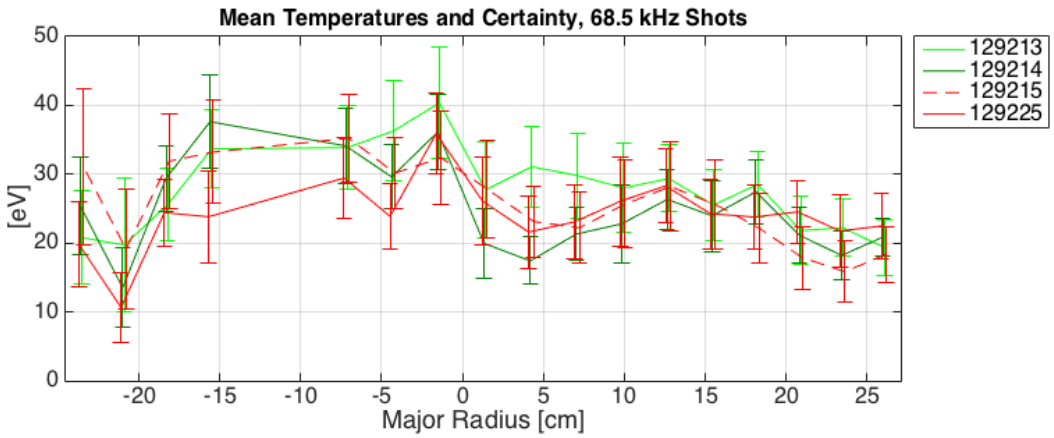

Figure 7.8: Mean, chord-averaged temperatures and certainties for four shots listed. The data selected are captured during sustainment from $t = 1.6$ to 1.85 ms. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 7.3 \pm 0.4$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

The chord-averaged temperature profiles for the higher-gain 68.5 kHz shots are shown in figure 7.9. For chords in the central region, $-10 < R < 10$ cm, the average temperature is 42 ± 7 eV and in the outer region, $|R| > 20$ cm, the temperature is 24 ± 4 eV. The average temperature and variation for all chords is 33 ± 6 eV.

The chord-averaged intensity profiles for 68.5 kHz shots with current gain of 3 are plotted together in figure 7.10. The average intensity variation for impact parameters > 20 cm is 22%. The intensities from high-gain shots are not presented since the exposure covers an entire injector cycle, effectively cycle-averaging over the fluctuations which are predominantly at the injector frequency.

7.2.4 Velocity Profiles

The mean, chord-averaged velocities and fluctuations for four similar shots with current gain of 3 are plotted in figure 7.11. The mean velocity for the three chords with impact parameters between $R \simeq -24$ and -16 cm is -5.9 ± 2.2 km/s and for the opposite region ($R \simeq +16$ to 24 cm) the value is -0.1 ± 1.7 km/s. Thus, the net toroidal velocity is

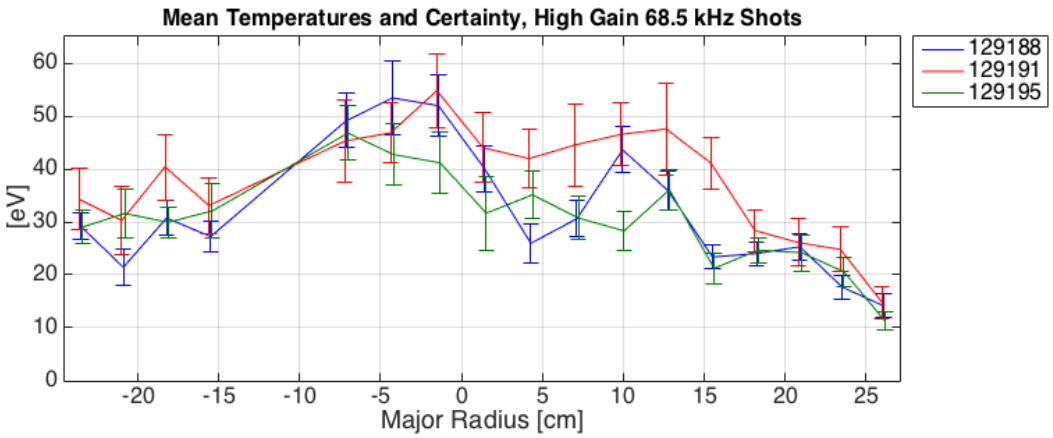

Figure 7.9: Mean, chord-averaged temperatures and certainties for high-gain 68.5 kHz shots. The data selected are captured from $t = 1.3$ to 1.5 ms. The color coding is random and only corresponds to figure 7.12. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 6$ and σ includes fitting uncertainty and the standard deviation of the data.

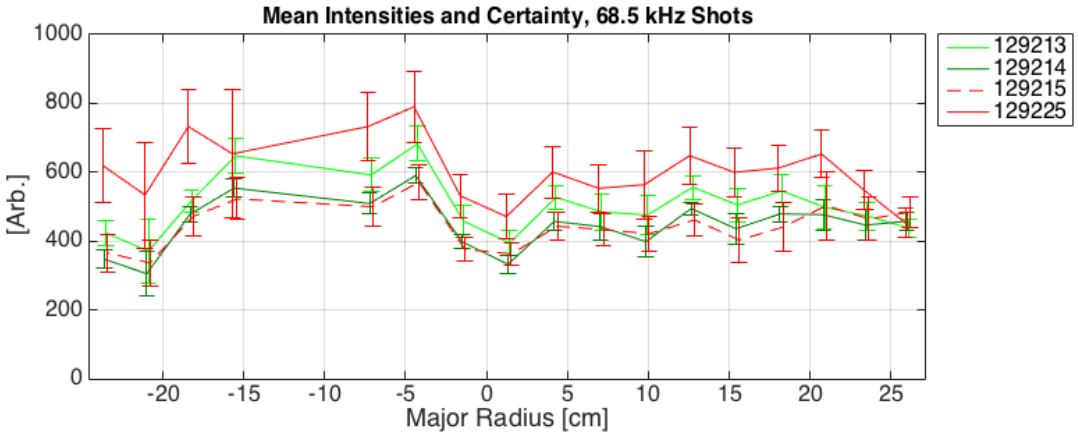

Figure 7.10: Mean, chord-averaged intensities and certainties for four shots listed. The data are selected for the time during sustainment from $t = 1.6$ to 1.85 ms. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 7.3 \pm 0.4$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

2.9 ± 2.8 km/s.

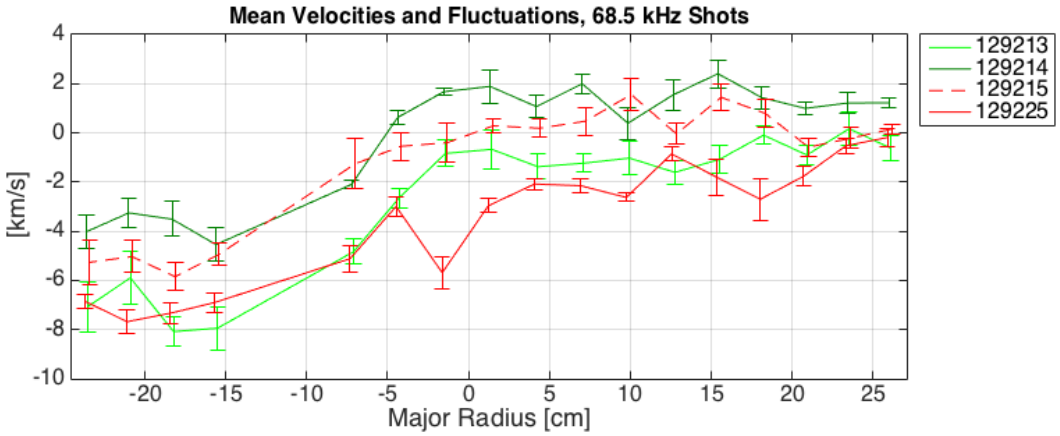

Figure 7.11: Mean, chord-averaged velocities and fluctuations for the same shots and time period as figure 7.10. The velocity zero-point is not absolutely calibrated.

The velocities from the high-gain shots are plotted in figure 7.12. The difference between the left most and right most chords is even smaller than for the other data set. There is a slight velocity trend which is reversed between the two data sets, implying toroidal flow associated with toroidal current (the high-gain shots have positive current and the shots with gain of 3 have negative current). The magnitude is still only ~ 3 km/s.

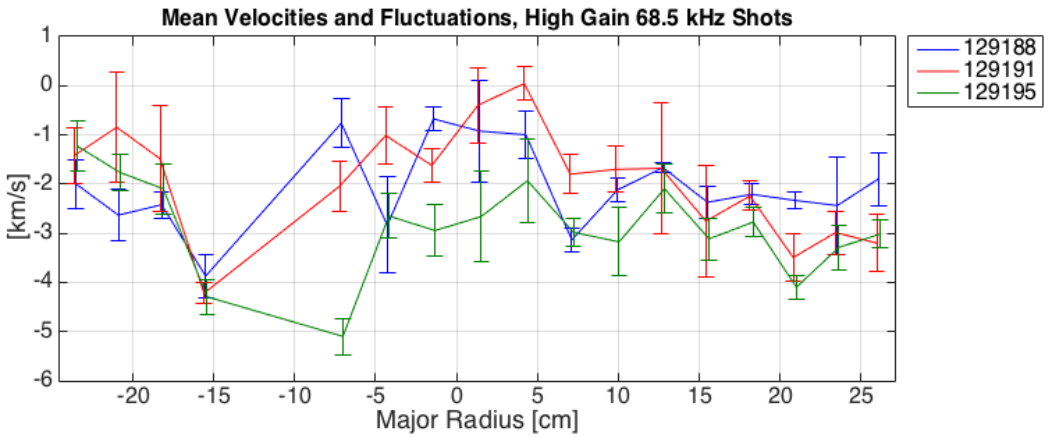

Figure 7.12: Mean velocities and fluctuations for high-gain 68.5 kHz shots, time averaged from $t = 1.3 - 1.5$ ms. The velocity zero-point is not absolutely calibrated. The color coding is random and only correspnds to figure 7.9.

7.3 14.5 kHz Operations

The IDS data presented in chapters 5 and 6 were taken from the angled midplane port (figure 4.1, right) but the high-frequency data in this chapter was only collected through the radially oriented port (figure 4.1, left). Some IDS data from high-current shots were collected through this port for the purpose of direct comparison between frequencies. Similar parameters were discussed in section 5.1 and data from the shot presented here, 129810, can be found in appendix B.5.

7.3.1 Plasma Spectral Lines

Since the image intensifier was removed the detector could be operated at nearly 100% duty cycle. Ten, 6.9 μs exposures were captured per injector cycle as illustrated in figure 7.13. Since the spectral lines for high-current data are much brighter than those from high-frequency, O II and C III are both usable (see figure 5.2).

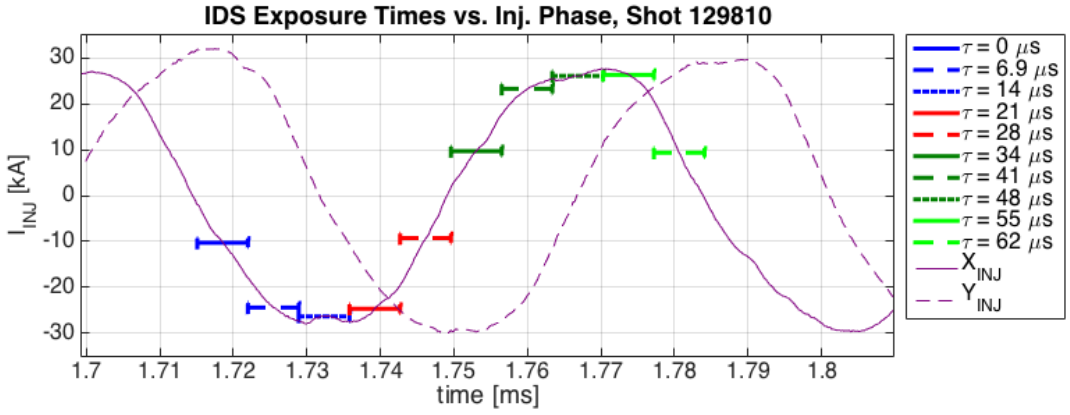

Figure 7.13: IDS Exposure times of shot 129810 in relation to the injector currents. Exposures have been color-coded by 1/4 injector period in the same way as in figures 7.2 and 7.7. "τ" is the relative time in the injector cycle.

7.3.2 Temperature and Intensity Profiles

The chord-averaged temperature profiles as a function of injector phase from the O II line in shot 129810 are plotted in figure 7.14. For chords in the central region, $-10 < R < 10$ cm, the average temperature is 8 ± 6 eV and in the outer region, $|R| > 20$ cm, the temperature is 11 ± 10 eV. The average temperature and variation for all chords is 9 ± 8 eV.

The chord-averaged temperature profiles from the brighter C III line in shot 129810 are plotted in figure 7.15. For chords in the central region, $-10 < R < 10$ cm, the average temperature is 24 ± 5 eV and in the outer region, $|R| > 20$ cm, the temperature is 25 ± 6 eV. The average temperature and variation for all chords is 23 ± 5 eV.

The chord-averaged intensities and their fluctuations for the O II line are plotted in figure 7.16. The average intensity variation for impact parameters > 20 cm is 29%. Similarly, the intensities and fluctuations for the brighter C III line are plotted in figure 7.17. The average intensity variation for impact parameters > 20 cm is 33%.

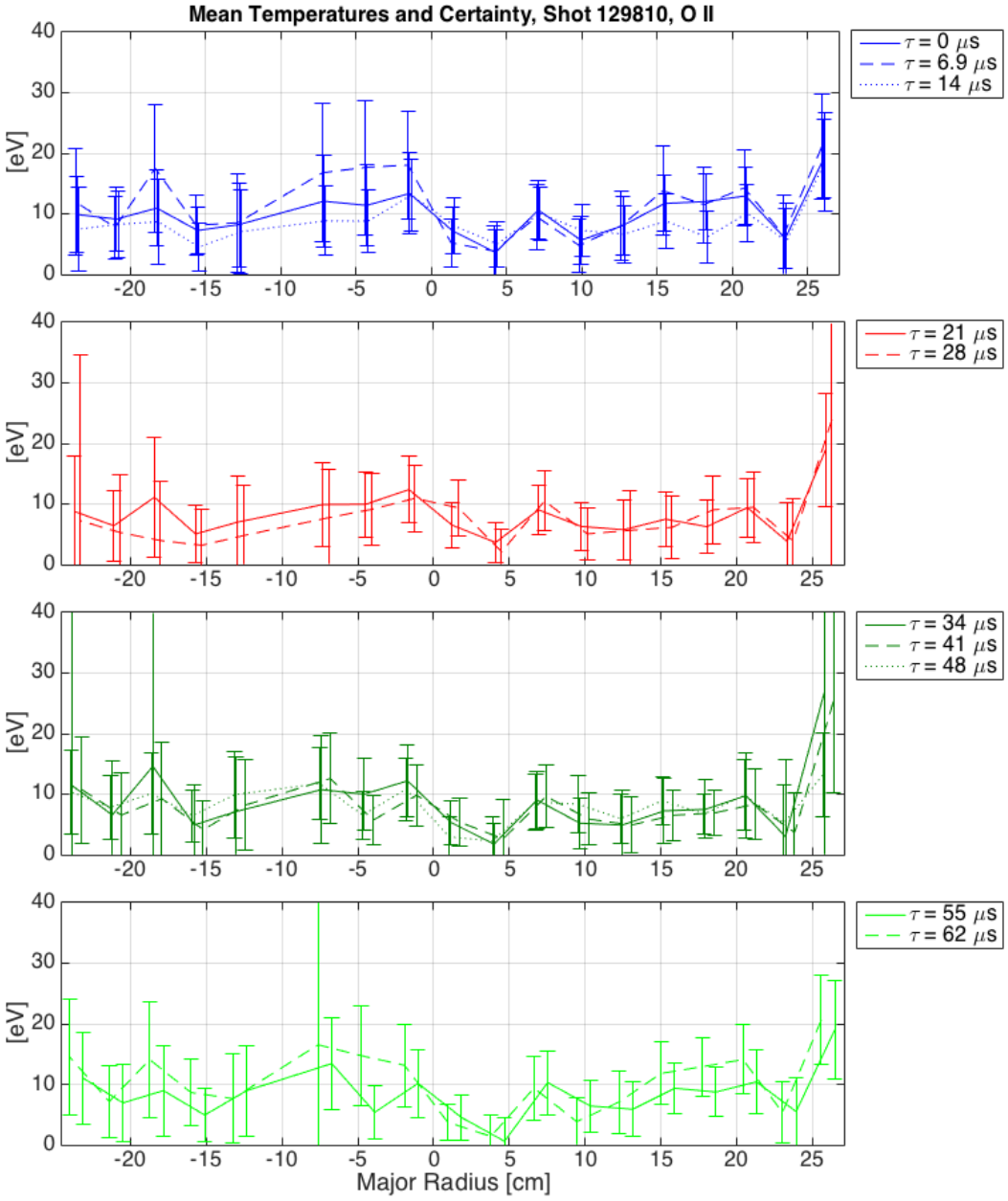

Figure 7.14: Mean, chord-averaged temperatures and their certainties for O II in shot 129810, separated by 1/4 injector cycle and color coded to match figure 7.13. Each of the ten lines is the temperature during a different 10^{th} of the injector cycle for several cycles during sustainment from $t = 1.6$ to 1.9 ms. The relative time within the 69 μs injector period is denoted by "τ". Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 4.5 \pm 0.6$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

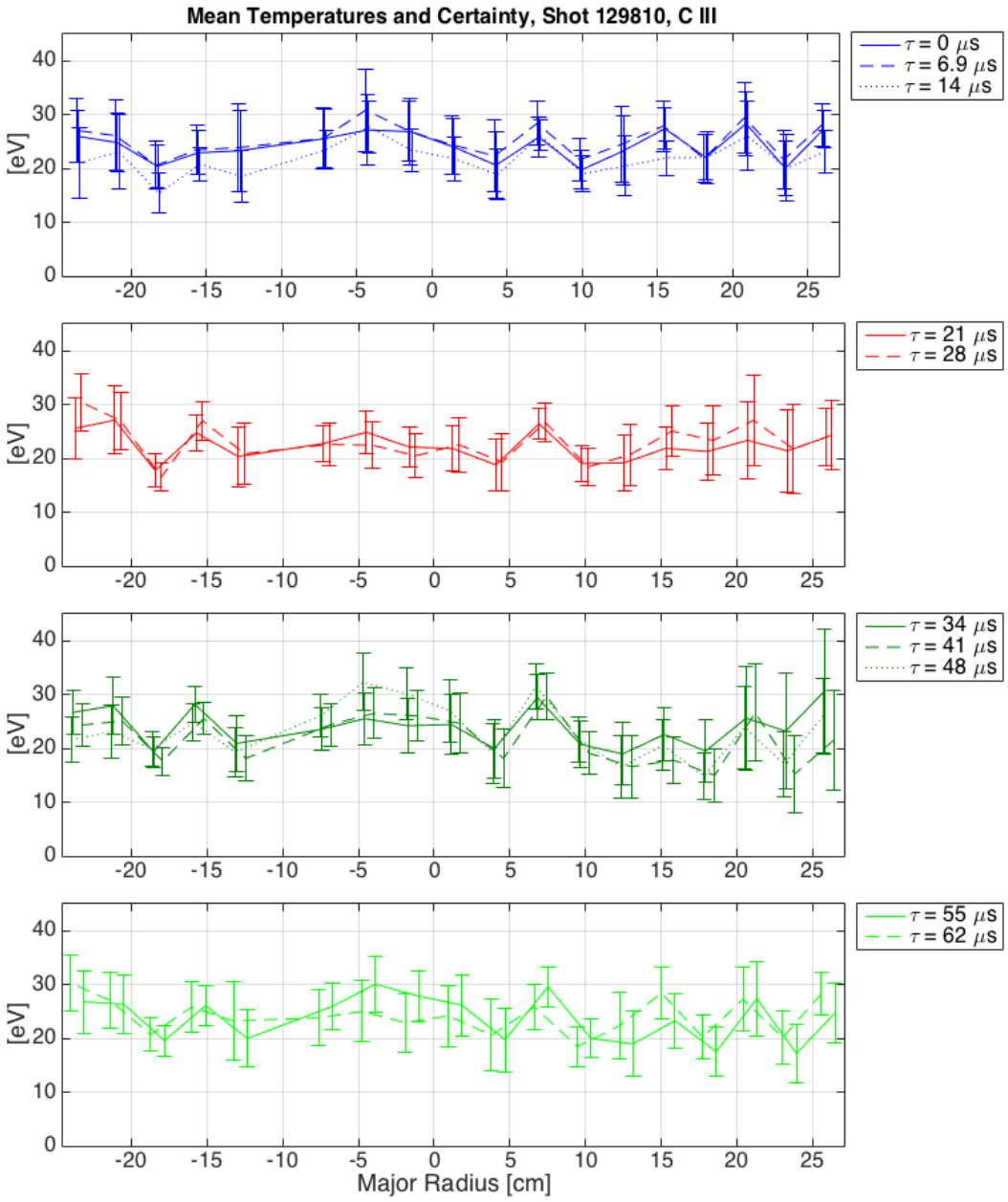

Figure 7.15: Mean, chord-averaged temperatures and their certainties for C III in shot 129810, separated by 1/4 injector cycle and color coded to match figure 7.13. Each of the ten lines is the temperature during a different 10^{th} of the injector cycle for several cycles during sustainment from $t = 1.6$ to 1.9 ms. The relative time within the 69 μs injector period is denoted by "τ". Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 4.6 \pm 0.5$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

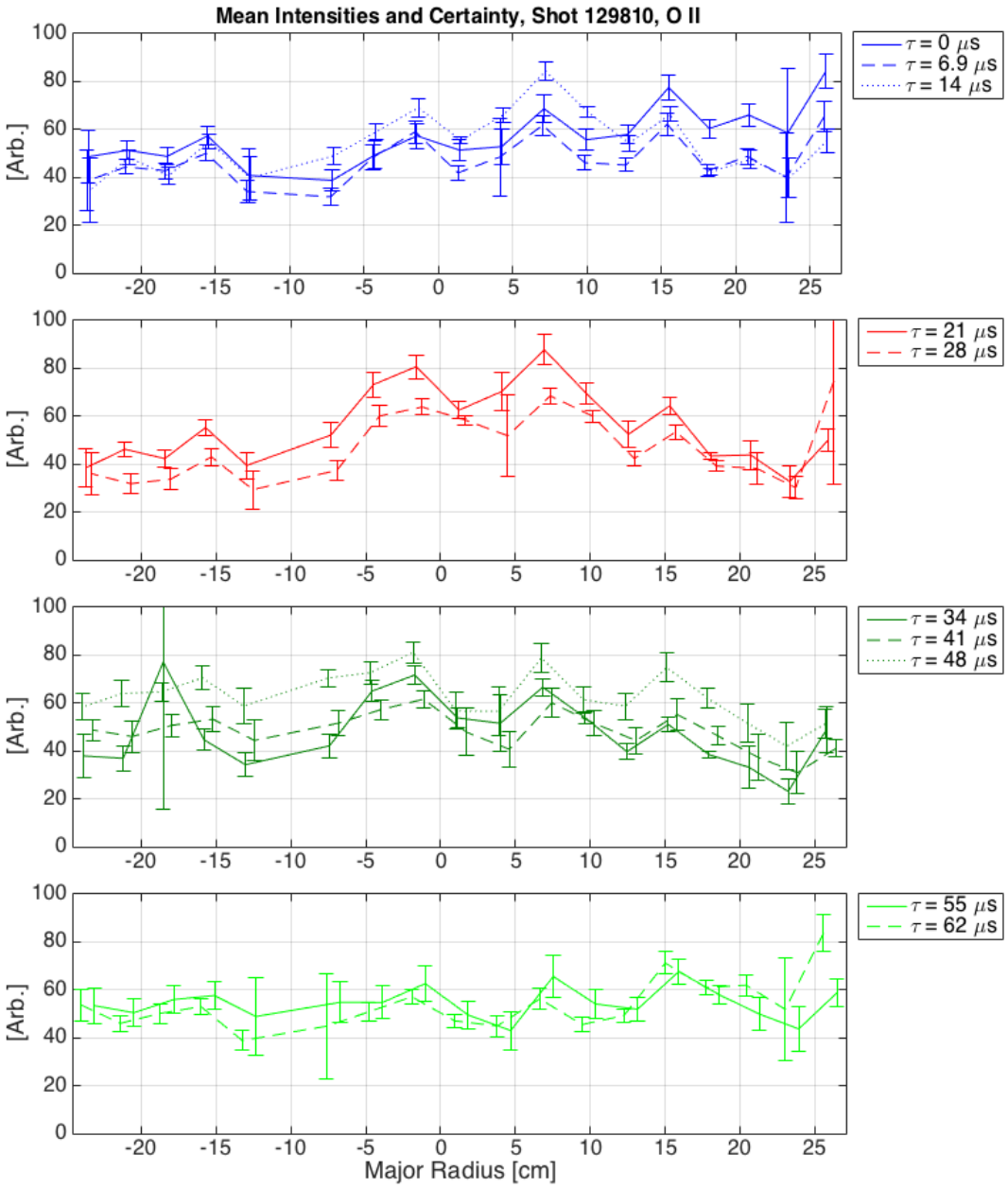

Figure 7.16: Mean, chord-averaged intensities and certainties for O II in shot 129810, separated by 1/4 injector cycle and color coded to match figure 7.13. Each of the ten lines is the mean and certainty of emission during a different 10^{th} of the injector cycle. The relative time within the 69 μs injector period is denoted by "τ". The data are selected during sustainment from $t = 1.6$ to 1.9 ms. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 4.6 \pm 0.5$ (depending on the number of points in the time window) and σ includes fitting uncertainty and the standard deviation of the data.

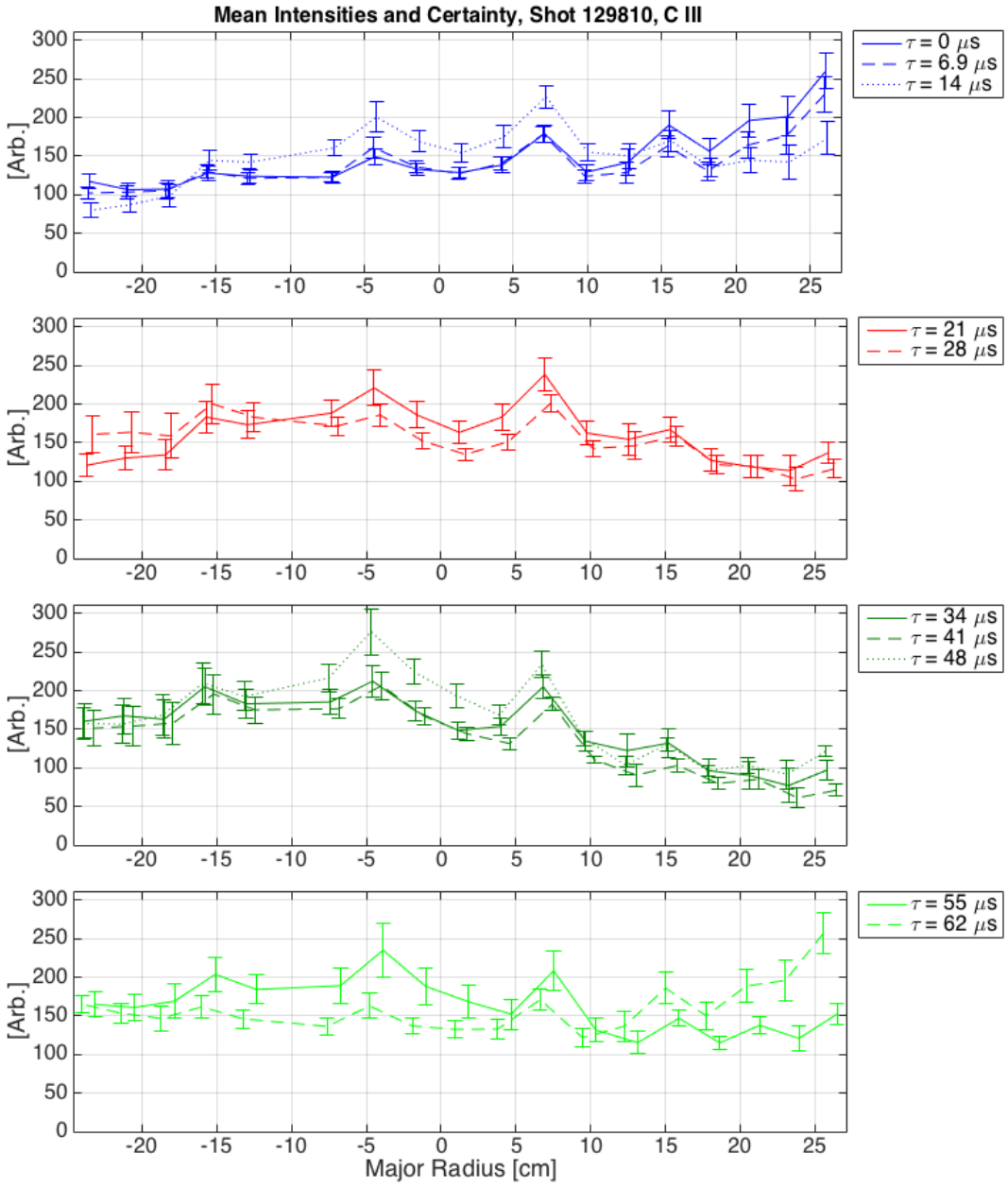

Figure 7.17: Mean, chord-averaged intensities and certainties for C III in shot 129810, separated by 1/4 injector cycle and color coded to match figure 7.13. Each of the ten lines is the mean and certainty of emission during a different 10^{th} of the injector cycle. The relative time within the 69 μs injector period is denoted by "τ". The data are selected during sustainment from $t = 1.6$ to 1.9 ms. Note the error bars are certainty ($\sigma/\sqrt{N}$) where $N = 4.6 \pm 0.5$ and σ includes fitting uncertainty and the standard deviation of the data.

7.4 Calculation of Radial Shift due to Flow

The pressure gradient due to rotation can be calculated using a rigid rotor model,

$$\nabla P = \rho R \Omega^2 \tag{7.2}$$

where ∇P is the pressure gradient, ρ is the mass density, R is the major radius, and Ω is the rotation frequency. The FIR interferometer was not operating during the shots when IDS data was taken, but the density is estimated to be 1×10^{19} m^{-3} [54]. The bulk ion is deuterium with mass $m_D = 3.34 \times 10^{-27}$ kg and the major radius is ~ 34 cm. To find the rotation frequency the toroidal velocity must be measured as close to the magnetic axis as is available. The toroidal flow presented in section 7.2.4 is 2.9 ± 2.8 km/s. With this value in equation 7.2, $\nabla P = 0.8 \pm 0.8$ Pa/m. Compared to the fitted pressure of 450 Pa/m, the observed velocity is clearly negligible. A toroidal velocity of ~ 68 km/s would be required to explain the magnetic profile shift.

7.5 Comparison between Injector Frequencies

7.5.1 Temperature Profiles

IDS data were collected through the reentrant port at toroidal location $\phi = 71°$ for all three injector frequencies considered (14.5, 53.5, and 68.5 kHz). IDS temperature data shown in sections 7.3.2, 7.1.3, and 7.2.3 are compiled in figure 7.18. Average temperatures and certainties for various regions are summarized in table 7.1.

Data Set	$\|R\| < 10$ cm	$\|R\| > 20$ cm	All Chords
14.5 kHz O II	8 ± 6	11 ± 10	9 ± 8
14.5 kHz C III	24 ± 5	25 ± 6	23 ± 5
53.5 kHz O II	29 ± 10	17 ± 6	24 ± 8
68.5 kHz O II	28 ± 6	20 ± 6	26 ± 6
68.5 kHz O II HG	42 ± 7	24 ± 4	33 ± 6

Table 7.1: Summary of temperatures and certainties in eV for regions and data sets listed. "HG" from high-gain, positive toroidal current shots.

The temperatures at 68.5 kHz are slightly higher than 53.5 kHz, and both are substantially higher than O II at 14.5 kHz. C III at 14.5 kHz, however, shows similar temperatures as O II at high frequency. All data sets display non-axismmetry, with higher frequency data being more pronounced. Note the similar profile between 53.5 kHz and 68.5 kHz even though they are compiled from shots with opposite toroidal current direction. This suggests the temperature profiles have at least a significant if not dominant contribution from the injector-linking plasma. This observation is consistent with the two data sets at 68.5 kHz which have opposite toroidal current, yet both show similar non-axisymmetry. No significant differences are apparent in the spatial profiles for different injector phases (see figures 7.4 and 7.8). For the high frequency data the higher temperatures at small major radii (inverse spheromak "temperature" gradient) and the larger fluctuations are indicative of injector flows near the geometric axis inflating the chord-averaged temperature. Assuming the outer region is more quiescent and dominated by the spheromak rather than injector plasma, and assuming a flat temperature profile, this means the injector flows are contributing ~ 12 eV to the chord-averaged temperatures for 53.5 kHz, ~ 8 eV for 68.5 kHz, and ~ 18 eV for high-gain 68.5 kHz.

7.5.2 Intensity Profiles

Intensity data has been compiled similarly in figure 7.19. The data from 53.5 kHz and 68.5 kHz have been scaled to account for their different exposure times. With this taken into account their emission levels are similar, with 68.5 kHz being slightly higher. It is not possible to calibrate between 14.5 kHz and high frequencies because of the different optics and image intensifier. No light source is available which is bright enough to simulate plasma emission on a relevant time scale. Thus, the 14.5 kHz data has been arbitrarily scaled to fit on the same plot. The spatial profiles are all relatively flat; the 53.5 kHz data set is the least symmetric and all data sets have a slight depression near the geometric axis. The most noticeable difference is the lower fluctuations at high frequency, summarized in table 7.2. Data from high-gain 68.5 kHz shots are excluded since the long exposure effectively cycle-averages over the dominant fluctuations.

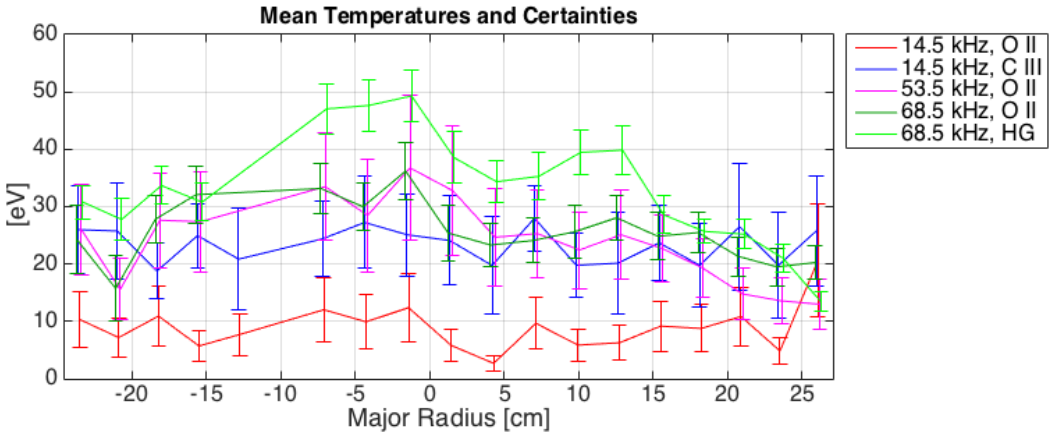

Figure 7.18: Mean temperatures and total certainties for all frequencies with IDS data including both impurity ions at 14.5 kHz. Data are compiled from figures 7.14, 7.15, 7.4, 7.8, and 7.9. Error bars include certainties of the means and fluctuations during times of interest, noted in their respective captions.

	14.5 kHz, O II	14.5 kHz, C III	53.5 kHz, O II	68.5 kHz, O II
$\delta I/I$	29%	33%	20%	22%

Table 7.2: Summary of intensity fluctuations for chords with impact parameter > 20 cm.

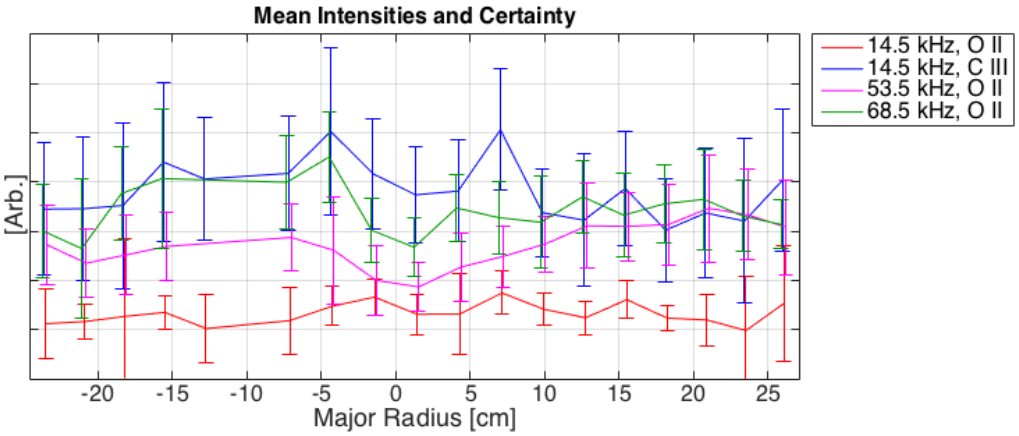

Figure 7.19: Mean intensities and total fluctuations for all frequencies with IDS data. Data are compiled from figures 7.16, 7.17, 7.5, and 7.10. Error bars include fluctuations of the means and uncertainties during times of interest, noted in their respective captions.

7.5.3 Scaling with Injector Frequency

To facilitate comparison between frequencies and with the FIR interferometer the data from figures 7.18 and 7.19 are compiled and plotted vs. characteristic injector time in figure 7.20. The O II temperature clearly increases with frequency but the data are too sparse and uncertain to constrain a scaling law. The temperature of C III at low frequency is higher than O II at high frequency. Measurements with the FIR interferometer showed that intensity fluctuations, $\delta n/n$, scale as

$$\frac{\delta n}{n} = A_0 \left(1 - exp\left(-\gamma/\omega_{inj}\right)\right) \tag{7.3}$$

where γ corresponds to a characteristic injector frequency of $\sim$30 kHz and A_0 is the amplitude in the limit of low injector frequency [54]. Interferometer data and the fit are reproduced in figure 7.20, right. Also plotted are IDS intensity and fluctuation data averaged and scaled assuming $I \propto n^2$. The IDS data for 68.5 kHz almost agrees with the fit to FIR but the 53.5 and 14.5 kHz fluctuations are clearly lower and do not agree. A fit to the

IDS data is also plotted, with γ corresponding to ∼65 kHz.

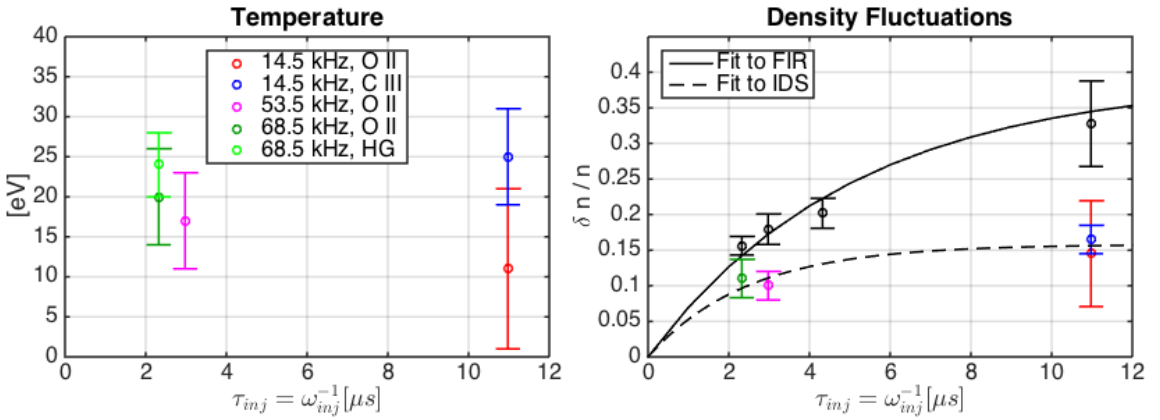

Figure 7.20: Mean temperatures for $|R| > 20$ cm plotted as a function of the characteristic injector time, τ_{inj} (left). Error bars include the variation between different chords and certainty. Density fluctuations are also plotted as a function of τ_{inj} (right). The black points are measurements from the FIR interferometer at $R \simeq 35$ cm and the solid line is a best fit including 5.8 kHz data [54]. The colored points are IDS intensity data scaled assuming $I \propto n^2$ for chords at $|R| > 20$ cm. The black, dashed line is a best fit to IDS data only. Error bars are calculated in the same way as for temperature.

7.5.4 Estimating Plasma β

To calculate β according to equation 7.1, plasma density and magnetic field are needed in addition to temperature from the IDS. The FIR interferometer was not operating for the 53.5 kHz shots already presented, so n_e and B are taken from the most similar shot available, 128533. The density and magnetic field are time-averaged over the same interval as the IDS data, from 1.5 to 2.0 ms. For the 68.5 kHz data set with current gain of 3 the interferometer was only on for shot 129225, so density and magnetic field are taken from this shot and time-averaged over the same interval as IDS: 1.6 to 1.85 ms. The FIR was not operating for any of the high-gain discharges presented, so the most similar shot available, 129175, is used instead and the density averaged from $t = 1.3 - 1.5$ ms. As mentioned in section 5.1, the FIR was not able to operate reliably for high-current discharges, but Brian

Victor estimates the density to be $6 \pm 1 \times 10^{19}$ m^{-3} for shot 129810.

Combining these data and their uncertainties yields estimates for $< \beta >_{wall}$ as shown in table 7.3. One assumption in these calculations is that the impurity ions are in thermal equilibrium with the bulk deuterium ions. Estimates of the ion-ion thermal equilibration time [26] are ~ 2 μs for high-gain 68.5 kHz data, ~ 1 μs for 68.5 kHz data with gain of 3, and less for all others. Another assumption is that the ions and electrons are in thermal equilibrium, which may not be true. The estimated electron-ion thermalization times [26] are also listed in table 7.3 and range from 42 to 820 μs, significant portions of the plasma discharge.

	14.5 kHz, O II	14.5 kHz, C III	53.5 kHz, O II	68.5 kHz, O II	68.5 kHz, HG
$< T_i >$ [eV]	11 ± 10	25 ± 6	17 ± 6	20 ± 6	24 ± 4
$< n_e >$ [m^{-3}]	$6 \pm 1 \times 10^{19}$	$6 \pm 1 \times 10^{19}$	$2.3 \pm 0.5 \times 10^{19}$	$1.2 \pm 0.2 \times 10^{19}$	$8.0 \pm 1.3 \times 10^{18}$
$< \lvert B \rvert >_{wall}$ [mT]	68.6 ± 3.2	68.6 ± 3.2	24.9 ± 0.8	17.9 ± 0.5	21.3 ± 0.8
$< \beta >_{wall}$ [%]	11 ± 10	26 ± 8	51 ± 21	58 ± 20	34 ± 8
τ_{ei}^{Th} [ms]	0.042 ± 0.039	0.13 ± 0.04	0.19 ± 0.08	0.45 ± 0.15	0.82 ± 0.19
$< \beta >_{vol}$ [%]	6 ± 5	13 ± 4	26 ± 11	29 ± 10	17 ± 4
$< \beta >_{vol}$ GS [%]	3 - 7	3 - 7		21 - 29	21 - 29

Table 7.3: Summary of parameters and βs for data sets listed. T_is are taken from table 7.1 for $|R| > 20$ cm. $< |B| >_{wall}$ is calculated from the toroidal current: $\mu_0 I_{tor}/2\pi a$, where $a = 0.233$ m. $< \beta >_{vol}$ is estimated to be 1/2 of $< |B| >_{wall}$ and $< \beta >_{wall}$ GS is taken from reference [54].

For spheromaks, $< \beta >_{wall}$ is approximately twice the volume-averaged beta, $< \beta >_{vol}$. Thus, the $< \beta >$ values for high frequency are above the volume-averaged Mercier beta limit of 10% (corresponding to $< \beta >_{wall}$ of $\sim$20% [30]). Additionally, these values are in agreement with those found by Grad-Shafranov (GS) fitting to internal magnetic probe data in reference [54], as shown by the last two lines of table 7.3. The 14.5 kHz O II data agrees with the GS fits, the 68.5 kHz data with gain of 3 is above the GS range but agrees, and the high-gain β falls just below the GS range but barely agrees. The sigificant difference in β between the two 68.5 kHz data sets is due to lower density and slightly higher magnetic field at high gain, illustrating the sensitivity in calculated β values.

7.6 Discussion

The results presented here support the conclusion that high frequency operations of HIT-SI yielded increased plasma pressure confinement and high β. The lack of significant toroidal rotation eliminates the hypothesis of a profile shift due to a centrifugal force. The $< \beta >_{vol}$ values calculated from IDS data at 68.5 kHz (17 ± 4 to $29 \pm 10\%$) are in agreement with those values found by GS fitting (21 – 29%). This highlights the variability of direct β measurements, mostly due to density variation during discharges.

An advantage offered by direct β measurements is better certainty of axisymmetry. A potential problem with the internal magnetic probe is that it measures fields at one toroidal location, so a strong $n = 2$ structure, for example, could shift the magnetic profile at the probe, but not over the entire spheromak. The IDS measurements are from several chords on either side of the geometric axis, so the high β values calculated are not likely to be caused by a local anomaly.

There is disagreement between the density fluctuations as measured by the FIR and IDS at 14.5 kHz. The FIR interferometer is a more direct instrument for measuring density, so the IDS results do not challenge the reported fluctuation scaling [54]. The implication is that ion emission, especially at 14.5 kHz, depends on more than density (e.g.: ionization fraction and temperature). Additionally, the IDS measurements are from chords at impact parameters $20 - 26$ cm whereas the interferometer's chord is at 35 cm. Thus, the density fluctuations may be lower on chords at lower impact parameter.

Chapter 8

CONCLUSIONS

Measurements of ion temperature, velocity, and emission have expanded our knowledge of plasma dynamics in HIT-SI and contributed toward the validation of simulations.

High-current, 14.5 kHz shots exhibit flat, chord-averaged temperature profiles which increase during the toroidal current ramp but decrease during sustainment. The significant difference between C III and O II is an outstanding problem that can only be solved by dynamic modeling of impurity ionization and energy balance. Previous work established the existence of "separate current" [53] but left open the possibility of a true separatrix and closed flux surfaces. Temperature and/or toroidal velocity gradients are indicators of closed flux. Analysis of IDS data fails to show either, although a temperature gradient is possible because analysis of simulations shows that a strong temperarure gradient can be obscured by chord-averaging. The lack of rotation is expected because the injector geometry makes the plasma nonaxisymmetric, inhibiting rotation. Additionally, the time-varying injector fluxes do not drive a toroidal rotation. The IDS flow measurements contributed to the design of the HIT-SI3 injectors which are less perturbative and can be phased to drive plasma rotation.

Analysis of chord-averaged velocity fluctuations yields strong evidence for a separate, stable spheromak. The velocity results showing coherent, rigid motion combined with the lack of magnetic instabilities are an important confirmation of the step-λ profile predicted by IDCD theory. The measured transition radius is consistent with IDCD and indicates a larger separate region than predicted by Taylor equilibria.

Comparing temperatures and velocities with simulations has given direction for improving the simulations. Limited agreement of the phase and amplitude of oscillations is a step toward validating the velocity fields. The simulated velocities are far less axisymmetric than magnetic fields and dominated by vortical injector flows. The simulated plasmas do

not exhibit the same rigid motion observed in HIT-SI and thus do not capture the IDCD step-λ profile. Strong flows on chords beyond the magnetic axis and severe reconnection events both indicate that higher viscosity may be needed to improve agreement. Work is underway to implement neutral dynamics in NIMROD and PSI-TET and new viscosity scans are being undertaken. Also, the discovery that differential velocities in the simulations contribute $\sim$5 eV to chord-averaged temperatures improves our understanding of the true ion temperature.

The results for high injector frequency support previous work [54] and bolster the argument that these plasmas have high-β pressure confinement and lower fluctuations. The evidence includes higher temperatures, lower intensity fluctuations, and higher $< \beta >_{wall}$. Additionally, the negligible toroidal velocity negates the possibility of the outward shift being due to rotation. The 68.5 kHz result agrees with Grad-Shafranov (GS) fitting to the internal magnetic probe and surface probes.

Chapter 9

SUGGESTIONS FOR FUTURE WORK

Since HIT-SI has been shut down, further experimental study is unlikely. Thus, refinement of the results presented in this work must be through simulations. As discussed in chapter 6, viscosity scans for finite-β simulations are likely to yield improved velocity agreement. Initial zero-β simulations show increasing fluctuation coherence with higher viscosity, as shown in figure 9.1. These initial results also show decreased velocity amplitudes on outer chords. The NIMROD and PSI-TET codes show significant disagreement with chord-averaged temperatures, so further verification between the codes is required. The disagreement may be due to the very different injector modeling schemes. The intensities yielded the worst agreement with simulated densities. Finding a way to run simulations with more realistic diffusivities may yield better agreement.

Neither NIMROD or PSI-TET have produced fully validated simulations at injector frequencies greater than 14.5 kHz, although they are getting close. Once those codes produce validated magnetic profiles, the velocity, temperature, and density profiles will be invaluable for interpreting IDS data.

Another path to improving agreement is to better model the emission. The OpenADAS [2] spectral database may be applicable. This database contains reaction rates and emissivity coefficients for various ions including C III and O II. Thus, given a local density and temperature the ionization balance and emission can be calculated. The models assume that the plasmas are in generalized collisional-radiative (GCR) equilibrium, but MHD simulations of HIT-SI do not satisfy this regime. Flows are dominated by injector dynamics which are faster than the timescales required. A previous study [12] employed similar modeling for time-averaged quantities during sustainment of the SPHEX spheromak.

The IDS is presently collecting data on HIT-SI3 with the same viewing configuration as on HIT-SI (see figure 9.2). Thus far the data has shown injector-dominated velocity

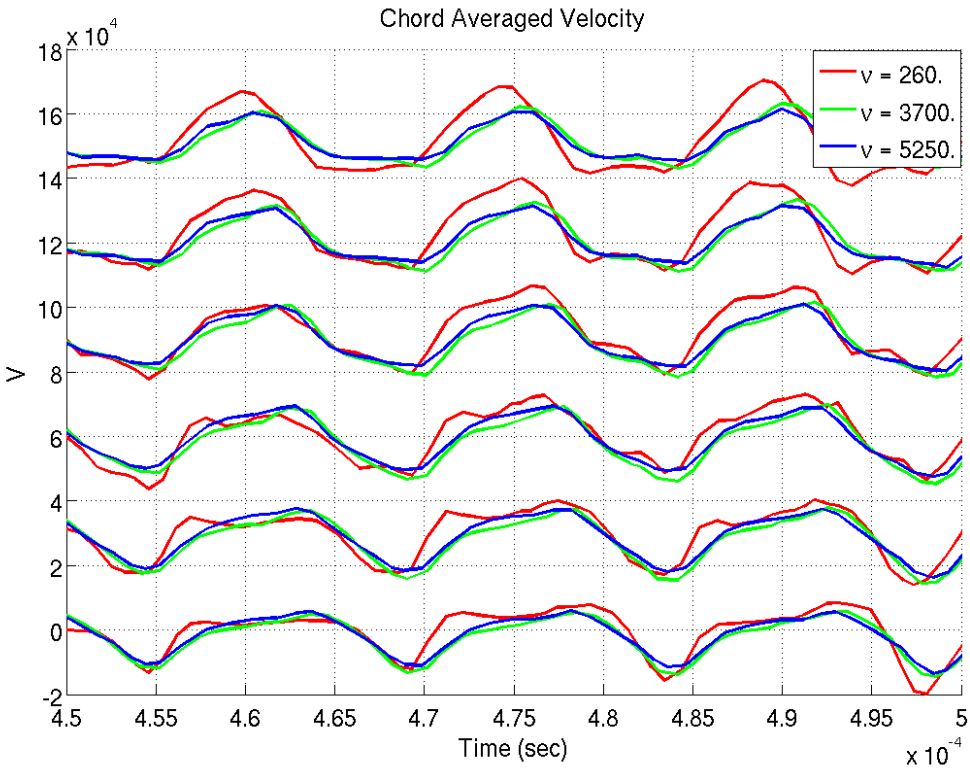

Figure 9.1: Chord-averaged velocities from NIMROD for selected chords (From the top, $R \simeq 39$, 33, 28, 23, 18, and 12 cm). Horizontal lines mark zero-velocities and chords are spaced by 2 km/s. Figure courtesy of Kyle Morgan.

oscillations and no toroidal flow, but HIT-SI3 has not achieved comparable current gain to HIT-SI. The HIT-SI3 injectors were designed partly to drive plasma rotation. As HIT-SI3 increases injector frequency and current gain it will be important to monitor toroidal velocities. Learning from experience on HIT-SI, a valuable addition for HIT-SI3 would be another toroidally viewing reentrant port on the equivalent port *below* the pumping-tee. Simultaneously viewing from $R \simeq -41$ to $+41$ cm would improve confidence in toroidal flow measurements and claims regarding nonaxisymmetric fluctuations. This would require using the fiber normally used for poloidal viewing, but this fiber has not produced much useful data due to its weaker signal because of shorter chords.

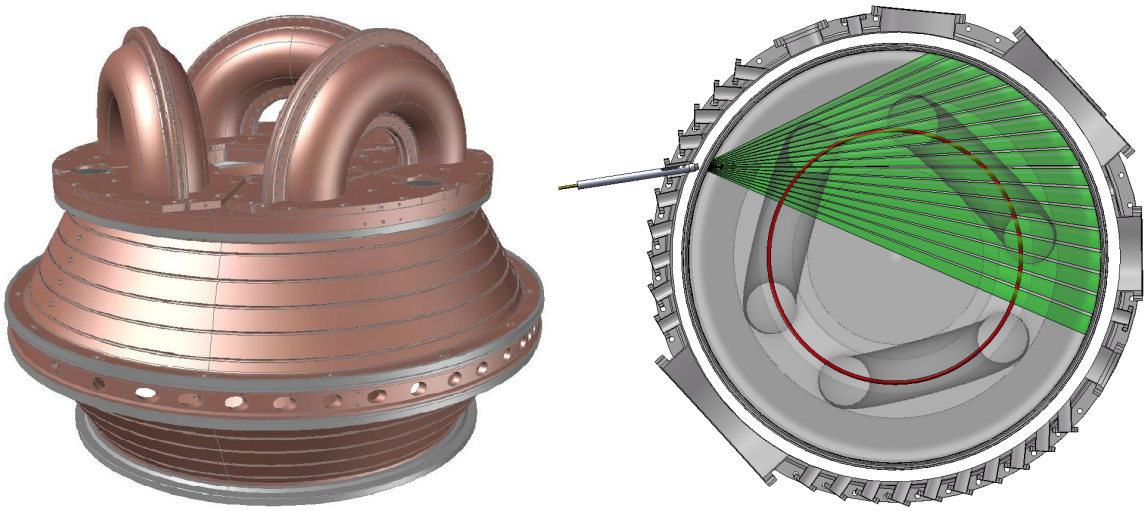

Figure 9.2: CAD rendering of the HIT-SI3 copper flux conserver (left) and IDS viewing configuration (right). IDS chords (green) and magnetic axis (red) are represented in the vacuum chamber viewed from the "injector side". Semi-transparent injectors are superimposed.

BIBLIOGRAPHY

[1] ITER: The way to new energy. `http://www.iter.org/`.

[2] OPEN-ADAS `http://open.adas.ac.uk/`.

[3] C. Akcay. *Extended Magnetohydrodynamic Simulations of the Helicity Injected Torus (HIT-SI) Spheromak Experiment with the NIMROD Code*. PhD thesis, University of Washington, 2013.

[4] C. Akcay, C.C. Kim, B.S. Victor, and T.R. Jarboe. Validation of single-fluid and two-fluid magnetohydrodynamic models of the helicity injected torus spheromak experiment with the NIMROD code. *Physics of Plasmas*, 20(8), 2013.

[5] S.C. Bates and K.H. Burrell. Fast gas injection system for plasma physics experiments. *Review of Scientific Instruments*, 55(6):934–939, 1984.

[6] M.A. Berger and G.B. Field. The topological properties of magnetic helicity. *Journal of Fluid Mechanics*, 147:133–148, 1984.

[7] P.I. Bernstein and N. Gerami. Proliferation risks of civilian nuclear power programs. In *Center for the Study of Weapons of Mass Destruction*. National Defense University, June 2012.

[8] H.A.B. Bodin and A.A. Newton. Reversed-field-pinch research. *Nuclear Fusion*, 20(10):1255, 1980.

[9] F.F. Chen. Experiments on helicon plasma sources. *Journal of Vacuum Science & Technology A*, 10(4):1389–1401, 1992.

[10] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G.H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E.W. Bethel, D. Camp, O. Rübel, M. Durant, J.M. Favre, and P. Navrátil. VisIt: An end-user tool for visualizing and analyzing very large data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

[11] R.W. Conn, J.P. Holdren, S. Sharafat, D. Steiner, D.A. Ehst, W.J. Hogan, R.A. Krakowski, R.L. Miller, F. Najmabadi, and K.R. Schultz. Economic, safety and environmental prospects of fusion reactors. *Nuclear Fusion*, 30(9):1919, 1990.

[12] G. Cunningham. Impurities and impurity transport in the spheromak SPHEX. *Plasma physics and controlled fusion*, 39(9):1339, 1997.

[13] T.D. de Wit, A.L. Pecquet, J.C. Vallet, and R. Lima. The biorthogonal decomposition as a tool for investigating fluctuations in plasmas. *Physics of plasmas*, 1:3288, 1994.

[14] D.A. Ennis, B.S. Victor, J.S. Wrobel, C. Akcay, T.R. Jarboe, G.J. Marklin, B.A. Nelson, and R.J. Smith. New understandings and achievements from independent-injector drive experiments on HIT-SI. *Nuclear Fusion*, 50(7):072001, 2010.

[15] J.C. Fernandez, B.L. Wright, G.J. Marklin, D.A. Platts, and T.R. Jarboe. The m= 1 helicity source spheromak experiment. *Physics of Fluids B: Plasma Physics*, 1:1254, 1989.

[16] N.J. Fisch. Theory of current drive in plasmas. *Rev. Mod. Phys.*, 59:175–234, Jan 1987.

[17] M. Greenwald. Density limits in toroidal plasmas. *Plasma Physics and Controlled Fusion*, 44(8):R27, 2002.

[18] W.T. Hamp. *Temperature and Density Characteristics of the Helicity Injected Torus-II Experiment Indicating Evidence of Closed Flux Sustainment Using Coaxial Helicity Injection*. PhD thesis, University of Washington, 2008.

[19] Chris Hansen. Private communication, 2014.

[20] C.J. Hansen. *MHD Modeling in Complex Geometries: Towards Predictive Simulation of SIHI Current Drive*. PhD thesis, University of Washington, 2014.

[21] J. Hemminger. New science for a secure and sustainable energy future. Technical report, U.S. Department of Energy, 2008.

[22] C.T. Holcomb. Master's thesis, University of Washington, 1999.

[23] E.B. Hooper, R.H. Bulmer, B.I. Cohen, D.N. Hill, C.T. Holcomb, B. Hudson, H.S. McLean, L.D. Pearlstein, C.A. Romero-Talamás, C.R. Sovinec, et al. Sustained spheromak physics experiment (SSPX): design and physics results. *Plasma Physics and Controlled Fusion*, 54(11):113001, 2012.

[24] E.B. Hooper, L.D. Pearlstein, and R.H. Bulmer. MHD equilibria in a spheromak sustained by coaxial helicity injection. *Nuclear Fusion*, 39(7):863, 1999.

[25] A.C. Hossack, T. Firman, T.R. Jarboe, J.R. Prager, B.S. Victor, J.S. Wrobel, and T. Ziemba. Reduction of plasma density in the helicity injected torus with steady inductance experiment by using a helicon pre-ionization source. *Review of Scientific Instruments*, 84(10):103506, 2013.

[26] J.D. Huba. NRL: Plasma formulary. Technical report, DTIC Document, 2007.

[27] I.H. Hutchinson. *Principles of plasma diagnostics*. Cambridge university press, 2005.

[28] A. Janos, G.W. Hart, and M. Yamada. Relaxation of spheromak plasmas toward a minimum-energy state and global magnetic fluctuations. *Physical review letters*, 55(26):2868, 1985.

[29] T.R. Jarboe. Review of spheromak research. *Plasma Physics and Controlled Fusion*, 36(6):945, 1994.

[30] T.R. Jarboe. Steady inductive helicity injection and its application to a high-beta spheromak. *Fusion Science and Technology*, 36(1):85–91, 1999.

[31] T.R. Jarboe. The spheromak confinement device. *Physics of Plasmas*, 12(5):058103, 2005.

[32] T.R. Jarboe, I. Henins, A.R. Sherwood, C.W. Barnes, and H.W. Hoida. Slow formation and sustainment of spheromaks by a coaxial magnetized plasma source. *Physical review letters*, 51(1):39, 1983.

[33] T.R. Jarboe, B.S. Victor, B.A. Nelson, C.J. Hansen, C. Akcay, D.A. Ennis, N.K. Hicks, A.C. Hossack, G.J. Marklin, and R.J. Smith. Imposed-dynamo current drive. *Nuclear Fusion*, 52(8):083017, 2012.

[34] H. Ji, A.F. Almagri, S.C. Prager, and J.S. Sarff. Time-resolved observation of discrete and continuous magnetohydrodynamic dynamo in the reversed-field pinch edge. *Physical review letters*, 73(5):668, 1994.

[35] E. Kawamori, Y. Murata, K. Umeda, D. Hirota, T. Ogawa, T. Sumikawa, T. Iwama, K. Ishii, T. Kado, T. Itagaki, et al. Ion kinetic effect on bifurcated relaxation to a field-reversed configuration in TS-4 CT experiment. *Nuclear fusion*, 45(8):843, 2005.

[36] K Miyamoto. *Controlled Fusion and Plasma Physics*. Taylor and Francis Group, 2007.

[37] Kyle Morgan. Private communication, 2014.

[38] M. Nagata, T. Kanki, T. Masuda, S. Naito, H. Tatsumi, and T. Uyama. Relaxation oscillations and toroidal-current regeneration in a helicity-driven spheromak. *Phys. Rev. Lett.*, 71:4342–4345, Dec 1993.

[39] F. Najmabadi, A. Abdou, L. Bromberg, T. Brown, V.C. Chan, M.C. Chu, F. Dahlgren, L. El-Guebaly, P. Heitzenroeder, D. Henderson, et al. The ARIES-AT advanced tokamak, advanced technology fusion power plant. *Fusion Engineering and Design*, 80(1):3–23, 2006.

[40] D.I. Naydenkova, F. Janky, V. Weinzettl, J. Stockel, D. Sestak, T. Odstrcil, J. Ghosh, R. Gomes, and T. Pereira. Measurements of ion temperature in the edge plasma of the COMPASS tokamak. In *WDS'11 Proceedings of Contributed Papers: Part II*, pages 233–236, 2011.

[41] R.G. O'Neill. *An Experimental Study of Helicity Injection Current Drive in the HIT-SI Spheromak*. PhD thesis, University of Washington, 2007.

[42] M.L. Parry, O. F. Canziani, J. P. Palutikof, P.J. van der Linden, and C.E. Hanson, editors. *Climate Change 2007: Impacts, Adaptations and Vulnerability*. Cambridge University Press, 2007.

[43] M.G. Rusbridge, S.J. Gee, P.K. Browning, G. Cunningham, R.C. Duck, A. Al-Karkhy, R. Martin, and J.W. Bradley. The design and operation of the SPHEX spheromak. *Plasma physics and controlled fusion*, 39(5):683, 1997.

[44] U. Shumlak and T.R. Jarboe. Stable high beta spheromak equilibria using concave flux conservers. *Physics of Plasmas*, 7:2959, 2000.

[45] P.E. Sieck. *Spheromak Formation Using Steady Inductive Helicity Injection*. PhD thesis, University of Washington, 2006.

[46] C.R. Sovinec, A.H. Glasser, T.A. Gianakon, D.C. Barnes, R.A. Nebel, S.E. Kruger, D.D. Schnack, S.J. Plimpton, A. Tarditi, and M.S. Chu. Nonlinear magnetohydrodynamics simulation using high-order finite elements. *Journal of Computational Physics*, 195(1):355–386, 2004.

[47] D.A. Sutherland, T.R. Jarboe, K.D. Morgan, M. Pfaff, E.S. Lavine, Y. Kamikawa, M. Hughes, P. Andrist, G. Marklin, and B.A. Nelson. The dynomak: An advanced spheromak reactor concept with imposed-dynamo current drive and next-generation nuclear power technologies. *Fusion Engineering and Design*, 89(4):412–425, 2014.

[48] J.B. Taylor. Relaxation of toroidal plasma and generation of reverse magnetic fields. *Physical Review Letters*, 33(19):1139, 1974.

[49] J.B. Taylor. Relaxation and magnetic reconnection in plasmas. *Rev. Mod. Phys.*, 58:741–763, Jul 1986.

[50] R. Toschi. Nuclear fusion, an energy source. *Fusion engineering and design*, 36(1):1–8, 1997.

[51] M. Tuszewski. Field reversed configurations. *Nuclear Fusion*, 28(11):2033, 1988.

[52] B.S. Victor. Density analysis of the HIT-SI spheromak. Master's thesis, University of Washington, 2009.

[53] B.S. Victor. *Effects of Density Control on Internal Plasma Dynamics and Current Drive in HIT-SI.* PhD thesis, University of Washington, 2012.

[54] B.S. Victor, T.R. Jarboe, C.J. Hansen, C. Akcay, K.D. Morgan, A.C. Hossack, and B.A. Nelson. Sustained spheromaks with ideal n= 1 kink stability and pressure confinement. *Physics of Plasmas*, 21(8):082504, 2014.

[55] B.S. Victor, T.R. Jarboe, A.C. Hossack, D.A. Ennis, B.A. Nelson, R.J. Smith, C. Akcay, C.J. Hansen, G.J. Marklin, N.K. Hicks, and J.S. Wrobel. Evidence for separatrix formation and sustainment with steady inductive helicity injection. *Phys. Rev. Lett.*, 107:165005, Oct 2011.

[56] J.S. Wrobel. *A Study of HIT-SI Plasma Dynamics Using Surface Magnetic Field Measurements.* PhD thesis, University of Washington, 2011.

[57] S. You. The transport of relative canonical helicity. *Physics of Plasmas*, 19(9):092107, 2012.

[58] T. Ziemba, J. Carscadden, J. Slough, J. Prager, and R. Winglee. High power helicon thruster. In *41th Joint Propulsion Conference, Tucson, AR*, 2005.

Appendices

Appendix 1

SUPPLEMENTAL DATA

A.1 Fast Ion Gauge (FIG) Measurements of Gas Injection

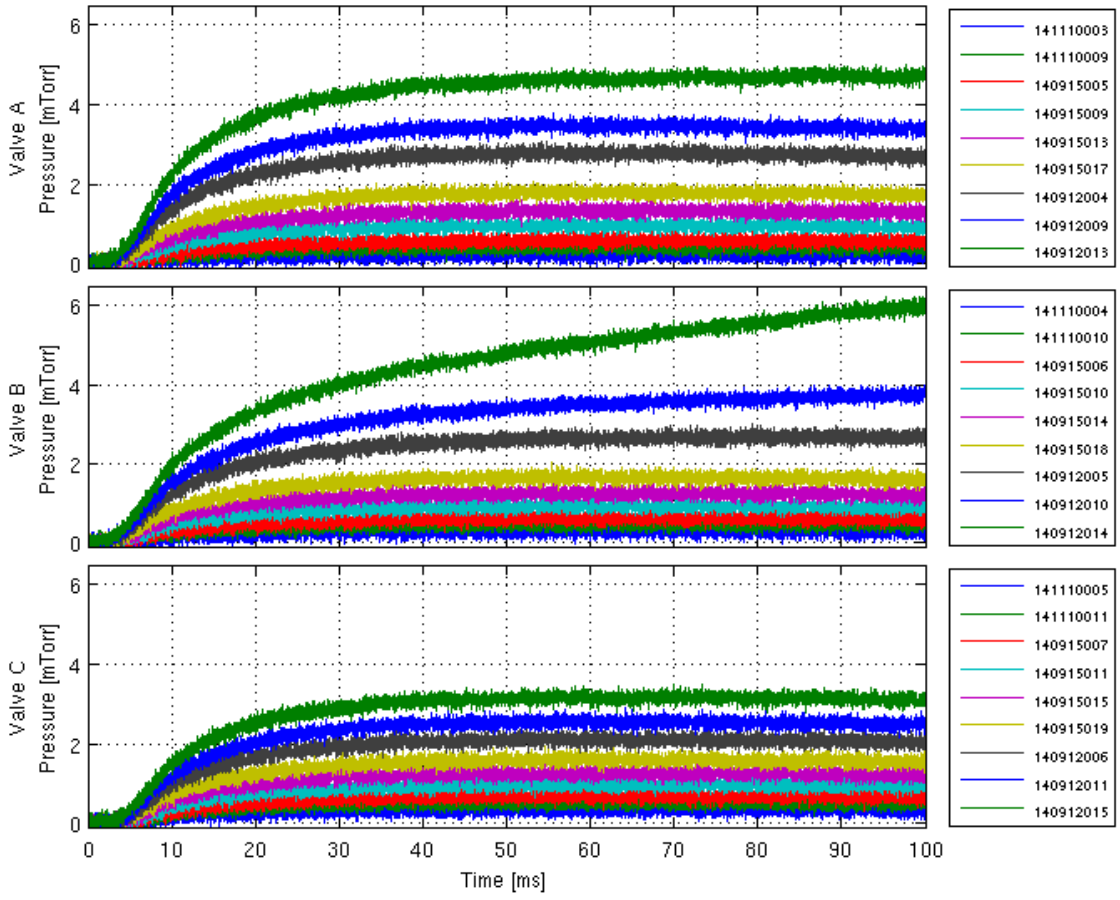

Figure A.1: Pressures in the HIT-SI3 vacuum tank as measured by the FIG with shot numbers displayed in the legends. The highest pressure traces had line pressures of 5000 Torr. For successively lower traces the pressure was lowered in increments of 500 Torr to 1000 Torr.

A.2 IDS Velocity Measurements used in FFT Filtering

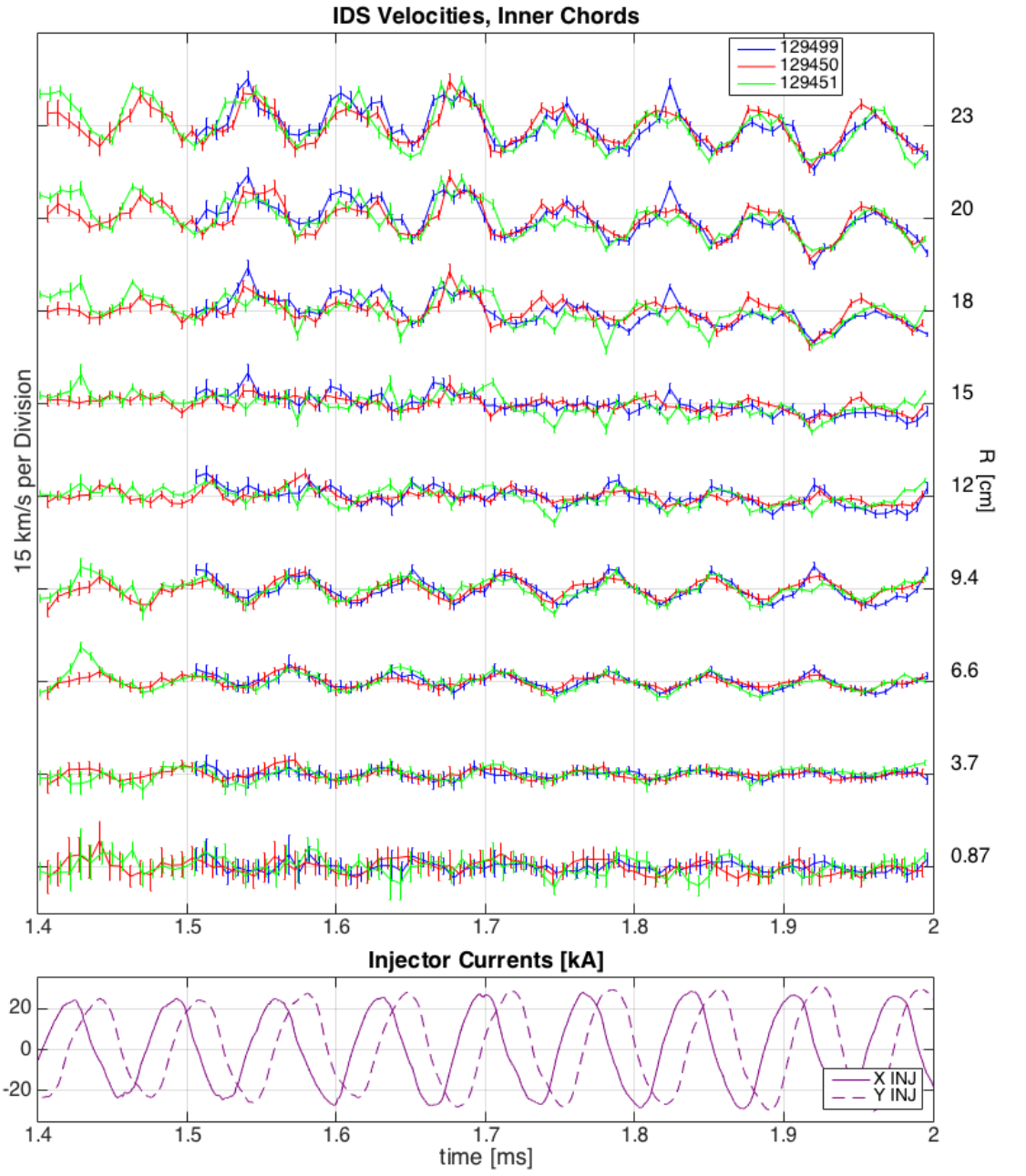

Figure A.2: Full timespan of velocities used in FFT filtering to obtain fundamental velocity components and displacements.

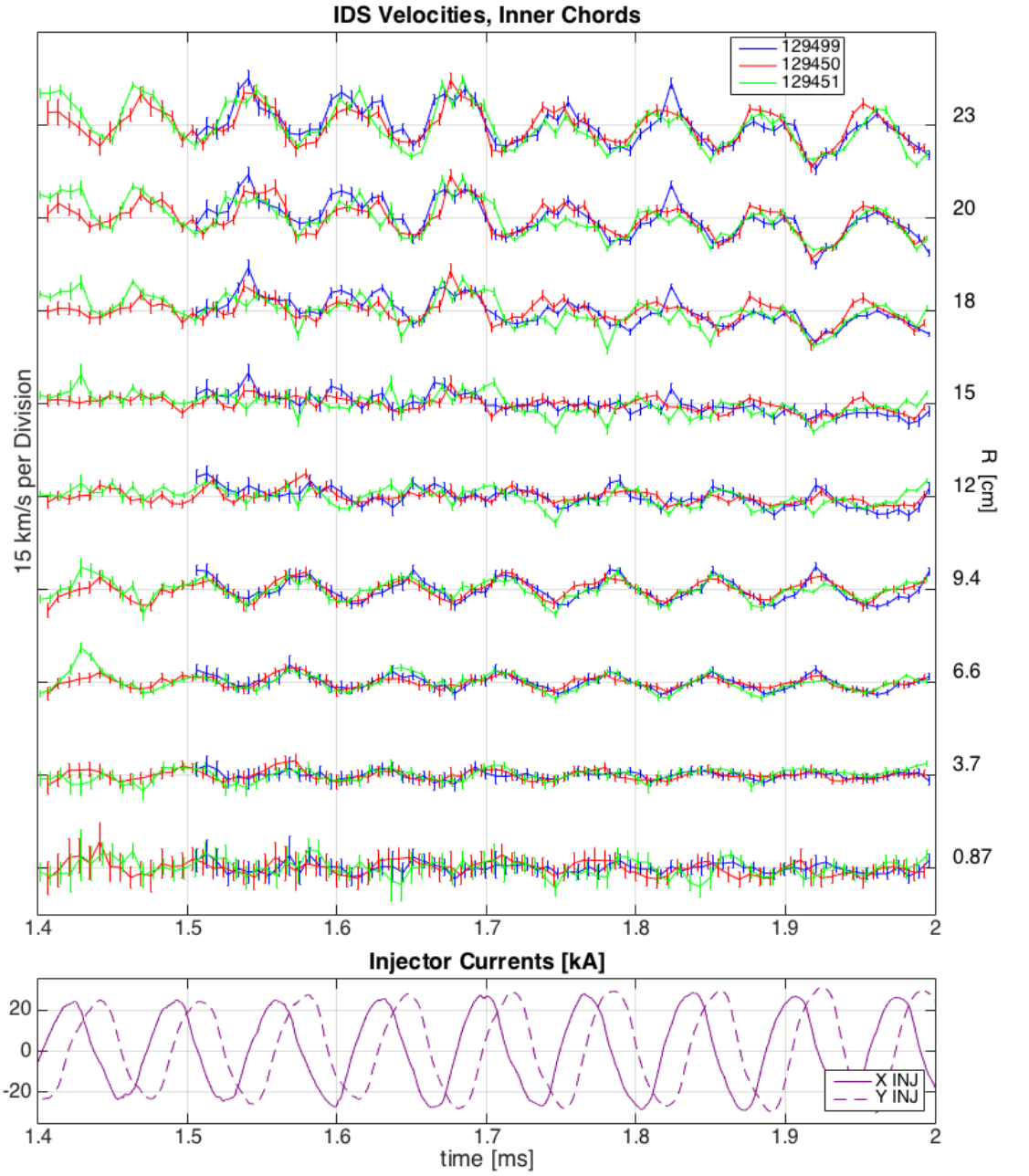

Figure A.3: Full timespan of velocities used in FFT filtering to obtain fundamental velocity components and displacements.

Appendix 2

GENERAL DATA FROM HIT-SI

B.1 Shot 129499

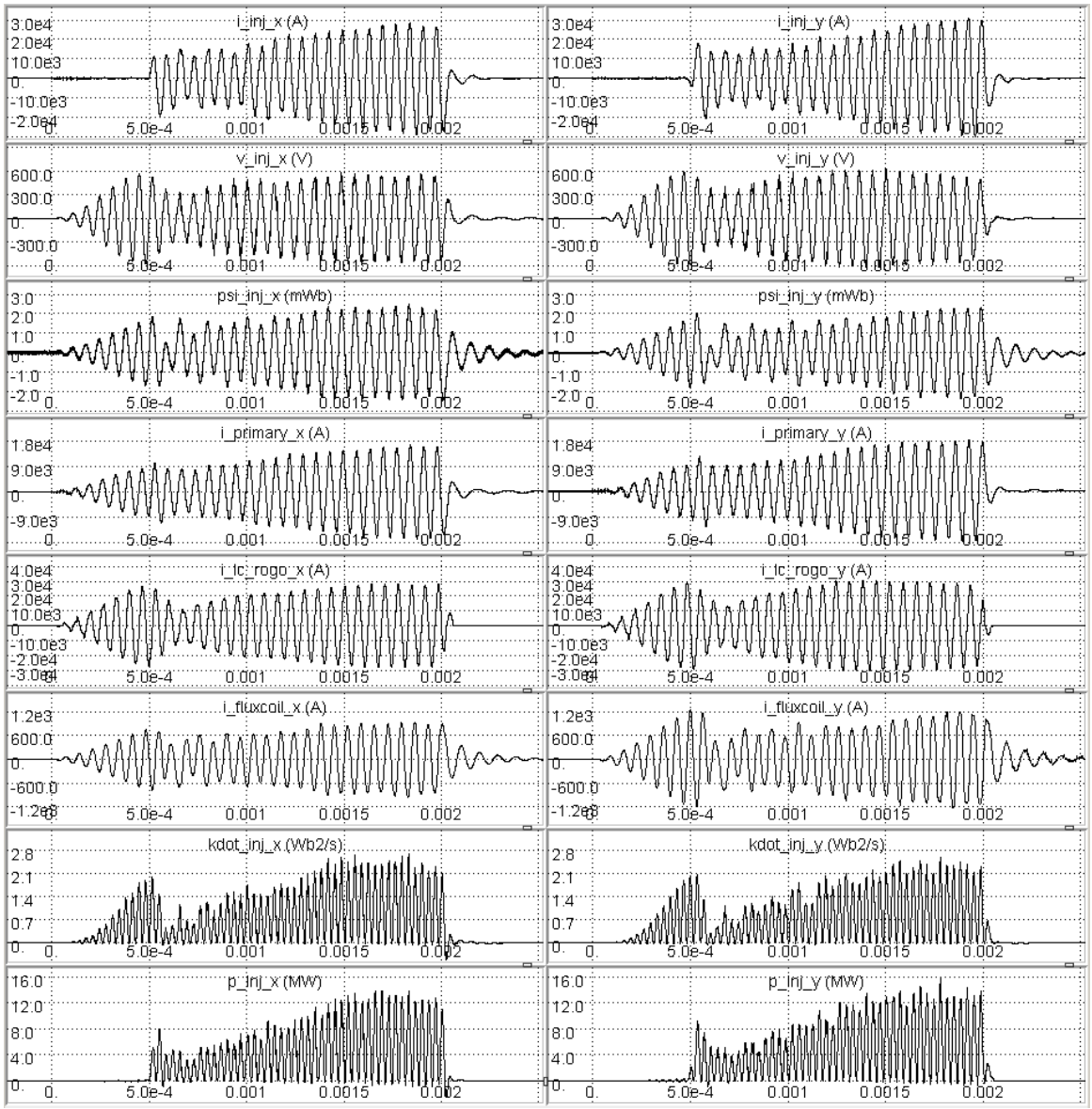

Figure B.1: Injector signals, shot 129499.

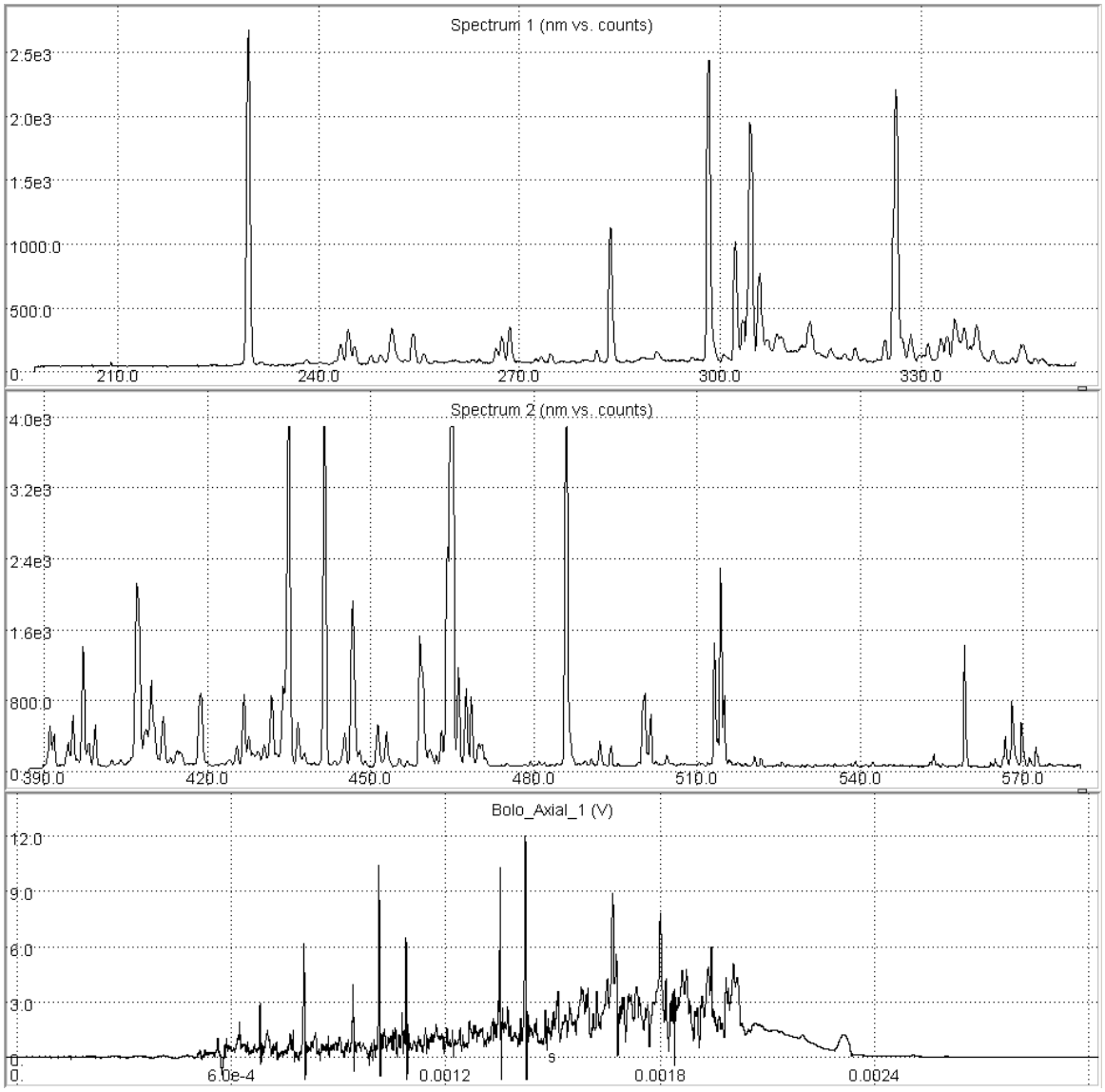

Figure B.2: SPRED spectrum and bolometer signal, shot 129499.

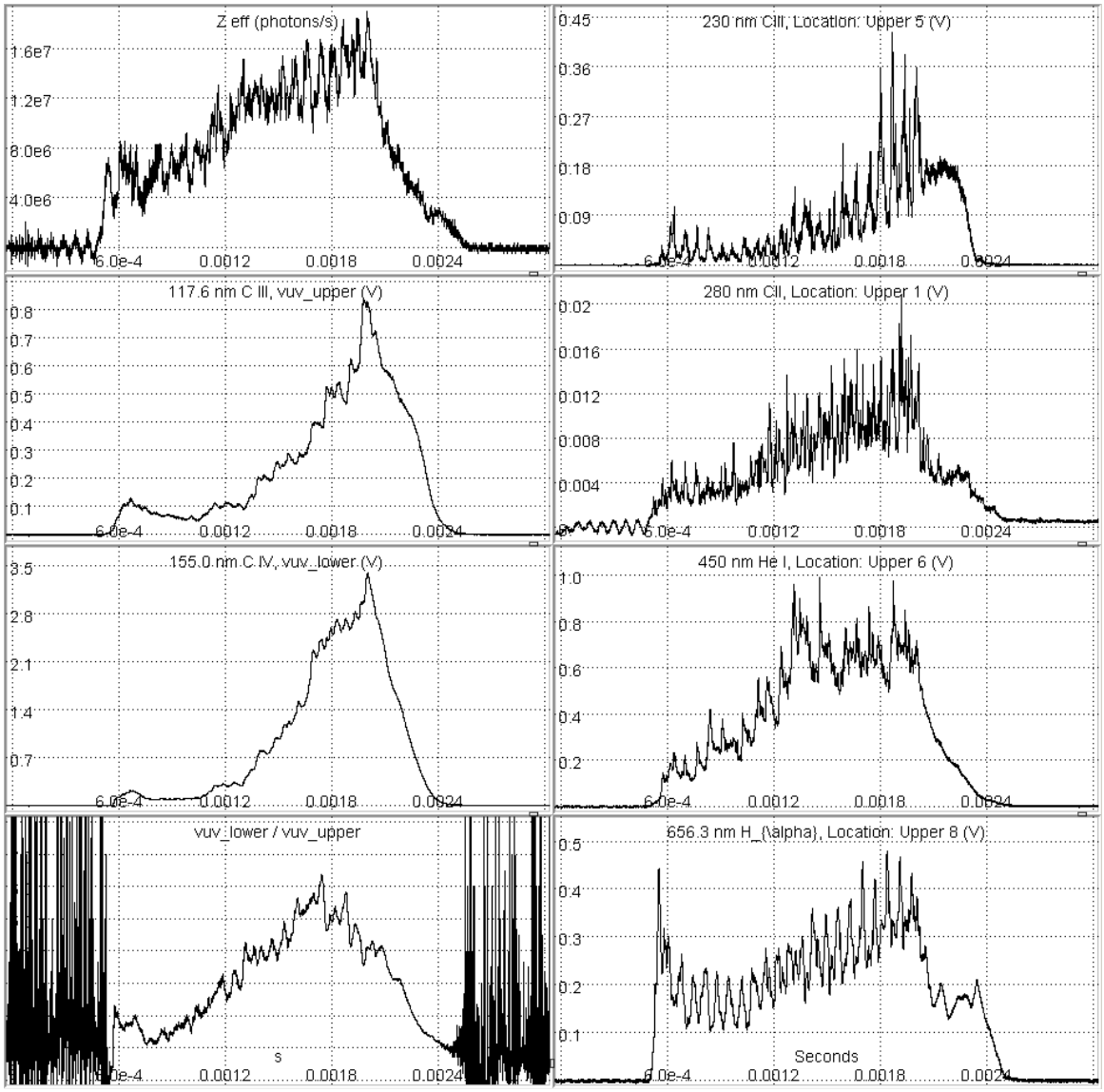

Figure B.3: Spectral signals, shot 129499.

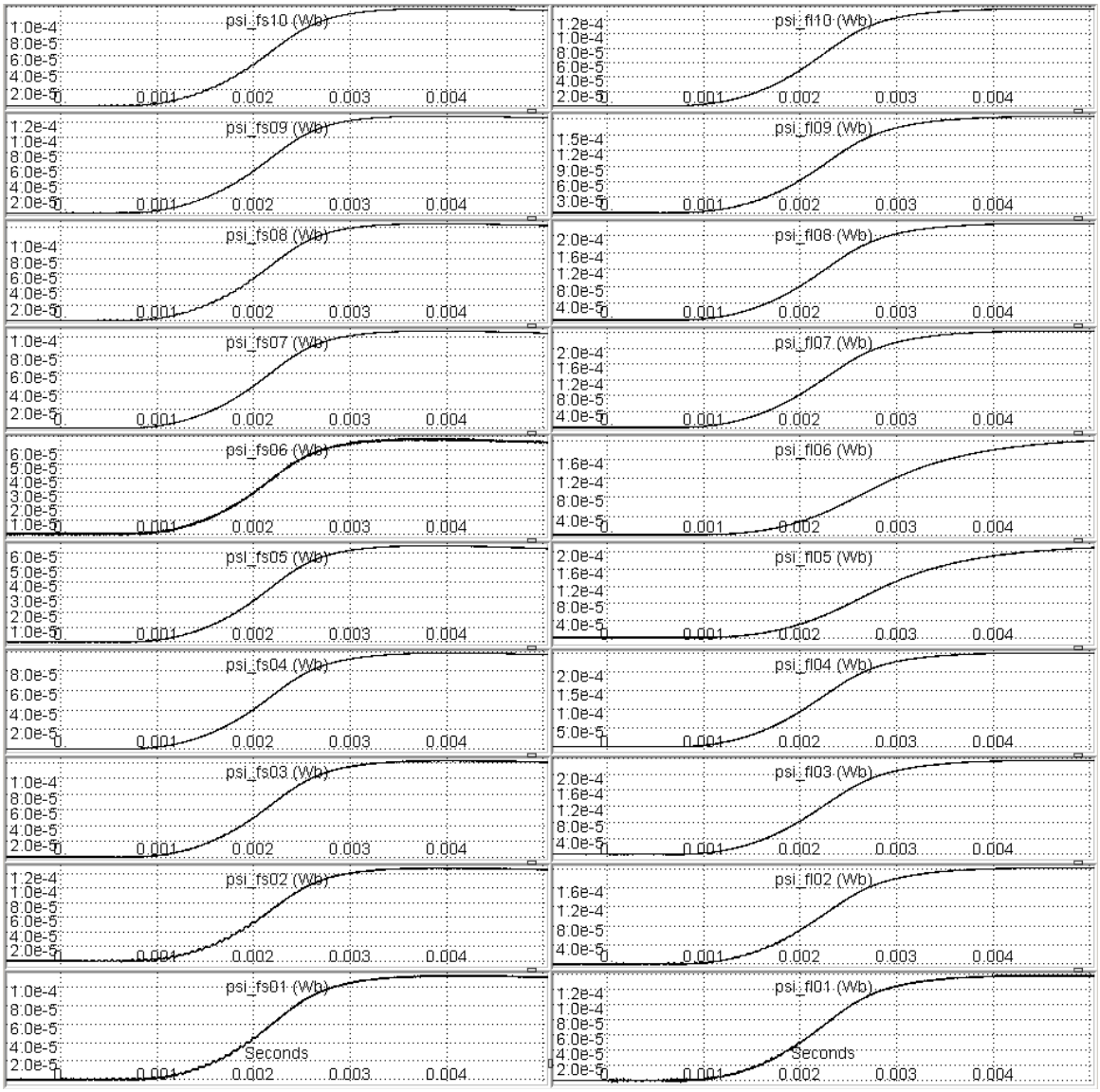

Figure B.4: Flux loop signals, shot 129499.

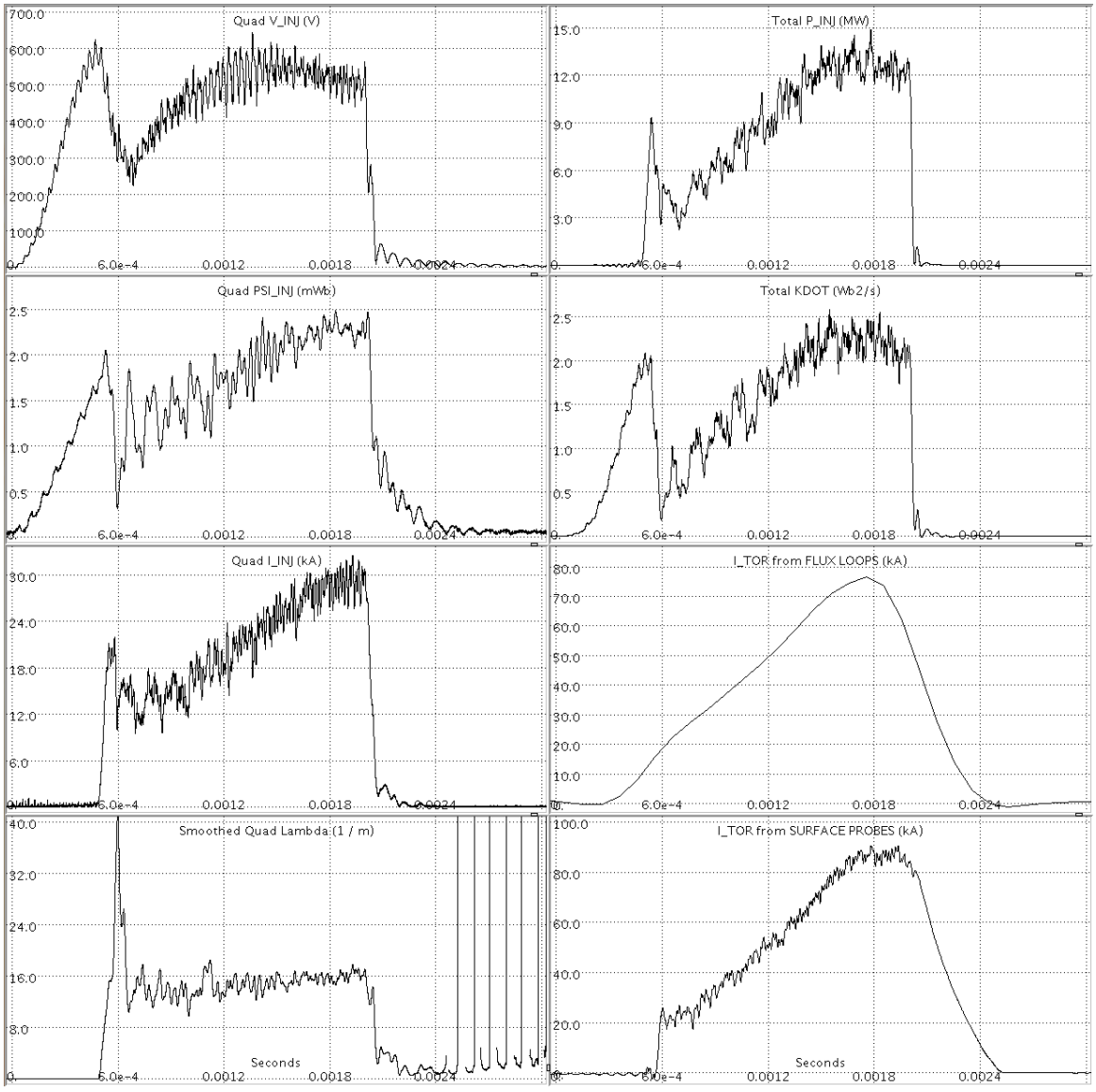

Figure B.5: Calculated parameters, shot 129499.

B.2 Shot 129496

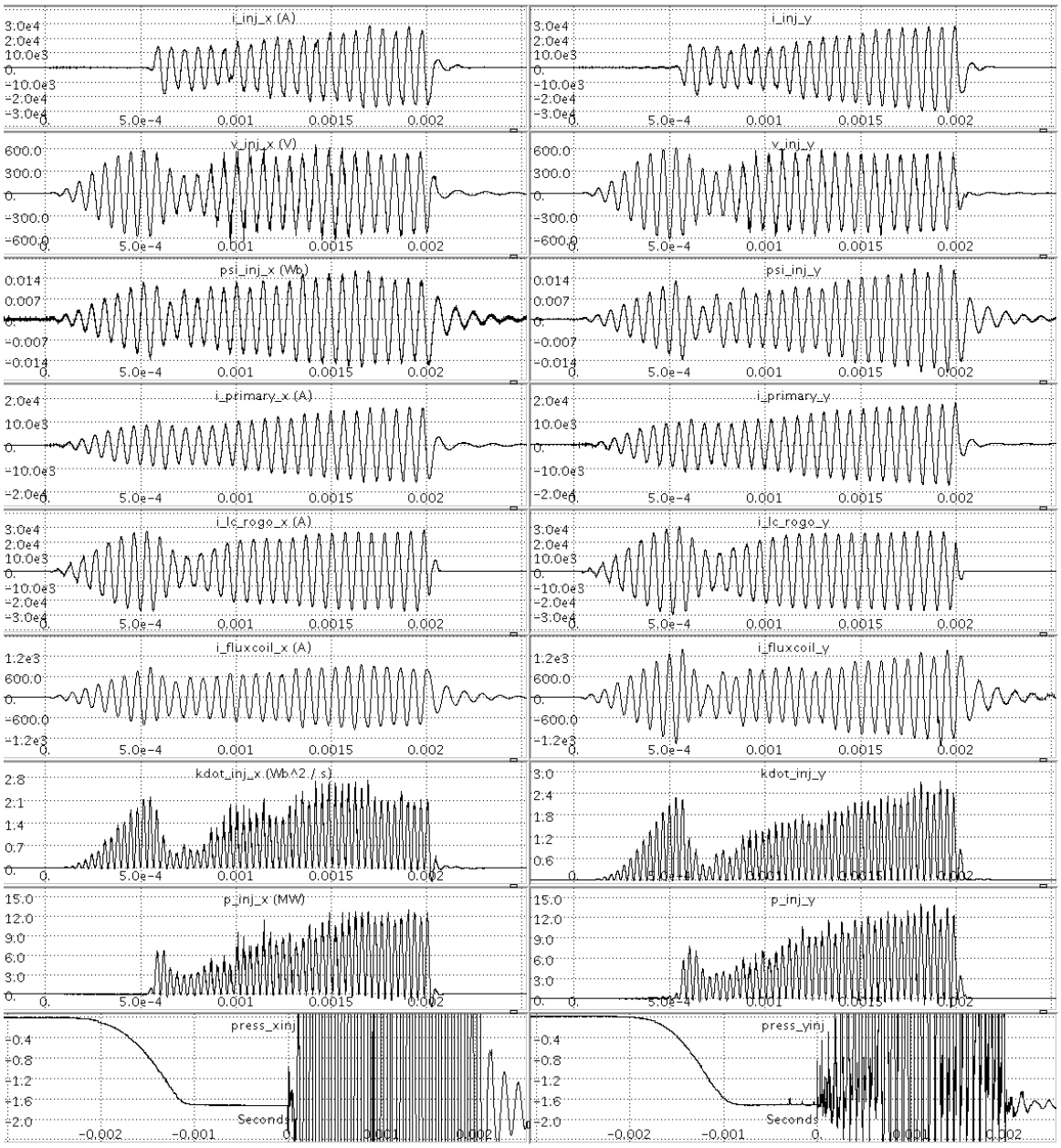

Figure B.6: Injector signals, shot 129496.

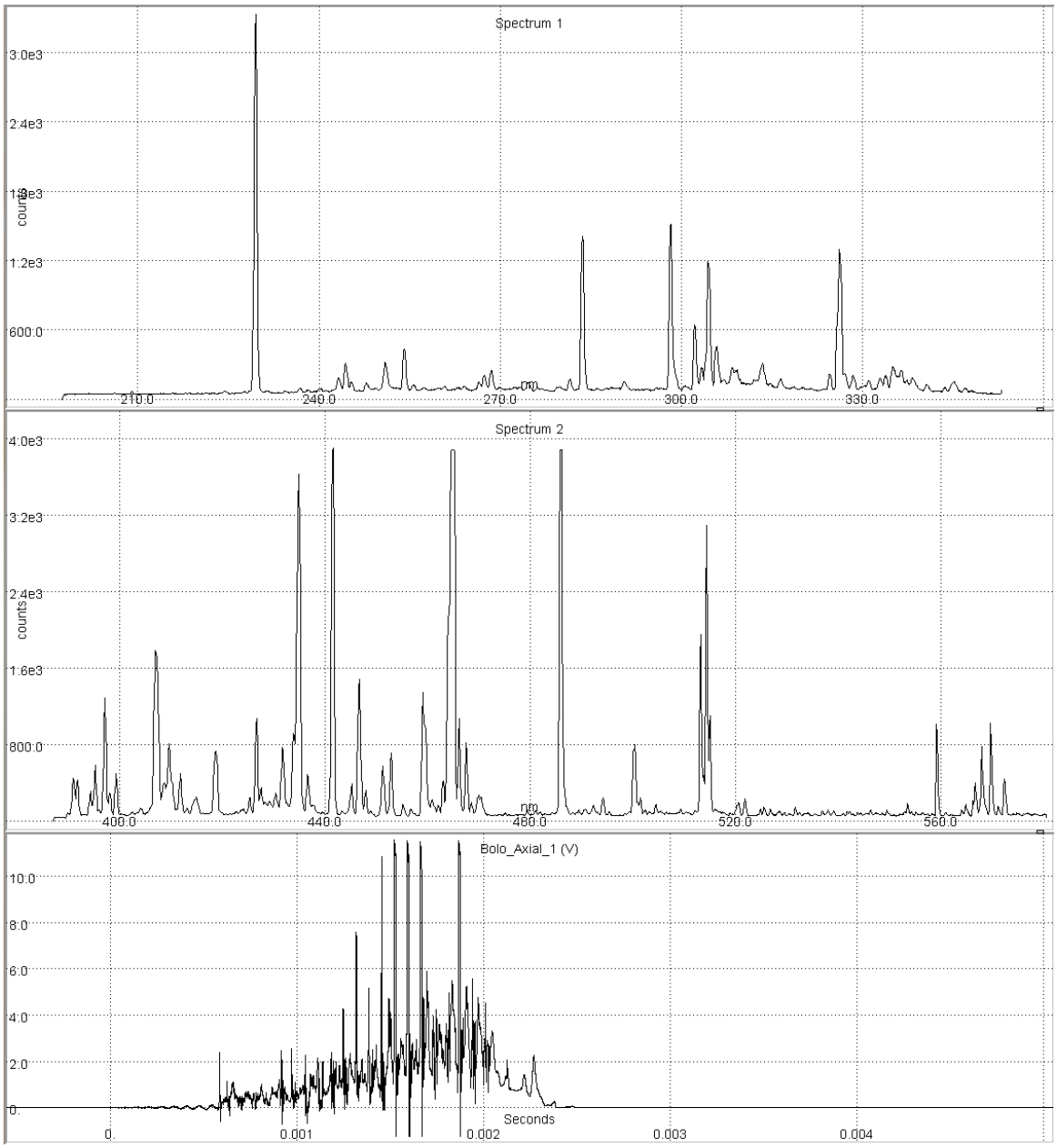

Figure B.7: SPRED spectrum and bolometer signal, shot 129496.

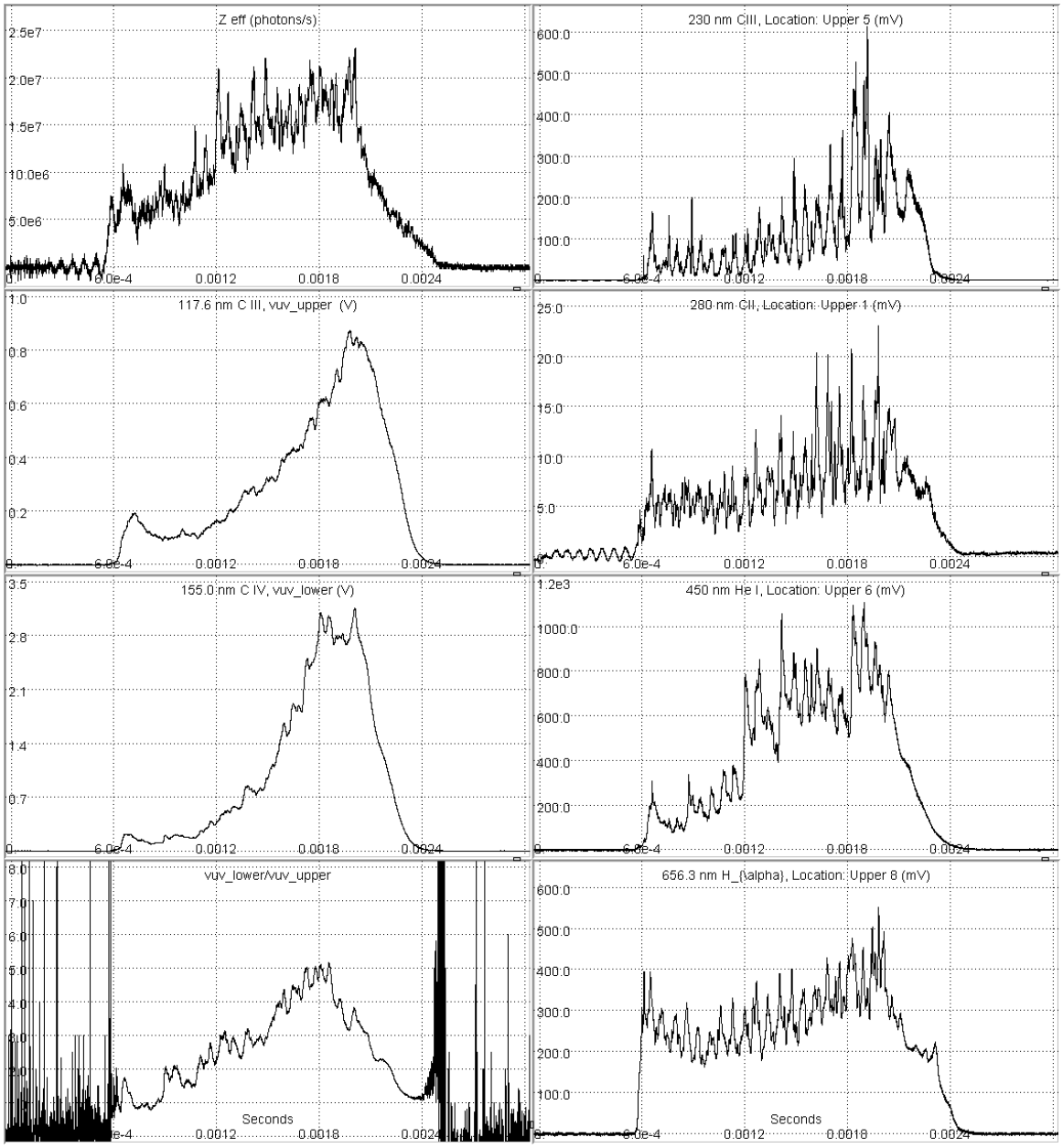

Figure B.8: Spectral signals, shot 129496.

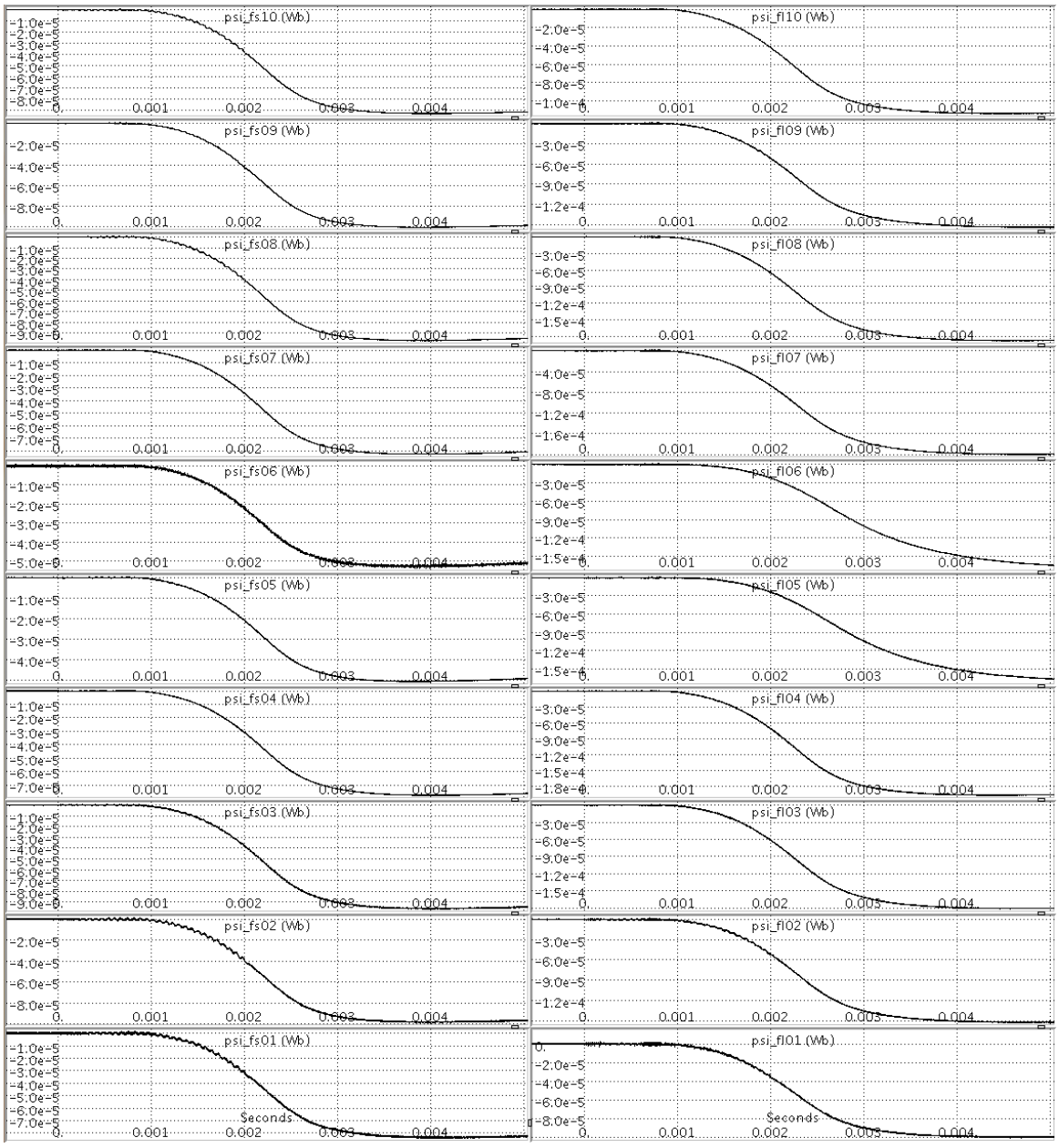

Figure B.9: Flux loop signals, shot 129496.

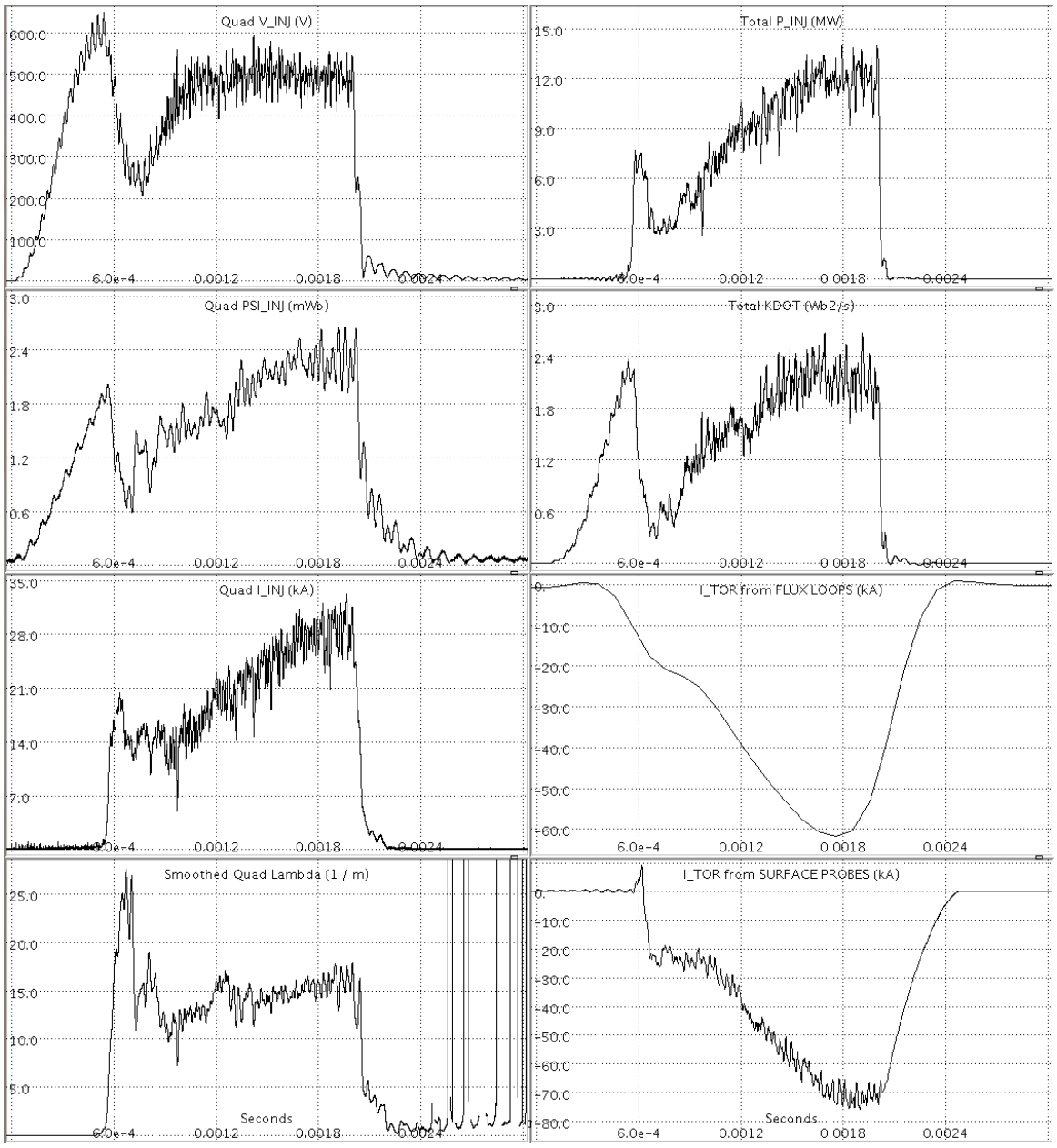

Figure B.10: Calculated parameters, shot 129496.

B.3 Shot 128580

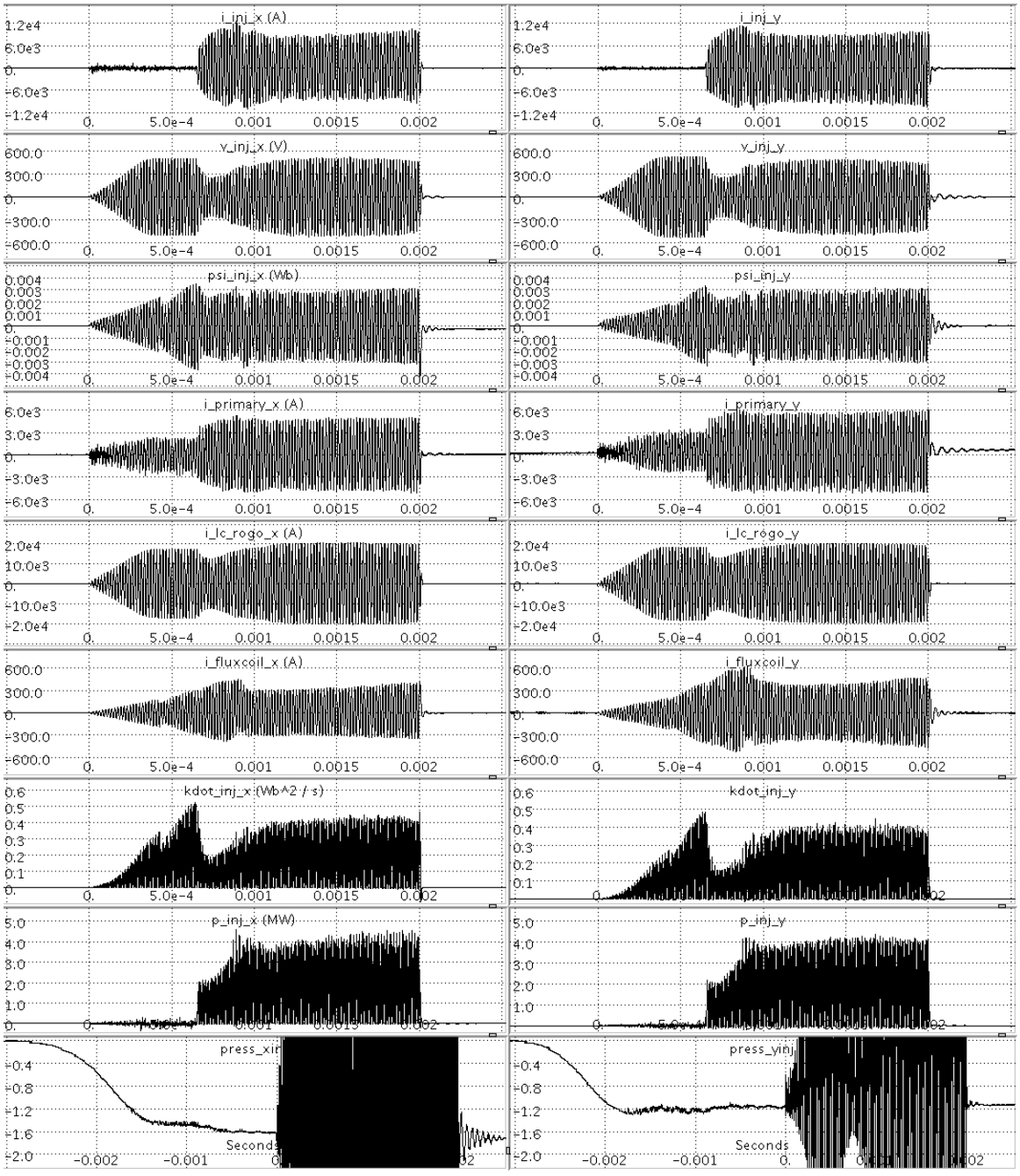

Figure B.11: Injector signals, shot 128580.

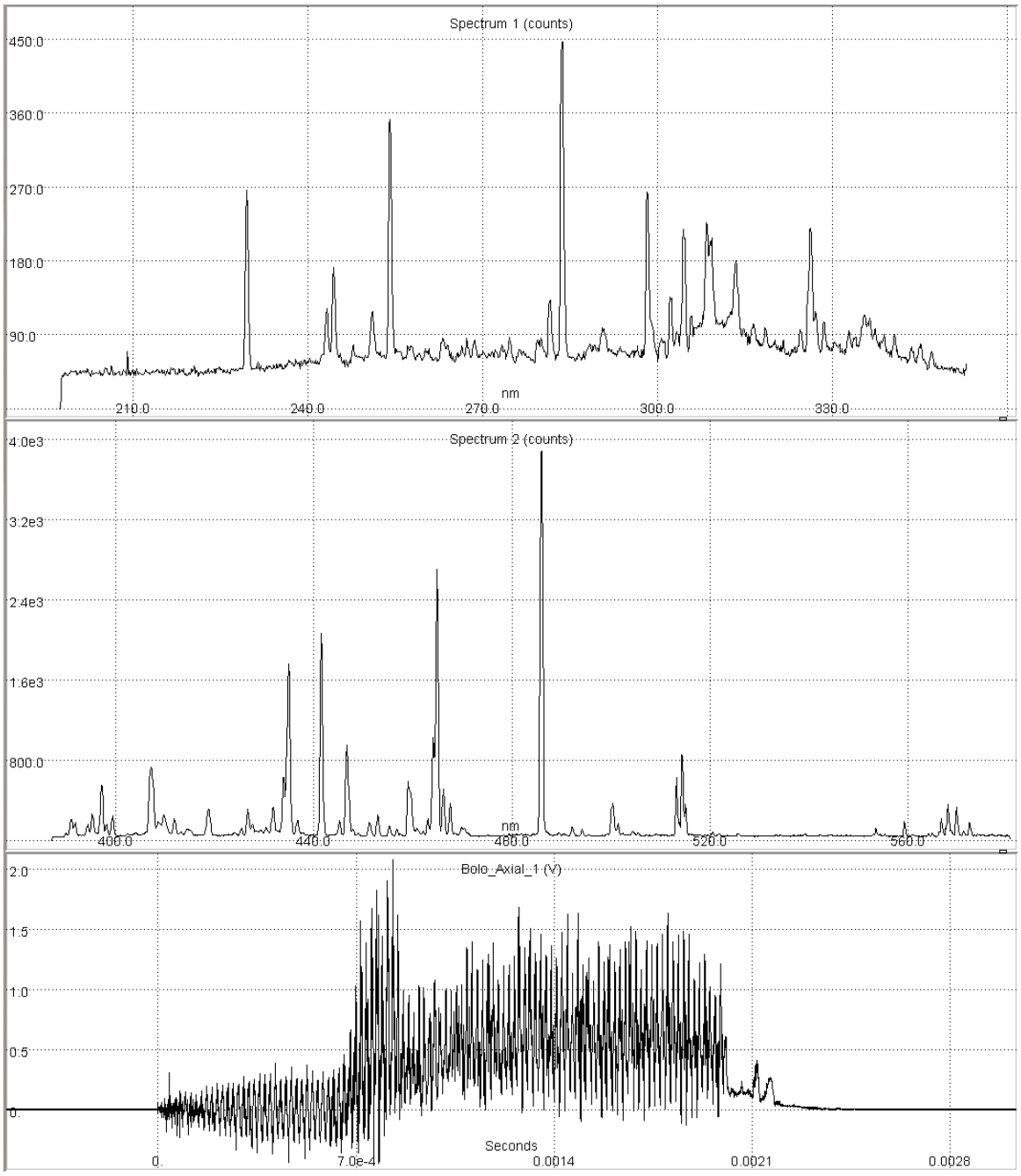

Figure B.12: SPRED spectrum and bolometer signal, shot 128580.

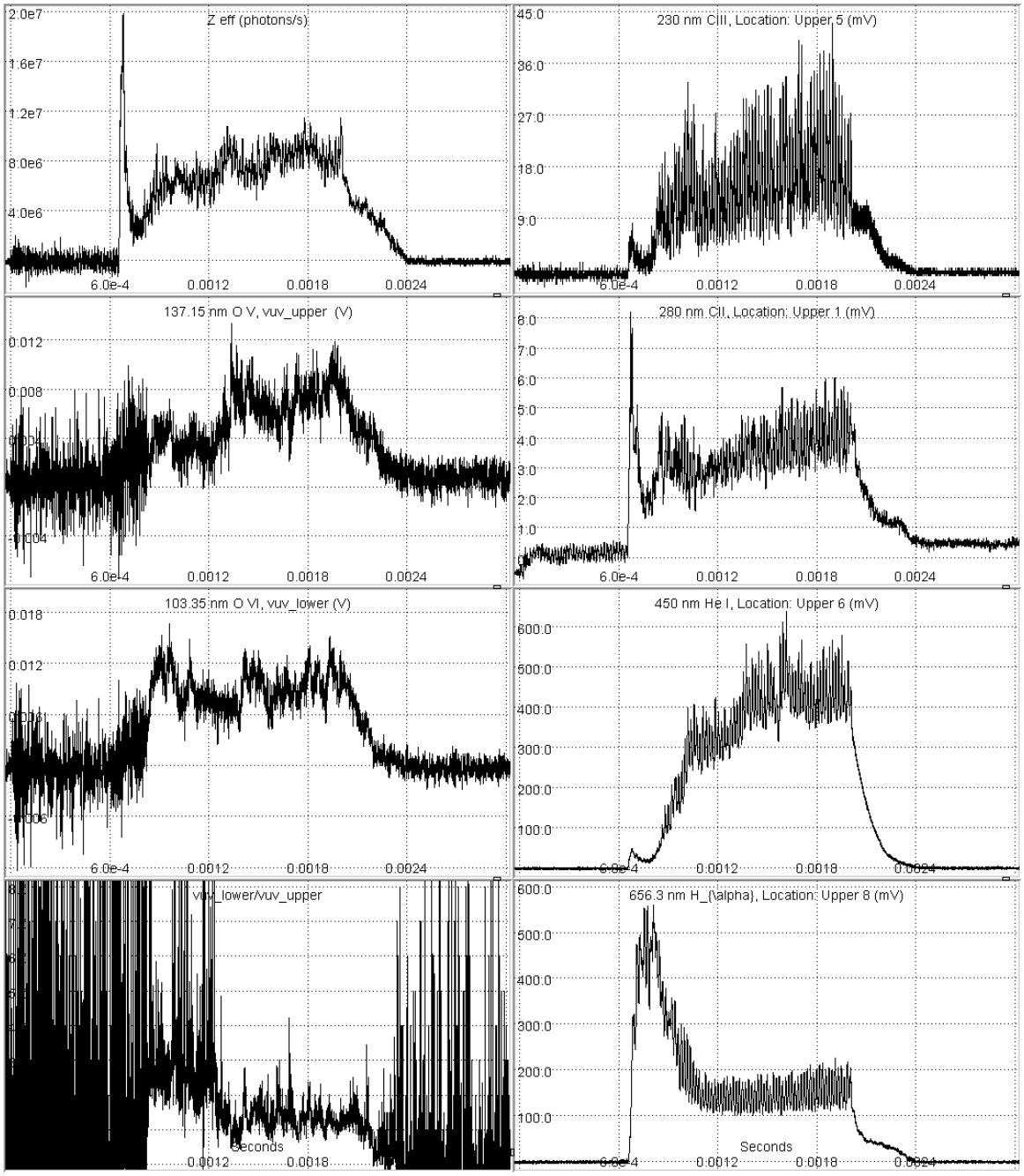

Figure B.13: Spectral signals, shot 128580.

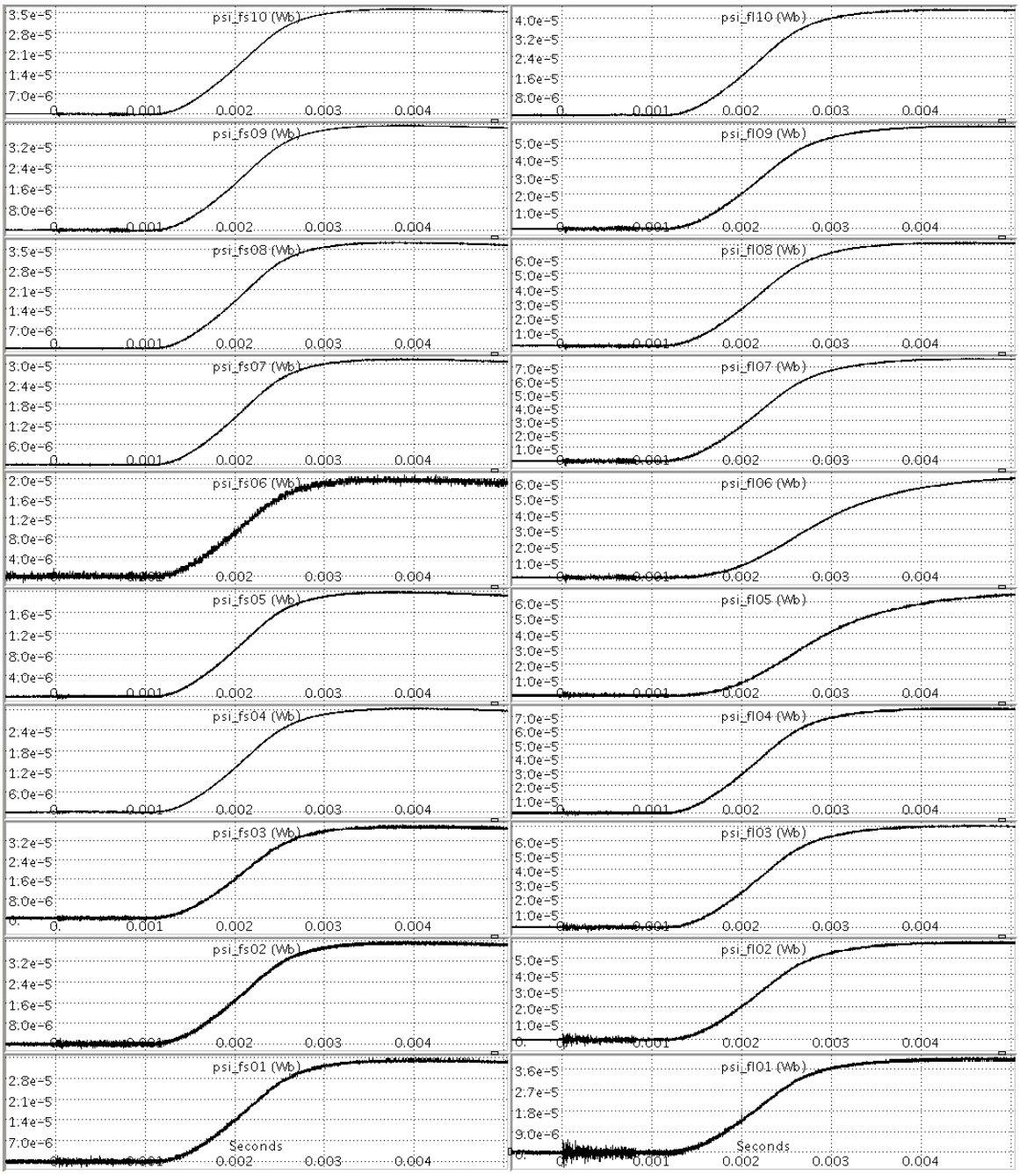

Figure B.14: Flux loop signals, shot 128580.

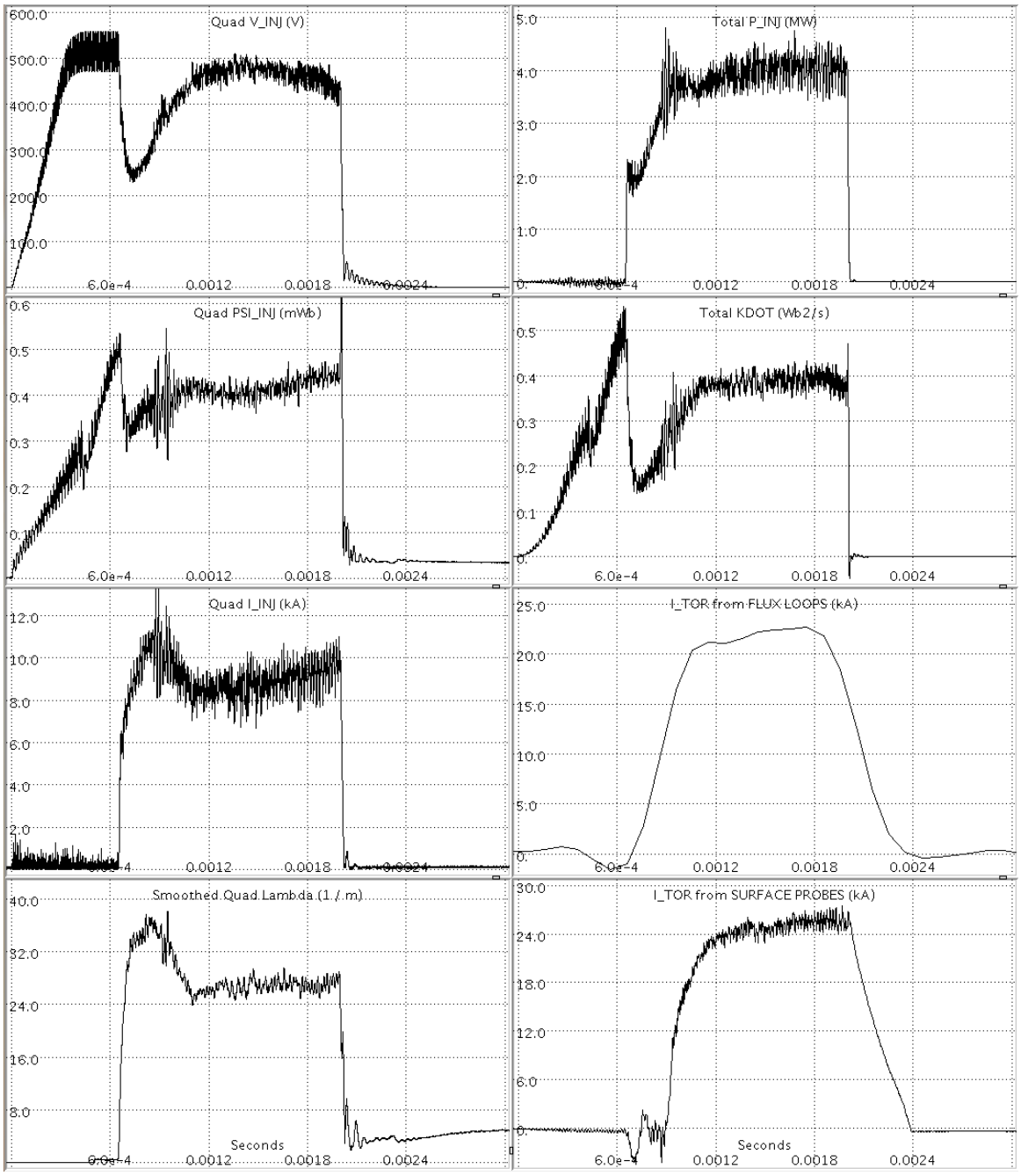

Figure B.15: Calculated parameters, shot 128580.

B.4 Shot 129213

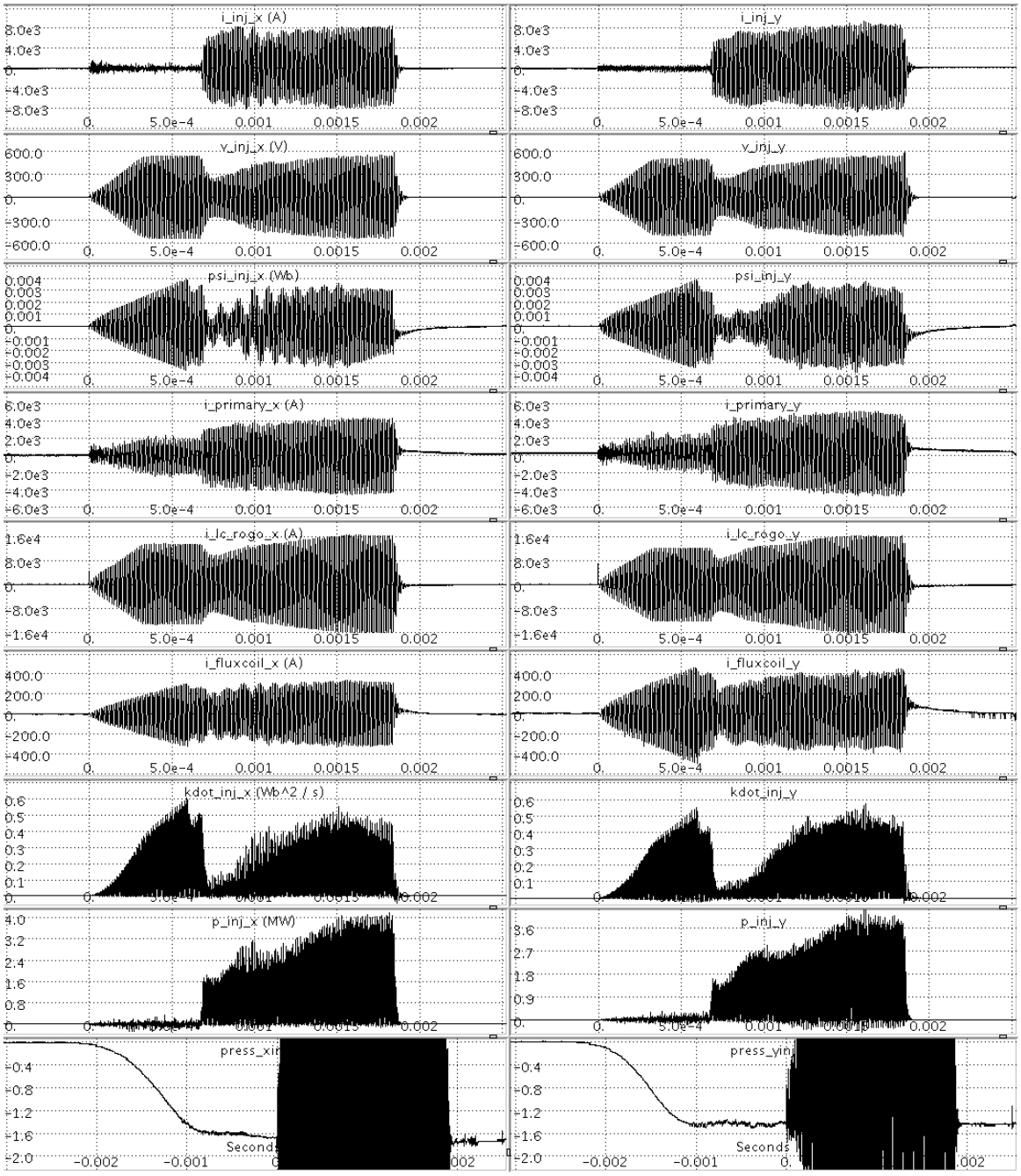

Figure B.16: Injector signals, shot 129213.

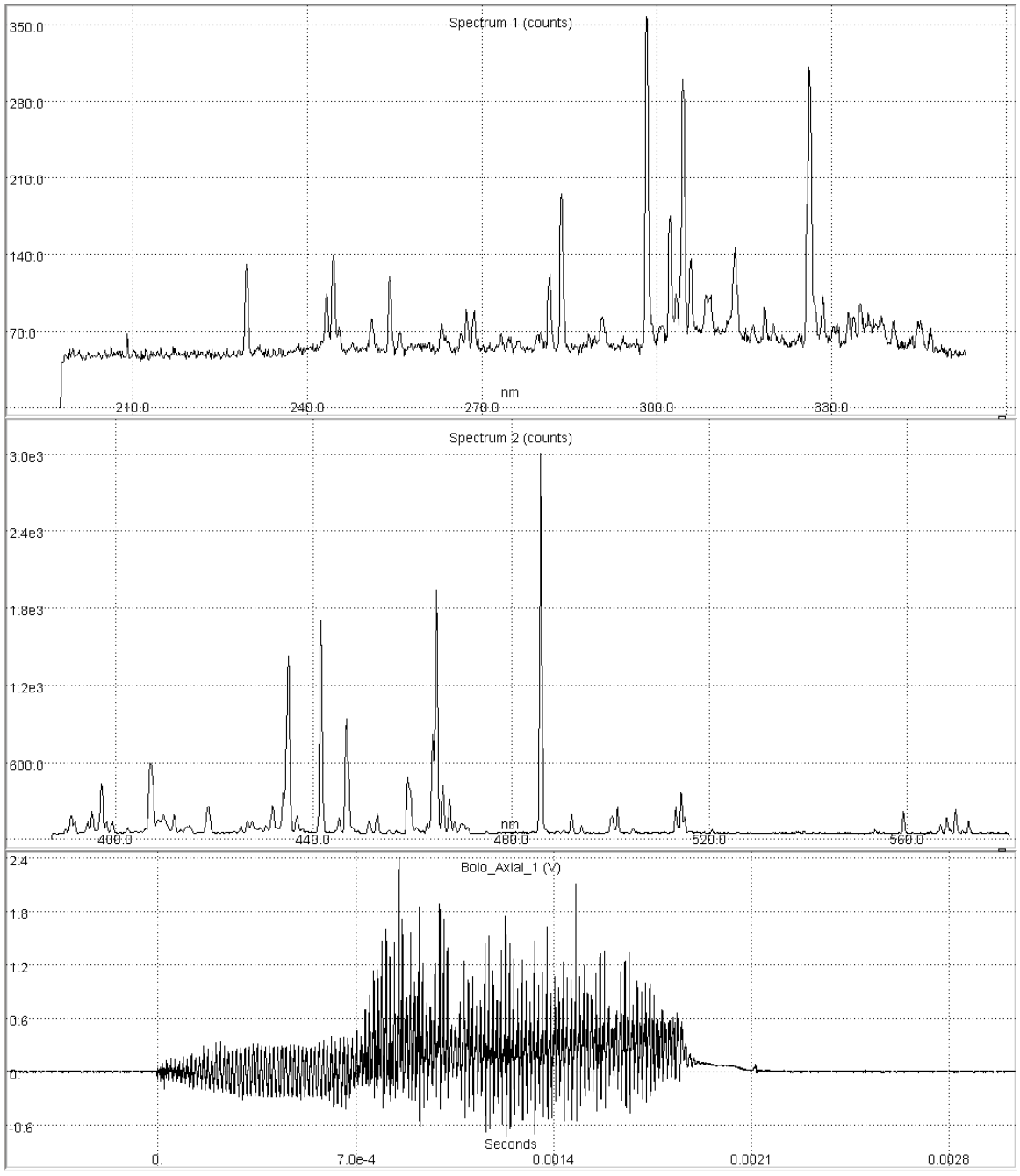

Figure B.17: SPRED spectrum and bolometer signal, shot 129213.

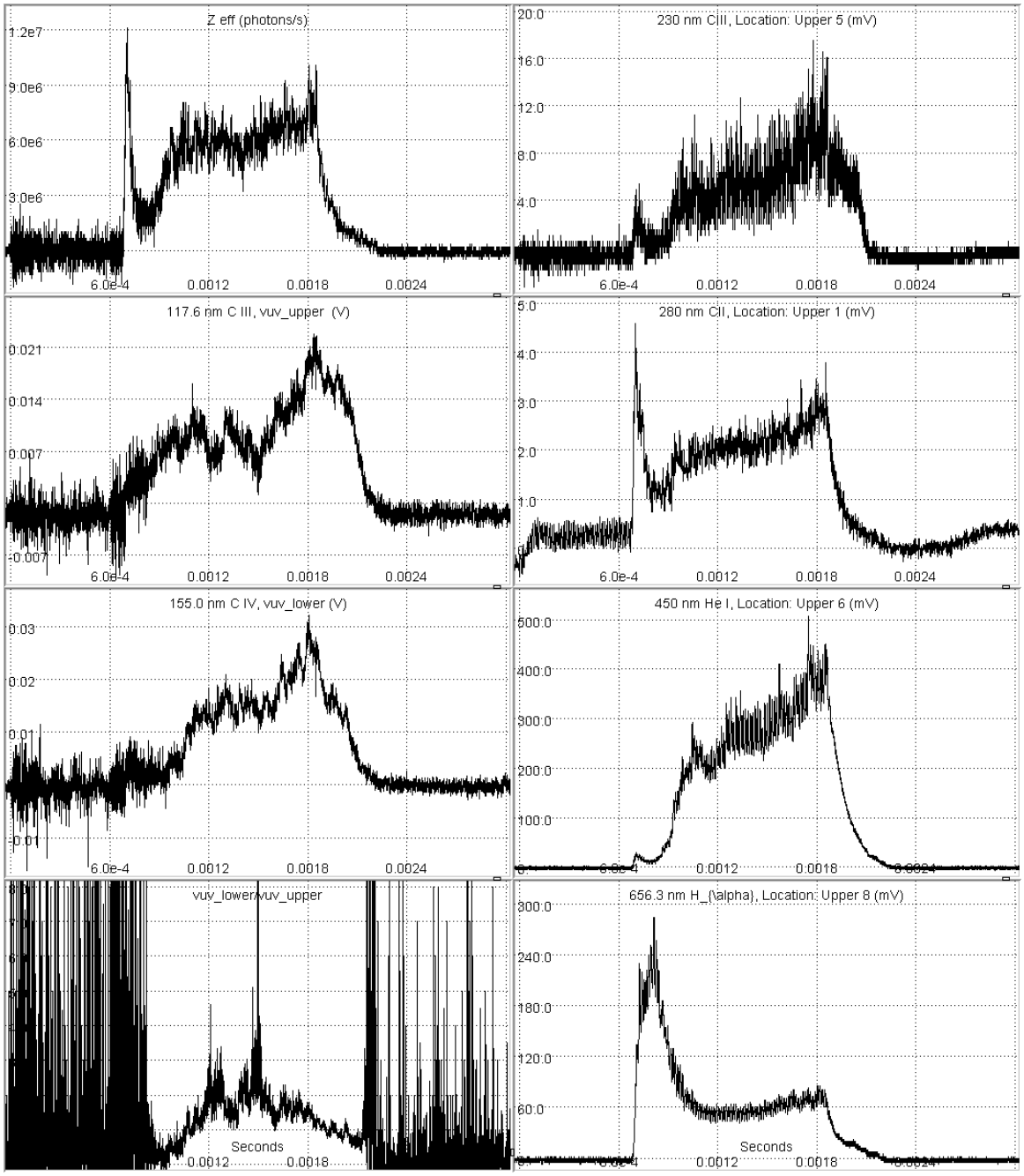

Figure B.18: Spectral signals, shot 129213.

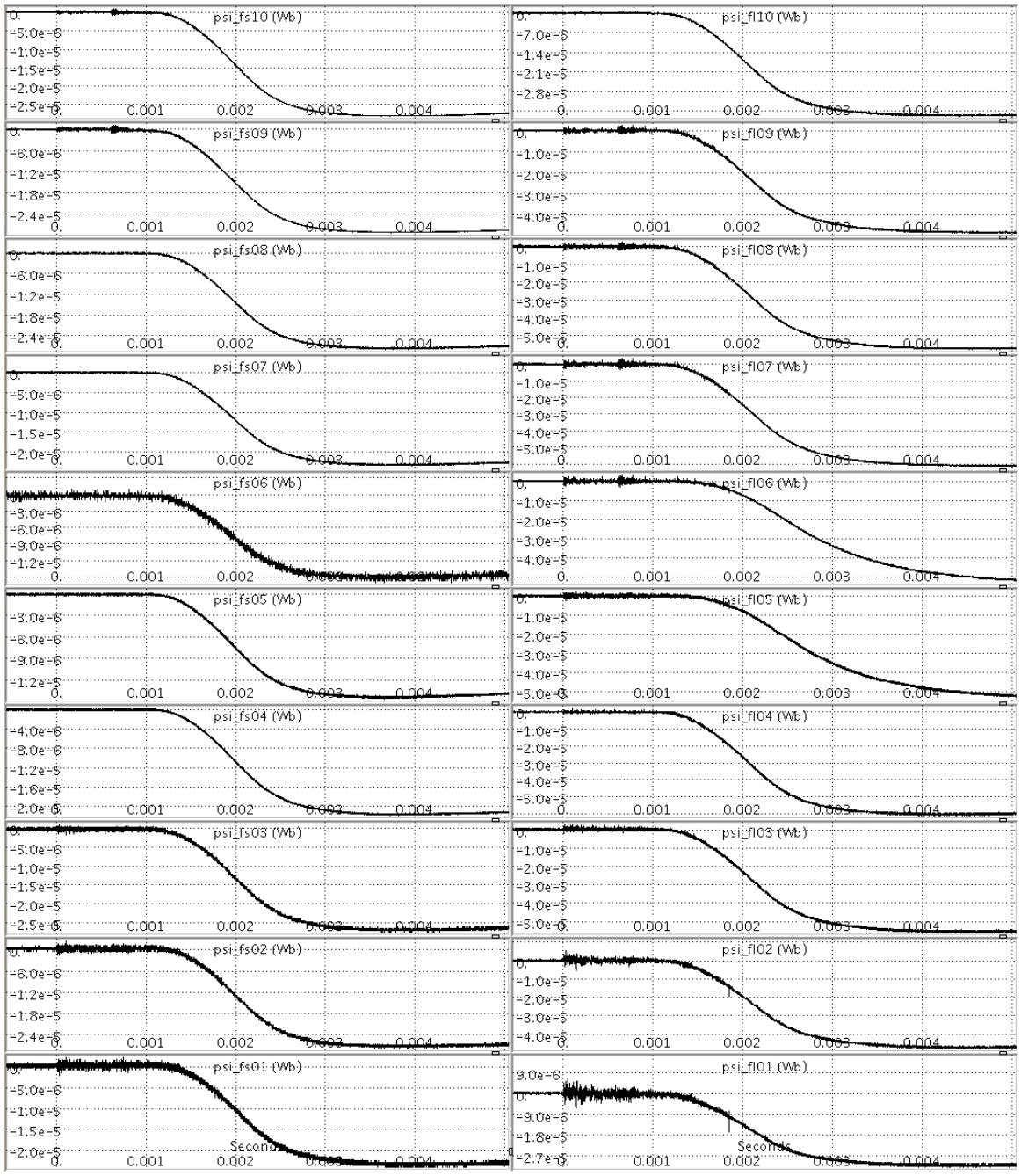

Figure B.19: Flux loop signals, shot 129213.

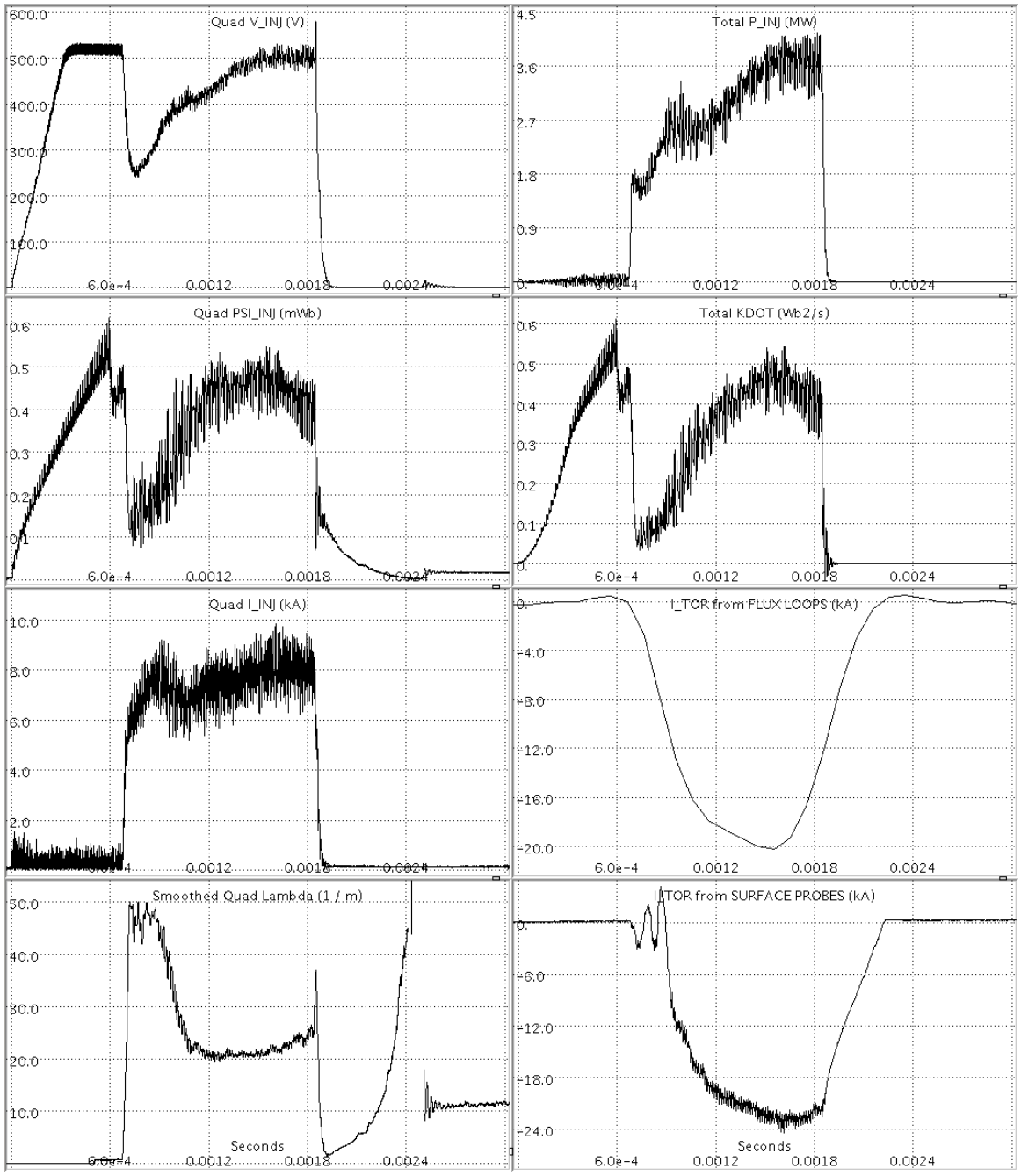

Figure B.20: Calculated parameters, shot 129213.

B.5 Shot 129810

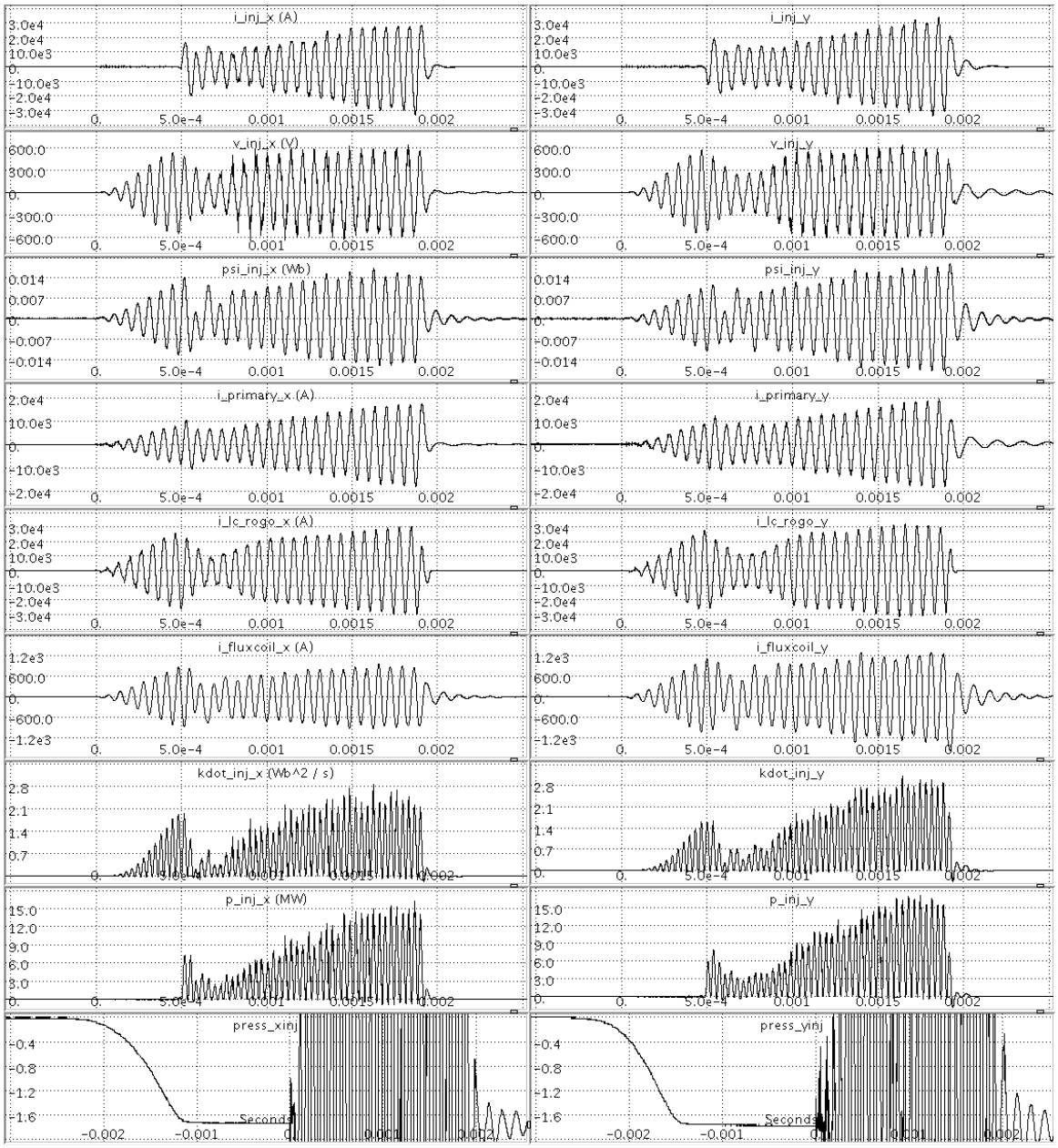

Figure B.21: Injector signals, shot 129810.

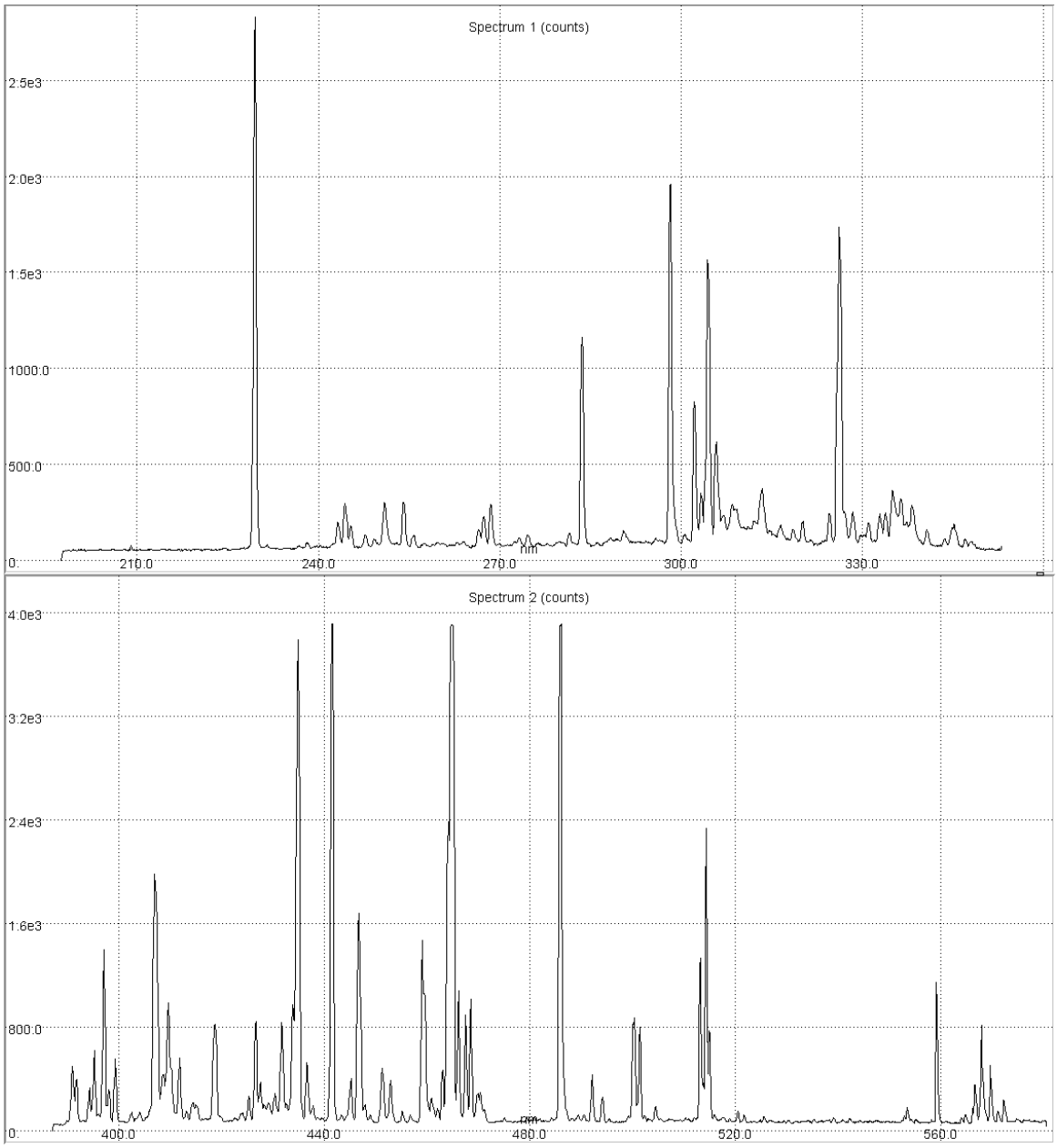

Figure B.22: SPRED spectrum, shot 129810. The bolometer was not operational.

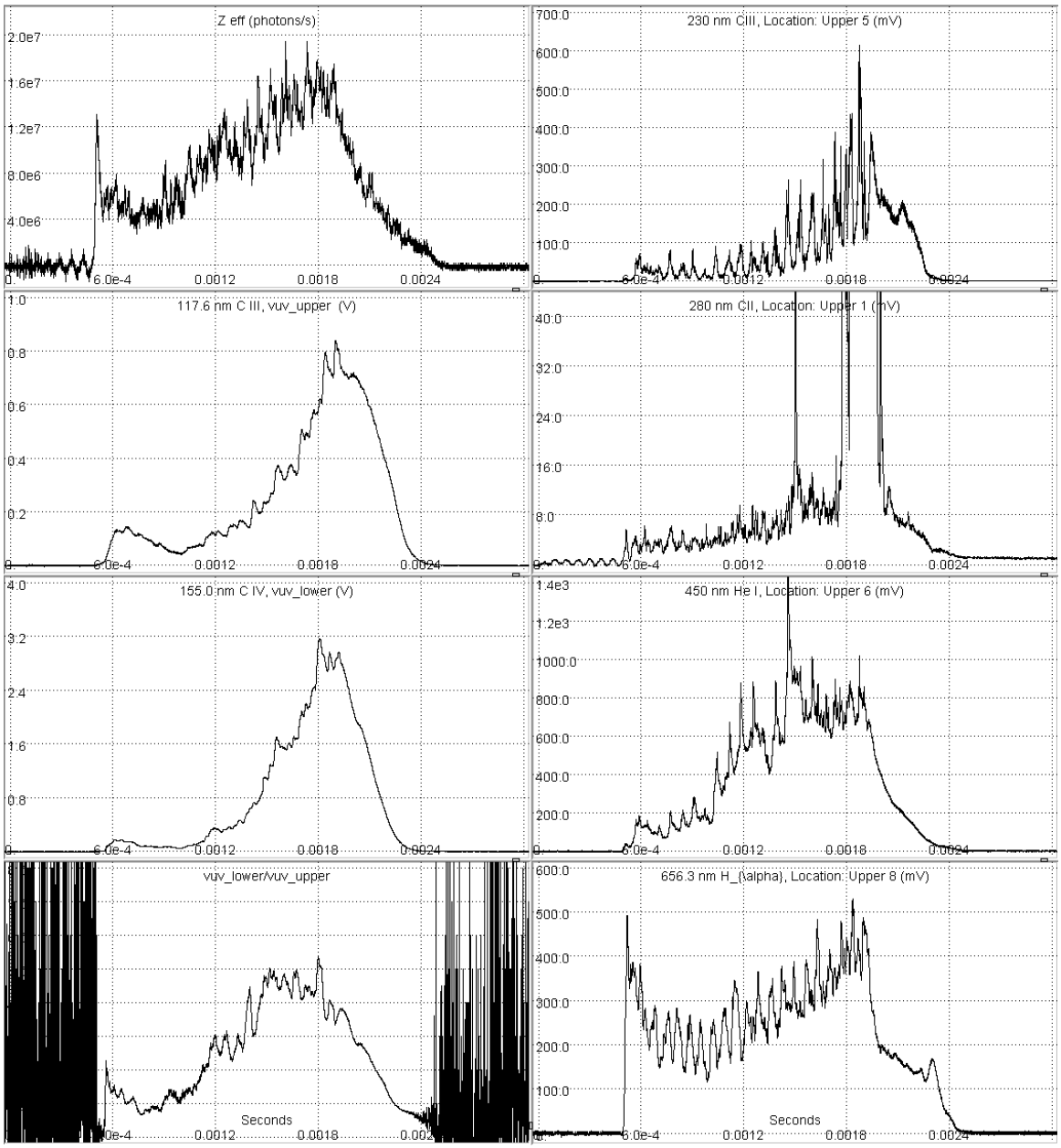

Figure B.23: Spectral signals, shot 129810.

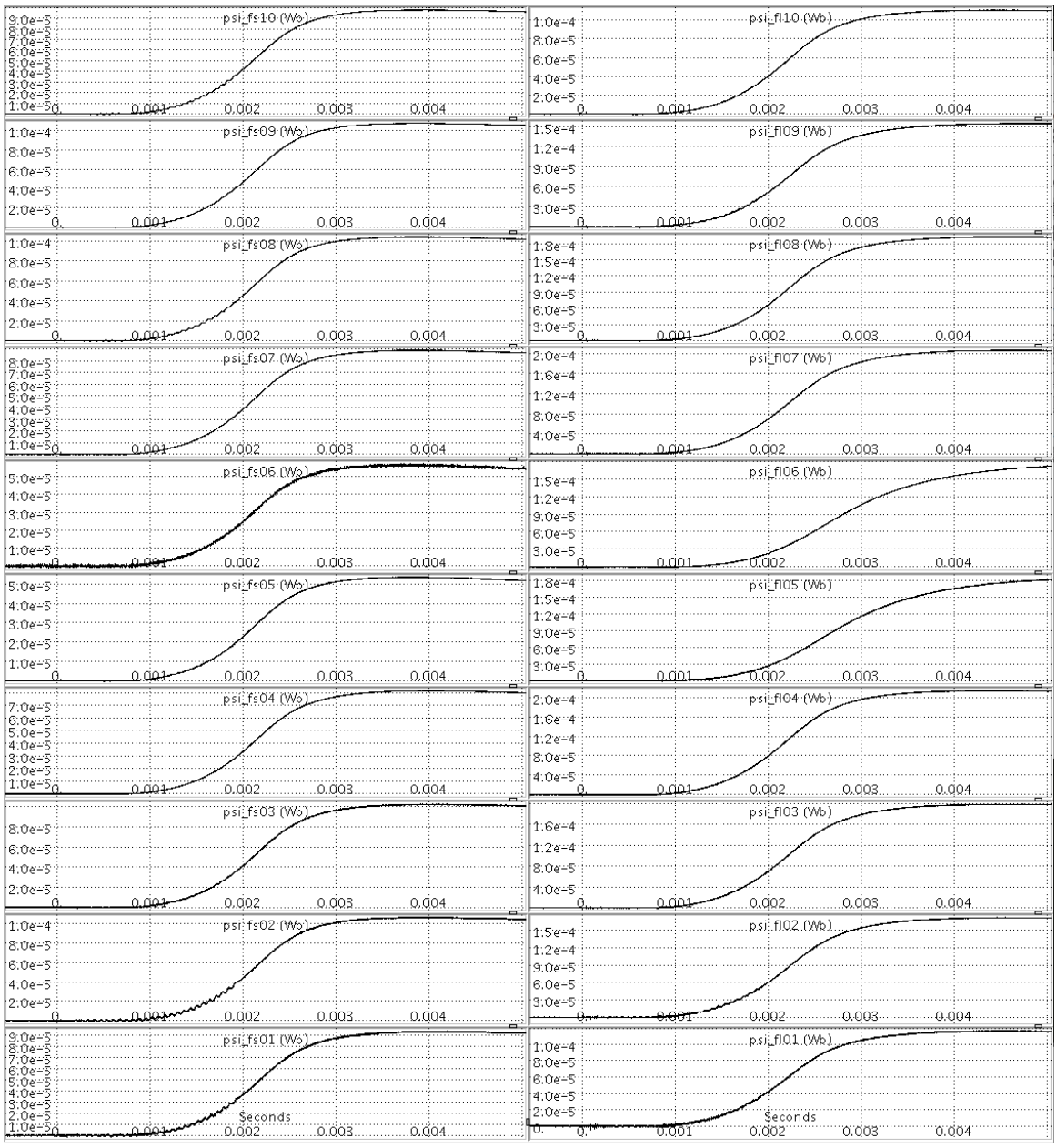

Figure B.24: Flux loop signals, shot 129810.

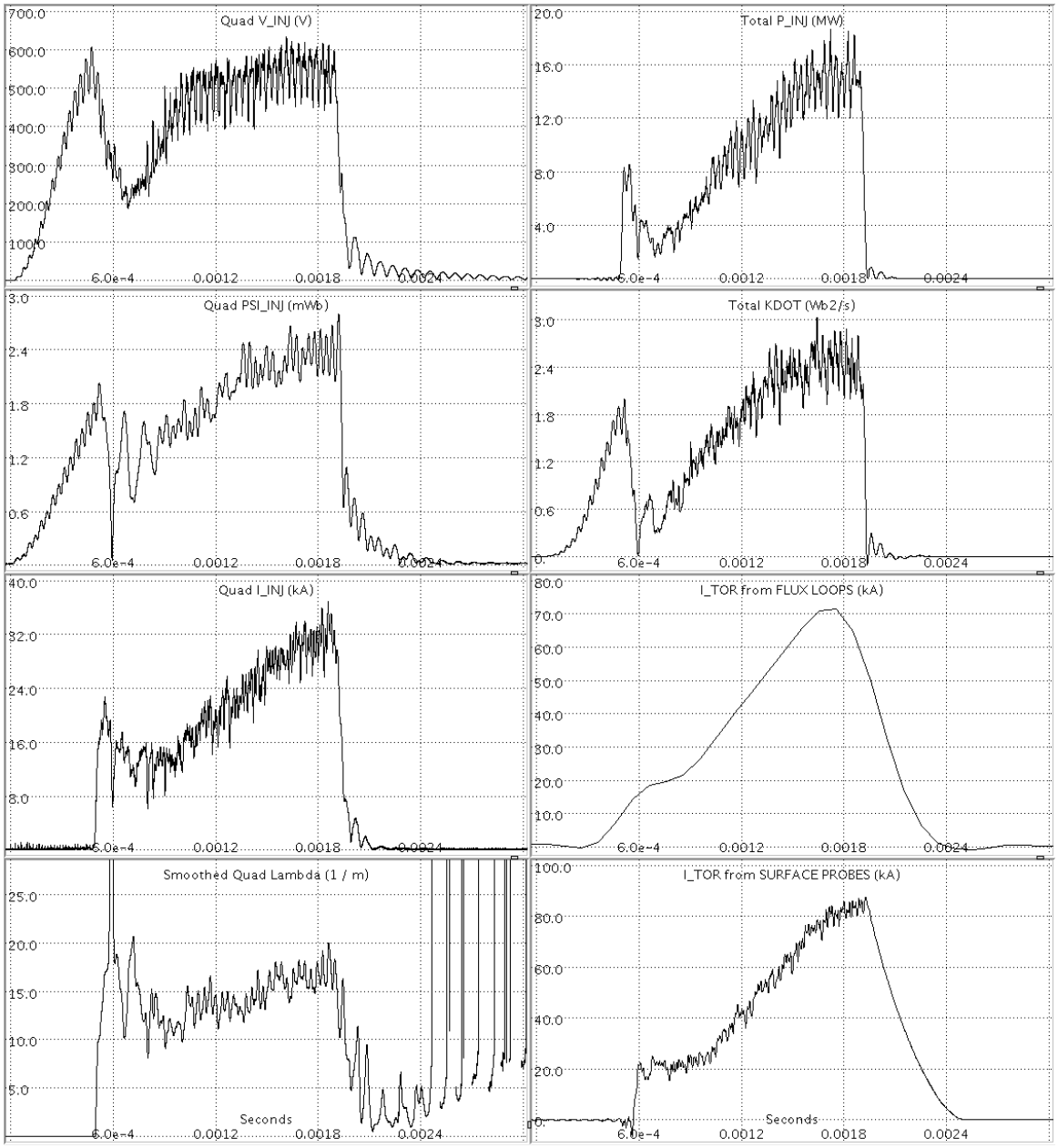

Figure B.25: Calculated parameters, shot 129810.

Appendix 3

MATLAB AND PYTHON CODE

C.1 Calibration

The calibration codes and functions can be found on the 27-inch iMac in the HIT-SI3 post-doc office in the directory:

`/Users/Aaron/IDS/Calibration/`

The following script, `calibrationMAIN_145kHz.m` has input settings optimized for 14.5 kHz, high-current shots.

```
% PHASE 1:
% Load in calibration movies for both upper and lower fibers, perform SVD,
% add first topos for both movies together, display.  User selects channel
% numbers and coordinates of first and last channel, and excludes dead
% channels.
% PHASE 2:
% Perform first fit for approximate coordinates of all channels.  Bin in
% WAVELENGTH direction and fit sum of all spatial direction Gaussians using
% the same fitting parameters for all.
% This yields approximate center coordinates
% for all channels and estimate of spatial FWHM.
% Perform accurate, 2D Gaussian fit.  Loop through each channel, select
% three pixel columns nearest peak in real space and appropriate bounds in
% wavelength space.  Create mesh grid and fit to Gaussian.  Output 'peaks'
% and 'bounds' with option to save to tree.  Also outputs CENTER and FWHM.
%---------- Doublet Option
% PHASE 3:
% Step one of doublet fitting - bin in real direction to visually find
% separation of peaks
% PHASE 4:
% Optional doublet calibration - moves the specified distance away from the
% main line to a dimmer line and repeats the 2D Gaussian fit to obtain an
% extimate of PIX_SP.
%--------- Motor Option
% PHASE 5:
% Motor Calibration for PIX_SP - based on slant of earlier calibration
% lines, calculates function for reestimating center at any wavelength
% position on CCD
% PHASE5B:
% The grating motor is very linear and consistent, but apparently
% SIGNIFICANTLY not close to its claimed speed.  This section corrects for
% this by loading in a plasma shot and fitting to two or more lines, then
```

```
% adjusting all the PIX_SP values.
%---------
% PHASE 6:
% Find FWHM and plot FWHM, PIX_SP, REL_INT
% PHASE 7:
% Discard marginal or bad channels
% PHASE 8:
% Save selected settings to tree.
clear all; close all; clc;
addpath('~/IDS/Matlab/');
addAllThePaths;
%% Settings
% SAVING FIGURES --------
% PHASE 1
approxFit.save = 0;
approxFit.file = '/home/aaron/IDS/Calibration/Images/test';
% PHASE 2
indivFit.save = 0;
indivFit.chan = 24;
indivFit.file = ['/home/aaron/IDS/Calibration/Images/example_cal_fit_' ...
    num2str(indivFit.chan)];
allFit.save = 0;
allFit.file = '/home/aaron/IDS/Calibration/Images/test';
% PHASE 3
dblBinFit.save = 0;
dblBinFit.file = [];
% PHASE 5B
calPlasmaFit.save = 0;
calPlasmaFit.file = '/home/aaron/IDS/Calibration/Images/calPlasma';
%%
% GENERAL ---------------------------------------------------------------
xWing = 2; % real space domain (1 => 3, 2 => 5)
% PHASE 1 ---------------------------------------------------------------
doPHASE1 = 1;
shot1 = 13053101; % always use this one
shot2 = 13053102; % set 'shot2' to zero if only using one fiber
xlim = [30 352]; % pixels for initial plotting, leave empty if unsure
ylim = [50 60]; % pixels for initial plotting, leave empty if unsure
% PHASE 2 ---------------------------------------------------------------
doPHASE2 = 1;
% chanNums = [1:4, 6:21, 23:72]; % 5 and 22 are dead
chanNums = [1:4, 6:72]; % 5 dead, 22 weak but including - 5/31/13
firstCenter = [37, 55]; % Center position of first channel, [real, wavelength]
lastCenter = [348, 55]; % Center position of last channel, [real, wavelength]
brightWing = 5;
% number of pixels in wavelength space for Gaussian fitting domain
force = []; % force finding channel(s) at specific x location(s)
remove = [];
% PHASE 3 ---------------------------------------------------------------
doPHASE3 = 0;
nPix2Bin = 20;
% number of pixels in real space (to the right of channel 1) to bin for display
% PHASE 4 ---------------------------------------------------------------
doPHASE4 = 0;
```

```
% PHASE 5 ------------------------------------------------------------------
% MUST MANUALLY CORRECT TIME BASE !
% BOTH MOVIES MUST HAVE SAME TIME INTERVAL !
doPHASE5 = 1;
shot3 = 13053103; % Motor Calibration Shot
shot4 = 13053104; % Optional second motor calibration shot
motorSpeed = 0.9867; % [nm per second]
% 4-16-15
% "motorSpeed" nominally 1, empirically set by analyzing shots
% 129499 and 129496 from 1.7 to 2.0 ms. Systematic velocity difference of
% 2.46 km/s between two C III lines observed. This means PIX_SP needs to
% decrease by 1.35%, thus the above number.
% PHASE 5 B ----------------------------------------------------------------
doPHASE5B = 0;
shotPlas = 129499; % Plasma shot to analyze
channel = 19; % Channel Number
timePt = 280; % time point
% pixelNums = [38 53 63]; % pixel numbers of peaks to within 1 pixel
% plasmaLams = 1e-9 * [464.913, 465.025, 465.147];
% corresponding line wavelengths
pixelNums = [38 53 63];
plasmaLams = 1e-9 * [464.742, 464.913, 465.025];
lineIDs = {'C III', 'O II', 'C III'};
calLam = 1e-9 * 465.025; % [m]
yWing = 7;
% similar to 'brightWing', but allowing different value for plasma lines
factor = 1; % estimated correction to motor speed
% PHASE 6 ------------------------------------------------------------------
doPHASE6 = 1;
% PHASE 7 ------------------------------------------------------------------
doPHASE7 = 1;
discard = [22, 53, 54];
% PHASE 8 ------------------------------------------------------------------
doPHASE8 = 1; % VERY IMPORTANT
% Uncomment desired shots and corresponding geometry file
%-------------------------------------------------------------------------
% saveToShots = [129510:129534, 129544:129571];
% mohawk port, orthogonal to midplane, and axial port. 'impacts4'
% impactsFile = '/home/aaron/IDS/Geometry/impacts4.mat';
%-------------------------------------------------------------------------
% saveToShots = [129438:129453, 129460:129480, 129485:129500, 129577:129600, ...
%     129611:129627, 129642:129661, 129662:129685, 129688:129711, ...
%     129719:129746, 129747:129759, 129800:129802];
% mohawk port, chords in midplane, axial port, 'impacts2'
% impactsFile = '/home/aaron/IDS/Geometry/impacts2.mat';
%-------------------------------------------------------------------------
% saveToShots = [129787:129799]; % axial ports facing each other
% impactsFile = '/home/aaron/IDS/Geometry/impacts3.mat';
%-------------------------------------------------------------------------
% saveToShots = [129807:129824];
% reentrant port at 71 degrees and axial port, 'impacts1'
% impactsFile = '/home/aaron/IDS/Geometry/impacts1.mat';
%-------------------------------------------------------------------------
stt.CAL_LAMBDA = 1;
```

```
stt.LAMBDA = 1;
stt.VOLTAGE = 0;
stt.MASS = 1;
stt.PEAKS = 0;
stt.REL_INT = 0;
stt.PIX_SP = 1;
stt.IMPACTS = 0;
hitsi3 = 0;
CAL_LAMBDA = 1e-9 * 464.913;
% [m], should be effective wavelength that matches PEAKS centers
LAMBDA = 1e-9 * [465.025, 464.913, 464.742];
% [m], plasma lines in descending wavelength
VOLTAGE = 0; % image intensifier voltage, '0' if not using
MASS = [12, 16, 12]; % [AMU], corresponding to LAMBDA
%% PHASE 1 - Load Data, SVD, Display
if doPHASE1
    [data, X, Y, n_pix, n_chan] = calSVD(shot1, shot2);
    %% Plot First Topo
    S = get(0,'ScreenSize');
    fntsz = 24;
    h1 = figure('Visible','on','Name','Calibration Data','Position',...
        [S(3)/12, S(4)/8, 5*S(3)/6 3*S(4)/4], 'Color', [1 1 1]);
    ax(1) = axes('Parent', h1, 'Position', [.1 .11 .8 .33], 'FontSize', fntsz);
    h3 = surf(X, Y, data);
    hold on;
    shading interp;
    colormap jet;
    colorbar;
    grid on;
    view([0 90]);
    set(gca, 'FontSize', fntsz);
    if length(xlim) == 2
        set(gca, 'XLim', xlim);
    else
        set(gca, 'XLim', [1 n_chan]);
    end
    if length(ylim) == 2
        set(gca, 'YLim', ylim);
    else
        set(gca, 'YLim', [1 n_pix]);
    end
    title('BD Filtered Calibration Data');
    xlabel('Pixel Number (real space)');
    ylabel({'Pixel Number'; '(wavelength space)'});
end
%% PHASE 2 - Approximate fit to find all channel centers
if doPHASE2
    % Find approximate center positions of all channels
    peaks = fitAllChans(data, chanNums, firstCenter, lastCenter,...
         brightWing, approxFit, force, remove);
    % Fit each peak to 2D Gaussian individually
    [PEAKS, REL_INT, par, fits] = calGauss2D(peaks, data,...
         brightWing, xWing, indivFit);
    % Plot Fits ----------------------------------
```

```
    figure(h1);
    ax(2) = axes('Parent', h1, 'Position', [.1 .57 .8 .33], 'FontSize', fntsz);
    for n = 1:size(peaks, 1) % loop over channels
        h3 = surf(fits.Xf(:,:,n), fits.Yf(:,:,n), fits.Zf(:,:,n));
        hold on;
    end
    shading interp;
    colormap jet;
    colorbar;
    view([0 90]);
    set(gca, 'FontSize', fntsz);
    set(gca, 'XTick', []);
    linkaxes(ax, 'xy');
    title('Reconstructed Fits');
    ylabel({'Pixel Number'; '(wavelength space)'});
    pause(0.1);
    gcf;
    % SAVE
    if allFit.save
        fig_save = getframe(h1);
        [Xfig, mapfig] = frame2im(fig_save);
        imwrite(Xfig, [allFit.file '.png']);
    end
end
%% PHASE 3 - Doublet Calibration -
% bin in real space to estimate offset of second peak
if doPHASE3
    % INCOMPLETE
end
%% PHASE 4 - Doublet Calibration - Calculate PIX_SP from doublet fitting
if doPHASE4
    % INCOMPLETE
end
%% PHASE 5 - Motor Calibration for PIX_SP
if doPHASE5
    PIX_SP = fitMotor(shot3, shot4, PEAKS, motorSpeed, brightWing, xWing);
end
%% PHASE 5 B - Correct Motor Calibration with Plasma Lines
if doPHASE5B
    PIX_SP = calPlasma(shotPlas, PIX_SP, PEAKS, channel, timePt, ...
        pixelNums, plasmaLams, calLam, xWing, yWing, factor,...
         calPlasmaFit, lineIDs);
end
%% PHASE 6 - Convert Fitting Parameters, plot FWHM, PIX_SP, and REL_INT
if doPHASE6
    FWHM = 2.35482 * PIX_SP .* PEAKS(:, 5)'; % convert sigma to FWHM
    h1 = figure('Visible','on','Name','Approximate Fit','Position',...
        [S(3)/12, S(4)/6, 5*S(3)/6 2*S(4)/3], 'Color', [1 1 1]);
    % REL_INT
    h2 = axes('Parent', h1, 'Position', [.1 .1 .8 .25], 'FontSize', fntsz);
    h3 = plot(PEAKS(:, 1), REL_INT, 'ob');
    hold on;
    grid on;
    set(gca, 'XLim', [PEAKS(1, 1) PEAKS(end, 1)]);
```

```
    xlabel('Channel Number');
    ylabel('REL INT');
    % FWHM
    h2 = axes('Parent', h1, 'Position', [.1 .4 .8 .25], 'FontSize', fntsz);
    h3 = plot(PEAKS(:, 1), FWHM, 'ob');
    hold on;
    grid on;
    set(gca, 'XLim', [PEAKS(1, 1) PEAKS(end, 1)]);
    set(gca, 'XTickLab', []);
    ylabel('FWHM [m]');
    % PIX_SP
    h2 = axes('Parent', h1, 'Position', [.1 .7 .8 .25], 'FontSize', fntsz);
    h3 = plot(PEAKS(:, 1), PIX_SP, 'ob');
    hold on;
    grid on;
    set(gca, 'XLim', [PEAKS(1, 1) PEAKS(end, 1)]);
    set(gca, 'XTickLab', []);
    ylabel('PIX SP [m]');
end
%% PHASE 7 - Discard Bad Channels
if and(doPHASE7, length(discard >= 1))
    bad = [];
    for n = 1:length(discard)
        b = find(PEAKS(:, 1) == discard(n)); % index of nth bad channel
        bad = [bad b]; % all indices of bad channels
    end
    c = PEAKS(:, 1); % alternate array of channel numbers
    c(bad) = zeros(1, length(discard));
    % zero out bad channels so I can use 'find'
    ind = find(c); % finds all non-zero elements of c
    % Use 'ind' to trim arrays
    PEAKS = PEAKS(ind, :);
    REL_INT = REL_INT(ind);
    PIX_SP = PIX_SP(ind);
end
%% Retrieve 'impacts', print all data before saving
load(impactsFile);
IMPACTS = impacts(PEAKS(:, 1)); % select only channel numbers being used
try
    REL_INT;
catch % not running the whole code
    REL_INT = NaN;
    PEAKS = NaN;
    PIX_SP = NaN;
    stt.REL_INT = 0; % make damn sure I don't write NaNs to the tree
    stt.PEAKS = 0;
    stt.PIX_SP = 0;
end
%% PHASE 8 - Save Settings to Tree
if doPHASE8
    saveToTree3(saveToShots, stt, CAL_LAMBDA, LAMBDA, VOLTAGE, MASS, PEAKS, ...
        REL_INT, PIX_SP, IMPACTS, hitsi3);
end
```

The following script, `calibrationMAIN_535_685kHz.m` is optimized for both 53.5 kHz and 68.5 kHz data. Comments have been removed which are included in the above code for brevity.

```
%% UPDATED AARON HOSSACK, JAN 2015
% Making sure same code works on my linux machine. This code was updated
% with the best available calibration data which applies to all 53.5 and
% 68.5 kHz data. Original update was done for B.S. Victor's PoP paper.
%% UPDATED BY RIAN CHANDRA, MAY 2014
    % Code now inclues doublet calibration, can interface with new cine
    % file formats, and is updated to new MDSplus. SPR 2014.
clear all; close all; clc;
addpath('~/IDS/Matlab/');
try
    addAllThePaths;
end
%% Settings
% SAVING FIGURES --------
% PHASE 1
approxFit.save = 0;
approxFit.file = 'T:\RChandra\A-A-Ron Code\test';
% PHASE 2
indivFit.save = 0;
indivFit.chan = 24;
indivFit.file = ['T:\RChandra\A-A-Ron Code\example_cal_fit_' ...
    num2str(indivFit.chan)];
allFit.save = 0;
allFit.file = 'T:\RChandra\A-A-Ron Code\\test';
% PHASE 3
dblBinFit.save = 0;
dblBinFit.file = [];
%%
% GENERAL ---------------------------------------------------------------
xWing = 1; % real space domain (1 => 3, 2 => 5)
% PHASE 1 ---------------------------------------------------------------
doPHASE1 = 1;
shot1 = 13041901; % always use this one
%shot2 = 0; % set 'shot2' to zero if only using one fiber
% shot1 = 13041901; % always use this one
shot2 = 13041903; % set 'shot2' to zero if only using one fiber
xlim = [0 256]; % pixels for initial plotting, leave empty if unsure
ylim = [48 68]; % pixels for initial plotting, leave empty if unsure
% PHASE 2 ---------------------------------------------------------------
doPHASE2 = 1;
chanNums = [1:4, 6:22, 24:36, 37:72];%, 72:86]; %
firstCenter = [12, 58]; % Center position of first channel, [real, wavelength]
lastCenter = [254, 58]; % Center position of last channel, [real, wavelength]
brightWing = 4;
% number of pixels in wavelength space for Gaussian fitting domain
force = [12, 77, 83, 193, 229];
% force finding channel(s) at specific x location(s)
remove = [84]; % pixel number in channel space to ignore a local max
% PHASE 3 ---------------------------------------------------------------
```

```
doPHASE3 = 1;
DoubletShot1 = 13030401;
DoubletShot2 = 13030402;
nPix2Bin = 20;
% number of pixels in real space (to the right of channel 1) to bin for display
% PHASE 4 ----------------------------------------------------------------
doPHASE4 = 1;
largerLam = 463.723e-9;
% smallerLam = 460.957e-9;
smallerLam = 460.463e-9;
% This is a kludgy correction to get PIX-SP to come out to the correct value
% - ACH, 2-3-15
% PHASE 5 ----------------------------------------------------------------
% MUST MANUALLY CORRECT TIME BASE !
% BOTH MOVIES MUST HAVE SAME TIME INTERVAL !
doPHASE5 = 0;
shot3 = 14062504; % Motor Calibration Shot
shot4 = 14062505; % Optional second motor calibration shot
motorSpeed = 1; % [nm per second]
% PHASE 5 B --------------------------------------------------------------
doPHASE5B = 0;
shotPlas = 140626021; % Plasma shot to analyze
channel = 21; % Channel Number
timePt = 50; % time point
pixelNums = [39 55 65]; % pixel numbers of peaks to within 1 pixel
plasmaLams = 1e-9 * [465.025, 464.913, 464.742]; % corresponding line wavelengths
calLam = 1e-9 * 465.025; % [m]
yWing = 7;
% similar to 'brightWing', but allowing different value for plasma lines
factor = 0.7; % estimated correction to motor speed
% PHASE 6 ----------------------------------------------------------------
doPHASE6 = 1;
% PHASE 7 ----------------------------------------------------------------
doPHASE7 = 1;
discard = [22, 53]; % added 22 on 4-27-15, ACH
% PHASE 8 ----------------------------------------------------------------
doPHASE8 = 1; % VERY IMPORTANT
hitsi3 = 0;
%% NB !!!! Different CAL_LAMBDA for 53.5 and 68.5 kHz !!!
%------------------------------
% saveToShots =[129225:129226];
% adding a couple more 68.5 kHz shots, 4-22-15, ACH
%------------------------------
saveToShots = [128457:128597, 129213:129215, 129225:129226];
                    % 53.5 kHz Toridal fiber, 71 degree re-enterant port,
                    % polidal fiber on 45 degree axial port, x-side
                    % ALSO valid for 68.5 kHz
impactsFile = '/home/aaron/IDS/Geometry/impacts1.mat';
%------------------------------
stt.CAL_LAMBDA = 1;
stt.LAMBDA = 1;
stt.VOLTAGE = 1;
stt.MASS = 1;
stt.PEAKS = 1;
```

```
stt.REL_INT = 1;
stt.PIX_SP = 1;
stt.IMPACTS = 1;
CAL_LAMBDA = 1e-9 * 464.913;
% [m], should be effective wavelength that matches PEAKS centers 68.5 kHz !!!
% CAL_LAMBDA = 1e-9 * 465.1096; % [m] % 53.5 kHz !!!
LAMBDA = 1e-9 * [465.025, 464.913, 464.742];
% [m], plasma lines in descending wavelength
VOLTAGE = 900; % image intensifier voltage, '0' if not using
MASS = [12, 16, 12]; % [AMU], corresponding to LAMBDA
%% PHASE 1 - Load Data, SVD, Display
if doPHASE1
    [data, X, Y, n_pix, n_chan] = calSVD(shot1, shot2);
    %% Plot First Topo
    S = get(0,'ScreenSize');
    fntsz = 24;
    h1 = figure('Visible','on','Name','Calibration Data','Position',...
        [S(3)/12, S(4)/8, 5*S(3)/6 3*S(4)/4], 'Color', [1 1 1]);
    ax(1) = axes('Parent', h1, 'Position', [.1 .11 .8 .33], 'FontSize', fntsz);
    h3 = surf(X, Y, data);
    hold on;
    shading interp;
    colormap jet;
    colorbar;
    grid on;
    view([0 90]);
    set(gca, 'FontSize', fntsz);
    if length(xlim) == 2
        set(gca, 'XLim', xlim);
    else
        set(gca, 'XLim', [1 n_chan]);
    end
    if length(ylim) == 2
        set(gca, 'YLim', ylim);
    else
        set(gca, 'YLim', [1 n_pix]);
    end
    title('BD Filtered Calibration Data');
    xlabel('Pixel Number (real space)');
    ylabel({'Pixel Number'; '(wavelength space)'});
end
%% PHASE 2 - Approximate fit to find all channel centers
if doPHASE2
    % Find approximate center positions of all channels
    peaks = fitAllChans(data, chanNums, firstCenter, lastCenter,...
         brightWing, approxFit, force, remove);
    % Fit each peak to 2D Gaussian individually
    [PEAKS, REL_INT, par, fits] = calGauss2D(peaks, data, brightWing,...
         xWing, indivFit);
    % Plot Fits ----------------------------------
    figure(h1);
    ax(2) = axes('Parent', h1, 'Position', [.1 .57 .8 .33], 'FontSize', fntsz);
    for n = 1:size(peaks, 1) % loop over channels
        h3 = surf(fits.Xf(:,:,n), fits.Yf(:,:,n), fits.Zf(:,:,n));
```

```
        hold on;
    end
    shading interp;
    colormap jet;
    colorbar;
    view([0 90]);
    set(gca, 'FontSize', fntsz);
    set(gca, 'XTick', []);
    linkaxes(ax, 'xy');
    title('Reconstructed Fits');
    ylabel({'Pixel Number'; '(wavelength space)'});
    pause(0.1);
    gcf;
    % SAVE
    if allFit.save
        fig_save = getframe(h1);
        [Xfig, mapfig] = frame2im(fig_save);
        imwrite(Xfig, [allFit.file '.png']);
    end
end
%% PHASE 3 - Doublet Calibration -
% bin in real space to estimate offset of second peak
if doPHASE3
    [data, X, Y, n_pix, n_chan] = calSVD(DoubletShot1, DoubletShot2);
    ypeaks = fitAllChansWavelength(data);
    % find y bounds for plotting
    ylim2 = [];
    xlim2 = [];
    ylim2 = [(ypeaks(1, 2) - 15) (ypeaks(2, 2) + 15)];
    S = get(0, 'ScreenSize');
    fntsz = 24;
    h1 = figure('Visible', 'on', 'Name', 'Calibration Data2', 'Position',...
        [S(3)/12, S(4)/8, 5*S(3)/6 3*S(4)/4], 'Color', [1 1 1]);
    %ax(1) = axes('Parent', h1, 'Position', [.1 .11 .8 .33], 'FontSize', fntsz);
    ax(1) = axes('Parent', h1, 'FontSize', fntsz);
    h3 = surf(X, Y, data);
    h3 = surf(X, Y, data);
    hold on;
    plot3(0:250, ypeaks(1, 2)*ones(1, 251), .025*ones(1, 251),...
         'linewidth', 3, 'Color', 'black');
    plot3(0:250, ypeaks(2, 2)*ones(1, 251), .025*ones(1, 251),...
         'linewidth', 3, 'Color', 'black');
    shading interp;
    colormap jet;
    colorbar;
    grid on;
    view([0 90]);
    set(gca, 'FontSize', fntsz);
    if length(xlim2) == 2
        set(gca, 'XLim', xlim2);
    else
        set(gca, 'XLim', [1 260]);
    end
    if length(ylim2) == 2
```

```
        set(gca, 'YLim', ylim2);
    else
        set(gca, 'YLim', [1 720]);
    end
    title('BD Filtered Doublet Calibration Data');
    xlabel('Pixel Number (real space)');
    ylabel({'Pixel Number'; '(wavelength space)'});
end
%% PHASE 4 - Doublet Calibration - Calculate PIX_SP from doublet fitting
if doPHASE4
    % INCOMPLETE
    %Test doing rough channels fit for x guess values.
    [m, n] = size(data);
    %take ypeaks, use that as the y-guess for 2dGauss, use peaks as xguess
    peaks(:, 1) = chanNums;
    peaks(:, 2) = par(:,2) + 1;%take the 2dgauss x-values from phase 2
    peaks(:, 3) = ypeaks(1, 2);%do bottom line first (ypeaks(1) is lower index)
    [~, ~, par1, fits1] = calGauss2D(peaks, data, brightWing, xWing, indivFit);
    peaks(:, 3) = ypeaks(2,2);
    %peaks(:,2)=par1(:,2); Tested: didnt improve accuracy much.
    [~, ~, par2, fits2] = calGauss2D(peaks, data, brightWing, xWing, indivFit);
    WavelengthOffset = par2(:, 3) - par1(:, 3);
    %Gauss fit using singlet gauss x values for x guess
    scatter3(par2(:, 2), par2(:, 3), 0.03 * ones(1, size(par2(:, 3), 1)),...
        'r', 'Marker', '*');
    scatter3(par1(:, 2), par1(:, 3), 0.03 * ones(1, size(par1(:, 3), 1)),...
        'r', 'Marker', '*');
    %Guess value, singlet gauss x and rough y
    scatter3(par(:, 2) + 1, ypeaks(1, 2) * ones(1, size(par1(:, 3), 1)),...
        0.03 * ones(1, size(par1(:, 3), 1)), 'g', 'Marker', '*');
    scatter3(par(:, 2) + 1, ypeaks(2, 2) * ones(1, size(par1(:, 3), 1)),...
        0.03 * ones(1, size(par1(:, 3), 1)), 'g', 'Marker', '*');
    iterator = 1; %create two vectors of points, to plot line segments
    Xvector = 0;
    Yvector = 0;
    for points=1:size(peaks(:, 2), 1)
        Xvector(iterator) = par1(points, 2);
        Xvector(iterator + 1) = par2(points, 2);
        Xvector(iterator + 3) = 0;
        Yvector(iterator) = par1(points, 3);
        Yvector(iterator + 1) = par2(points, 3);
        Yvector(iterator + 3) = 0;
        iterator = iterator + 3;
    end
    Xvector(3:3:(size(Xvector, 2))) = NaN;
    Yvector(3:3:(size(Xvector, 2))) = NaN;
    plot3(Xvector, Yvector, 0.03*ones(1, size(Yvector, 2)), 'color', 'red');
    %%% Divide wavelength by pixel offset
    DifferenceInWavelength = ((largerLam) - (smallerLam)) .*...
        ones(size(WavelengthOffset));
    PIX_SP = DifferenceInWavelength ./ WavelengthOffset;
end
%% PHASE 5 - Motor Calibration for PIX_SP
if doPHASE5
```

```
    PIX_SP = fitMotor(shot3, shot4, PEAKS, motorSpeed, brightWing, xWing);
end
%% PHASE 5 B - Correct Motor Calibration with Plasma Lines
if doPHASE5B
    PIX_SP = calPlasma(shotPlas, PIX_SP, PEAKS, channel, timePt, ...
        pixelNums, plasmaLams, calLam, xWing, yWing, factor);
end
%% PHASE 6 - Convert Fitting Parameters, plot FWHM, PIX_SP, and REL_INT
if doPHASE6
    FWHM = 2.35482 * PIX_SP .* PEAKS(:, 5); % convert sigma to FWHM
    h1 = figure('Visible','on','Name','Approximate Fit','Position',...
        [S(3)/12, S(4)/6, 5*S(3)/6 2*S(4)/3], 'Color', [1 1 1]);
    % REL_INT
    h2 = axes('Parent', h1, 'Position', [.1 .1 .8 .25], 'FontSize', fntsz);
    h3 = plot(PEAKS(:, 1), REL_INT, 'ob');
    hold on;
    grid on;
    set(gca, 'XLim', [PEAKS(1, 1) PEAKS(end, 1)]);
    xlabel('Channel Number');
    ylabel('REL INT');
    % FWHM
    h2 = axes('Parent', h1, 'Position', [.1 .4 .8 .25], 'FontSize', fntsz);
    h3 = plot(PEAKS(:, 1), FWHM, 'ob');
    hold on;
    grid on;
    set(gca, 'XLim', [PEAKS(1, 1) PEAKS(end, 1)]);
    set(gca, 'XTickLab', []);
    ylabel('FWHM [m]');
    % PIX_SP
    h2 = axes('Parent', h1, 'Position', [.1 .7 .8 .25], 'FontSize', fntsz);
    h3 = plot(PEAKS(:, 1), PIX_SP, 'ob');
    hold on;
    grid on;
    set(gca, 'XLim', [PEAKS(1, 1) PEAKS(end, 1)]);
    set(gca, 'XTickLab', []);
    ylabel('PIX SP [m]');
end
%% PHASE 7 - Discard Bad Channels
if and(doPHASE7, length(discard >= 1))
    bad = [];
    for n = 1:length(discard)
        b = find(PEAKS(:, 1) == discard(n)); % index of nth bad channel
        bad = [bad b]; % all indices of bad channels
    end
    c = PEAKS(:, 1); % alternate array of channel numbers
    c(bad) = zeros(1, length(discard));
    % zero out bad channels so I can use 'find'
    ind = find(c); % finds all non-zero elements of c
    % Use 'ind' to trim arrays
    PEAKS = PEAKS(ind, :);
    REL_INT = REL_INT(ind);
    PIX_SP = PIX_SP(ind);
end
%% Retrieve 'impacts', print all data before saving
```

```
try load(impactsFile); end;
IMPACTS = impacts(PEAKS(:, 1)); % select only channel numbers being used
try
    REL_INT;
catch % not running the whole code
    REL_INT = NaN;
    PEAKS = NaN;
    PIX_SP = NaN;
    stt.REL_INT = 0; % make damn sure I don't write NaNs to the tree
    stt.PEAKS = 0;
    stt.PIX_SP = 0;
end
%% PHASE 8 - Save Settings to Tree
if doPHASE8
    try
        saveToTree3(saveToShots, stt, CAL_LAMBDA, LAMBDA, VOLTAGE, MASS,...
             PEAKS, REL_INT, PIX_SP, IMPACTS, hitsi3);
    end
end
```

The following functions are called by the above scripts.

```
function[data, X, Y, n_pix, n_chan] = calSVD(shot1, shot2)
try
    importdata(['Shot ' int2str(shot1) '.mat']);
    data = CineArray;
    clear CineArray TimeVector;
catch
    data = importdata(['shot' int2str(shot1) '.mat']);
    % [counts] (time x wavelength space x channel space)
    % Unflip images to correct for flip during Python conversion
    data = data(:, end:-1:1, end:-1:1);
end
% Arrange in vertical columns
[n_time, n_pix, n_chan] = size(data);
BDdat = NaN*zeros(n_chan * n_pix, n_time);
for n = 1:n_time
    BDdat(:, n) = reshape(squeeze(data(n, :, :)), n_chan * n_pix, 1);
end
clear data;
BDdat = BDdat - min(BDdat(:)); % subtract minimum value in data
% Perform SVD ----------------------
[U, S, V] = svd(BDdat, 'econ');
clear S V;
% Pull out first topo
topo1 = reshape(U(:, 1), n_pix, n_chan);
clear U;
% Make sure data is positive
if min(topo1(:)) < 0
    topo1 = -topo1;
end
[X, Y] = meshgrid(1:n_chan, 1:n_pix);
if shot2 ~= 0 % load the second shot, if it exists
```

```
    data = importdata(['shot' int2str(shot2) '.mat']);
    % [counts] (time x wavelength space x channel space)
    % Unflip images to correct for flip during Python conversion
    data = data(:, end:-1:1, end:-1:1);
    % Arrange in vertical columns
    [n_time, n_pix, n_chan] = size(data);
    BDdat = NaN*zeros(n_chan * n_pix, n_time);
    for n = 1:n_time
        BDdat(:, n) = reshape(squeeze(data(n, :, :)), n_chan * n_pix, 1);
    end
    clear data;
    BDdat = BDdat - min(BDdat(:)); % subtract minimum value in data
    % Perform SVD ----------------------
    [U, S, V] = svd(BDdat, 'econ');
    clear S V;
    % Pull out first topo
    topo2 = reshape(U(:, 1), n_pix, n_chan);
    clear U;
    % Make sure data is positive
    if min(topo2(:)) < 0
        topo2 = -topo2;
    end
    % Add topos from both shots together
    data = topo1 + topo2;
    clear topo1 topo2;
else
    data = topo1;
    clear topo1;
end
```

```
function peaks = fitAllChans(data, chanNums, firstCenter,...
     lastCenter, brightWing, approxFit, force, remove)
plotFit = 1;
% bin data into line
[n_pix, n_chan] = size(data);
lamBound = round(firstCenter(2)) - brightWing : round(lastCenter(2))...
     + brightWing;
chanBound = round(firstCenter(1)) - brightWing : round(lastCenter(1))...
     + brightWing;
% Catches in case brightWing extends beyond CCD
if chanBound(1) < 1
    chanBound = chanBound(find(chanBound == 1)) : chanBound(end);
end
if chanBound(end) > n_chan
    chanBound = chanBound(1):chanBound(find(chanBound == n_chan));
end
% Sum data in wavelength direction
data = sum(data(lamBound, chanBound), 1);
peaks(:, 1) = chanNums; % put channel numbers into peaks array
% Use simple interpolation to guess at peak locations in wavelength space
allChans = linspace(firstCenter(2), lastCenter(2),...
     chanNums(end) - chanNums(1) + 1);
peaks(:, 3) = allChans(chanNums); % remove dead channels
```

```
% Find local maxima using 'extrema' function
[xmax, imax, xmin, imin] = extrema(data); % xmax is local maxima, imax is indices
if ~isempty(remove)
    remove = remove - chanBound(1) + 1;
    chanBound
    for n = 1:length(remove)
        imax
        ind = find(remove(n) == imax);
        ind
        imax = imax([1:ind-1, ind+1:end]);
        find(imax ~= ind)
        imax
    end
end
if ~isempty(force)
    for n = 1:length(force) % index of 'data' does not start at 1.
    % Make 'force' correspond to index of data
        force(n) = find(force(n) == chanBound);
    end
    imax = [imax, force]; % add on forced index of maxima
end
centers = sort(imax); % sort indices in ascending order
amps = data(centers); % arrange maxima values to match indices
centers = chanBound(1) + centers - 1;
if centers(1) < firstCenter(1) - 1;
% catch if it finds a max at the very beginning
    centers = centers(2:end);
    amps = amps(2:end);
end
if centers(end) > lastCenter(1) + 1;
% catch if it finds a max at the very beginning
    centers = centers(1:end-1);
    amps = amps(1:end-1);
end
disp(['Number of extrema found: ' num2str(size(centers, 2))]);
disp(['Requested: ' num2str(length(chanNums))]);
if length(centers)>length(chanNums)
    centers=centers(1:length(chanNums))';
    amps=amps(1:length(chanNums));
end
% Plot summed Data
if plotFit
    S = get(0,'ScreenSize');
    fntsz = 20;
    h1 = figure('Visible','on','Name','Approximate Fit','Position',...
        [S(3)/12, S(4)/6, 5*S(3)/6 2*S(4)/3], 'Color', [1 1 1]);
    h2 = axes('Parent', h1, 'Position', [.1 .2 .8 .6], 'FontSize', fntsz);
    h3 = plot(chanBound, data, '-b');
    hold on;
    h4 = plot(centers, amps, '+r');
    for n = 1:length(centers)
        text(centers(n), 1.15*amps(n), int2str(chanNums(n)));
    end
    grid on;
```

```
    set(gca, 'XLim', [chanBound(1) chanBound(end)]);
    xlabel('Pixel Number (Real Space)');
    ylabel('Pixel Number (Wavelength Space)');
    % SAVE
    if approxFit.save
        fig_save = getframe(h1);
        [Xfig, mapfig] = frame2im(fig_save);
        imwrite(Xfig, [approxFit.file '.png']);
    end
end
size(peaks(:,2))
size(centers)
peaks(:, 2) = centers; % store approximate centers in peaks array
end
```

```
function[PEAKS, REL_INT, par, fits] = calGauss2D(peaks, data,...
     brightWing, xWing, indivFit)
% This function loops through each channel and fits a 2D Gaussian, then
% returns the parameters.
%
% par(:,1) = "volume" (the function is a normal distribution)
% par(:,2) = x0
% par(:,3) = y0
% par(:,4) = sigx
% par(:,5) = sigy
% par(:,6) = offset
plotFits = 0;
if plotFits
    S = get(0,'ScreenSize');
    fntsz = 24;
    h1 = figure('Visible','on','Name','Individual Fit','Position',...
        [S(3)/10, S(4)/8, 4*S(3)/5, 3*S(4)/4], 'Color', [1 1 1]);
end
par = NaN*zeros(size(peaks, 1), 6); % allocate fit parameter array
guesses = par; % initial guess array
try
    options = optimsetv61('lsqcurvefit'); % set options for curve fitting
catch
    options = optimset('lsqcurvefit'); % set options for curve fitting
end
options.TolFun = 1e-8 * max(data(:)); % set tolerance for curve fitting
f = @singletGauss2D; % function handle
for n = 1:size(peaks, 1) % loop through channels
    % set grid domain size and create mesh
    xBound = round(peaks(n, 2)) - xWing : round(peaks(n, 2)) + xWing;
    yBound = round(peaks(n, 3)) - brightWing : round(peaks(n, 3)) + brightWing;
    [X, Y] = meshgrid(xBound, yBound);
    Z = data(yBound, xBound); % Select subset of data for mesh
    if plotFits
        clf; % Plot Raw Data ----------------------------
        ax(1) = axes('Parent', h1, 'Position', [0.1 0.1 0.2 0.8],...
            'FontSize', fntsz);
        h3 = surf(X, Y, Z);
```

```
        hold on;
        shading interp;
        colormap jet;
        colorbar;
        view([0 90]);
        set(gca, 'FontSize', fntsz);
        set(gca, 'XLim', [xBound(1) xBound(end)], 'YLim',...
             [yBound(1) yBound(end)]);
        xlabel('Real Space');
        ylabel('Wavelength Space');
        title(['Raw Data, Channel ' num2str(peaks(n, 1))]);
    end
    % Initial Guesses
    sigx = 1.5;
    sigy = 1.0;
    vol = 6 * sigx * sigy * max(Z(:));
    guess = [vol, peaks(n, 2), peaks(n, 3), sigx, sigy, min(Z(:))];
    lb = [0.3*vol, guess(2)-0.5, guess(3)-1, 0.7*guess(4), 0.7*guess(5), 0];
    % lower bound
    ub = [1.8*vol, guess(2)+0.5, guess(3)+1, 1.3*guess(4), 1.3*guess(5),...
         0.5*max(Z(:))]; % upper bound
    % Reshape data and grid
    x(:, 1) = X(:);
    x(:, 2) = Y(:);
    z = Z(:);
    % Fit Gaussian ----------------------------------------
    [par(n, :), resnorm, residual, exitflag] = ...
        lsqcurvefit(f, guess, x, z, lb, ub, options);
    % create fine mesh --------------------------------
    nfx = 100;
    nfy = 200;
    xBoundf = linspace(xBound(1), xBound(end), nfx);
    yBoundf = linspace(yBound(1), yBound(end), nfy);
    [Xf, Yf] = meshgrid(xBoundf, yBoundf);
    % Reshape fine mesh for execution by Gaussian function
    xf(:, 1) = Xf(:);
    xf(:, 2) = Yf(:);
    zf = singletGauss2D(par(n, :), xf); % calculate fit on fine mesh
    Zf = reshape(zf, size(Xf, 1), size(Xf, 2)); % reshape into 2D image
    if plotFits
        % Plot Fit ----------------------
        ax(2) = axes('Parent', h1, 'Position', [.4 .1 .2 .8], 'FontSize', fntsz);
        h4 = surf(Xf, Yf, Zf);
        hold on;
        shading interp;
        colormap jet;
        colorbar;
        grid on;
        view([0 90]);
        set(gca, 'FontSize', fntsz);
        set(gca, 'XLim', [xBound(1) xBound(end)], 'YLim',...
             [yBound(1) yBound(end)]);
        xlabel('Real Space');
        title(['Gaussian Fit, Channel ' num2str(peaks(n, 1))]);
```

```
        % Link Z Axis
        zmin = min(Z(:));
        zmax = max(Z(:));
        zfmin = min(Zf(:));
        zfmax = max(Zf(:));
        set(ax(1), 'ZLim', [min([zmin zfmin]), max([zmax zfmax])]);
        set(ax(2), 'ZLim', [min([zmin zfmin]), max([zmax zfmax])]);
        % Plot Y 2D cross section data and fit ---------------
        ax(3) = axes('Parent', h1, 'Position', [.68 .55 .27 .35],...
             'FontSize', fntsz);
        h5 = plot(yBound, Z(:, ceil(size(Z, 2)/2)), 'or');
        set(h5, 'MarkerFaceColor', 'r');
        hold on;
        h6 = plot(yBoundf, Zf(:, ceil(size(Zf, 2)/2)), '-b');
        set(h6, 'LineWidth', 2);
        set(gca, 'XTickLabel', [], 'YTickLabel', []);
        set(gca, 'XLim', [yBound(1) yBound(end)]);
        grid on;
        title('\lambda Cross Section (y)');
        % Plot X 2D cross section data and fit ---------------
        ax(4) = axes('Parent', h1, 'Position', [.68 .1 .2 .35], 'FontSize',...
             fntsz);
        h7 = plot(xBound, Z(ceil(size(Z, 1)/2), :), 'or');
        set(h7, 'MarkerFaceColor', 'r');
        hold on;
        h8 = plot(xBoundf, Zf(ceil(size(Zf, 1)/2), :), '-b');
        set(h8, 'LineWidth', 2);
        set(gca, 'XTickLabel', [], 'YTickLabel', []);
        grid on;
        title('Spatial (x)');
        % Text -------------
        text(1.08, 0.7, ['\sigma_y = ' num2str(par(n, 5), 4)],...
             'Units', 'normalized', 'FontSize', fntsz);
        text(1.08, 0.4, ['\sigma_x = ' num2str(par(n, 4), 4)],...
             'Units', 'normalized', 'FontSize', fntsz);
        pause(0.1);
        % SAVE
        if and(indivFit.save, indivFit.chan == peaks(n, 1))
            fig_save = getframe(h1);
            [Xfig, mapfig] = frame2im(fig_save);
            imwrite(Xfig, [indivFit.file '.png']);
        end
    end % End plotting fits
    guesses(n, :) = guess;
    fits.Xf(:, :, n) = Xf; % stack all grids for plotting in main routine
    fits.Yf(:, :, n) = Yf;
    fits.Zf(:, :, n) = Zf;
end % End loop over channels
% Fill in parameters for PEAKS ------------------
PEAKS(:, 1) = peaks(:, 1); % channel numbers
PEAKS(:, 2:5) = par(:, 2:5);
% channel centers, x0 and y0, and standard deviations, sigx and sigy
% Convert 'volume' to 'amplitude' to 'area' in y direction only for REL_INT
amp = par(:, 1) ./ (2*pi * par(:, 4) .* par(:, 5)); % volume to amplitude
```

```
area = sqrt(2*pi) * par(:, 5) .* amp; % ampitude to area of 1D Gaussian (y only)
REL_INT = mean(area) ./ area; % actually INVERSE relative intensity
end
```

```
function z = singletGauss2D(par, x)
% par(1) = "volume" (the function is a normal distribution)
% par(2) = x0
% par(3) = y0
% par(4) = sigx
% par(5) = sigy
% par(6) = offset
% x is a 2 column array where x(:, 1) is mesh X and x(:, 2) is mesh Y
z = par(1)/(2*pi*par(4)*par(5)) * exp(-0.5 * (((x(:,1) - par(2)) ./...
     par(4)).^2 + ((x(:,2) - par(3)) ./ par(5)).^2)) + par(6);
end
```

```
function PIX_SP = fitMotor(shot3, shot4, PEAKS, motorSpeed, brightWing, xWing)
plotFits = 0;
% Load Data
data3 = importdata(['shot' int2str(shot3) '.mat']);
% [counts] (time x wavelength space x channel space)
time3 = importdata(['t' int2str(shot3) '.mat']); % [ms]
data3 = data3(:, end:-1:1, end:-1:1);
if shot4 ~= 0 % using both fibers
    data4 = importdata(['shot' int2str(shot4) '.mat']);
    % [counts] (time x wavelength space x channel space)
    time4 = importdata(['t' int2str(shot4) '.mat']); % [ms]
    data4 = data4(:, end:-1:1, end:-1:1);
    % Add movies together
    if length(time3) > length(time4)
        data = data4 + data3(1:length(time4), :, :);
        time = time4;
        clear data3 data4 time3 time4;
    else
        data = data3 + data4(1:length(time3), :, :);
        time = time3;
        clear data3 data4 time3 time4;
    end
else % only using one fiber
    data = data3;
    time = time3;
    clear data3 time3;
end
data = cast(data, 'double');
[n_time, n_pix, n_chan] = size(data);
%% Loop over Channels
try
    options = optimsetv61('lsqcurvefit'); % set options for curve fitting
catch
    options = optimset('lsqcurvefit'); % set options for curve fitting
end
```

```
options.TolFun = 1e-8 * max(data(:)); % set tolerance for curve fitting
f = @singletGauss2D; % function handle
centers = NaN*zeros(size(PEAKS, 1), length(time));
% initialize array to store centers for all time
if plotFits
    S = get(0,'ScreenSize');
    fntsz = 20;
    h1 = figure('Visible','on','Name','Individual Fit','Position',...
        [S(3)/10, S(4)/8, 4*S(3)/5, 3*S(4)/4], 'Color', [1 1 1]);
end
for n = 1:size(PEAKS, 1)
    % find real space bound for fitting domain
    xBound = round(PEAKS(n, 2)) - xWing : round(PEAKS(n, 2)) + xWing;
    for m = 1:n_time
        % find wavelength coordinate of starting peak
        [amp, y] = max(max(data(m, :, xBound))); % which column has the max
        [amp, y] = max(squeeze(data(m, :, xBound(y))));
        % which row in that column
        % find wavelength direction bound for fitting
        yBound = y - brightWing : y + brightWing;
        % catch in case out of bounds
        if yBound(1) < 1
            yBound = yBound(find(yBound == 1)) : yBound(end);
        end
        if yBound(end) > n_pix
            yBound = yBound(1) : yBound(find(yBound == n_pix));
        end
        % make grid
        [X, Y] = meshgrid(xBound, yBound);
        Z = squeeze(data(m, yBound, xBound));
        if plotFits
            clf; % Plot Raw Data ----------------------------
            ax(1) = axes('Parent', h1, 'Position', [.1 .1 .2 .8],...
                'FontSize', fntsz);
            h3 = surf(X, Y, Z);
            hold on;
            shading interp;
            colormap jet;
            colorbar;
            grid on;
            view([0 90]);
            set(gca, 'FontSize', fntsz);
            xlabel('Real Space');
            ylabel('Wavelength Space');
            title(['Channel ' num2str(PEAKS(n, 1)) ', Time pt. ' num2str(m)]);
        end
        % Initial Guesses
        sigx = PEAKS(n, 4);
        sigy = PEAKS(n, 5);
        vol = 6 * sigx * sigy * amp;
        guess = [vol, PEAKS(n, 2), y, sigx, sigy, min(Z(:))];
        lb = [0.4*vol, guess(2)-0.5, guess(3)-0.5, 0.8*guess(4),...
            0.8*guess(5), 0]; % lower bound
        ub = [1.8*vol, guess(2)+0.5, guess(3)+0.5, 1.2*guess(4),...
```

```
            1.2*guess(5), 0.5*amp]; % upper bound
        % Reshape data and grid
        x(:, 1) = X(:);
        x(:, 2) = Y(:);
        z = Z(:);
        % Fit Gaussian ----------------------------------------
        parameters = lsqcurvefit(f, guess, x, z, lb, ub, options);
        centers(n, m) = parameters(3); % save center in wavelength space only
        if plotFits
            % create fine mesh ------------------------------
            nfx = 100;
            nfy = 200;
            xBoundf = linspace(xBound(1), xBound(end), nfx);
            yBoundf = linspace(yBound(1), yBound(end), nfy);
            [Xf, Yf] = meshgrid(xBoundf, yBoundf);
            % Reshape fine mesh for execution by Gaussian function
            xf(:, 1) = Xf(:);
            xf(:, 2) = Yf(:);
            zf = singletGauss2D(parameters, xf); % calculate fit on fine mesh
            Zf = reshape(zf, size(Xf, 1), size(Xf, 2)); % reshape into 2D image
            % Plot Fit ----------------------
            ax(2) = axes('Parent', h1, 'Position', [.4 .1 .2 .8],...
                'FontSize', fntsz);
            h4 = surf(Xf, Yf, Zf);
            hold on;
            shading interp;
            colormap jet;
            colorbar;
            grid on;
            view([0 90]);
            set(gca, 'FontSize', fntsz);
            set(gca, 'XLim', [xBound(1) xBound(end)], 'YLim',...
                [yBound(1) yBound(end)]);
            xlabel('Real Space');
            title(['Fit, Channel ' num2str(PEAKS(n, 1)) ',...
                Time pt. ' num2str(m)]);
            % Link Z Axis
            zmin = min(Z(:));
            zmax = max(Z(:));
            zfmin = min(Zf(:));
            zfmax = max(Zf(:));
            set(ax(1), 'ZLim', [min([zmin zfmin]), max([zmax zfmax])]);
            set(ax(2), 'ZLim', [min([zmin zfmin]), max([zmax zfmax])]);
            pause(0.1);
        end % plotting
        clear x z xf zf; % when they change size, the dimensions mismatch
    end % time loop
end % channel loop
%% Calculate PIX_SP
PIX_SP = zeros(1, size(PEAKS, 1)); % allocate array
nm = motorSpeed * 1e-3 * time; % convert 'time' to [s], [s] to [nm]
S = get(0,'ScreenSize');
fntsz = 20;
h1 = figure('Visible','on','Name','Linear Fit','Position',...
```

```
    [S(3)/10, S(4)/8, 4*S(3)/5, 3*S(4)/4], 'Color', [1 1 1]);
for n = 1:size(PEAKS, 1)
    p = polyfit(centers(n, :)', nm, 1);
    PIX_SP(n) = abs(p(1));
    if plotFits
        clf; % Plot Raw Data and Fit ----------------------------
        h4 = axes('Parent', h1, 'Position', [.2 .2 .6 .6], 'FontSize', fntsz);
        h5 = plot(centers(n, :), p(1)*centers(n, :) + p(2), '-r');
        hold on;
        h6 = plot(centers(n, :), nm, 'ob');
        title(['Raw Wavelength vs. Pixel and Linear Fit, Channel '...
             num2str(PEAKS(n,1))]);
        ylabel('Wavelength [nm]');
        xlabel('Pixel (wavelength space)');
        pause(0.5);
    end
end
PIX_SP = 1e-9 * PIX_SP; % convert from nm to meters
end
```

```
function saveToTree3(saveToShots, stt, CAL_LAMBDA, LAMBDA, VOLTAGE,...
     MASS, PEAKS, REL_INT, PIX_SP, IMPACTS, hitsi3)
% SAVE TO TREE 3, UPDATED TO OBJECT ORIENTED JAVA FOR MDSPLUS
%% GET THE MDSPLUS DIRECTORY STUFF TOGETHER, SO WE CAN IMPORT IT
import MDSplus.*
for n = 1:length(saveToShots)
    shot = saveToShots(n);
    disp(['Editing ANALYSIS Tree for shot ' num2str(shot)]);
    % Open Tree for Edit JAVA
    if(~hitsi3)
        HitTree = Tree('analysis',shot,'EDIT');
    else
        HitTree = Tree('analysis3',shot,'EDIT');
    end
    try
        HitTree.setDefault(HitTree.getNode('.spectroscopy.ids'))
    catch
        HitTree.addNode('.SPECTROSCOPY', 'STRUCTURE')
        HitTree.addNode('.SPECTROSCOPY.IDS', 'STRUCTURE')
        HitTree.setDefault(HitTree.getNode('.spectroscopy.ids'))
    end
    if stt.CAL_LAMBDA
        CleanNode(HitTree,'CAL_LAMBDA');
    end
    if stt.LAMBDA
        CleanNode(HitTree,'IDS_LAMBDA');
    end
    if stt.VOLTAGE
        CleanNode(HitTree,'IDS_VOLTAGE');
    end
    if stt.MASS
        CleanNode(HitTree,'IDS_MASS');
    end
```

```
    if stt.PEAKS
        CleanNode(HitTree,'IDS_PEAKS');
    end
    if stt.REL_INT
        CleanNode(HitTree,'IDS_REL_INT');
    end
    if stt.PIX_SP
        CleanNode(HitTree,'IDS_PIX_SP');
    end
    if stt.IMPACTS
        CleanNode(HitTree,'IDS_IMPACTS');
    end
    % JAVA WRITE, CLOSE, AND OPEN AGAIN
    HitTree.write()
    HitTree.close()
    %%%%%%%%%%%%%%
    % Fill in data ( reopen to prevent further tree modification
    if(~hitsi3)
        HitTree = Tree('analysis', shot);
    else
        HitTree = Tree('analysis3', shot);
    end
    % spectroscopy will for sure be here by now.
    HitTree.setDefault(HitTree.getNode('.spectroscopy.ids'));
    if stt.CAL_LAMBDA
        AddData(HitTree,'CAL_LAMBDA',CAL_LAMBDA);
    end
    if stt.LAMBDA
        AddData(HitTree,'IDS_LAMBDA',LAMBDA);
    end
    if stt.VOLTAGE
        AddData(HitTree,'IDS_VOLTAGE',VOLTAGE);
    end
    if stt.MASS
        AddData(HitTree,'IDS_MASS',MASS);
    end
    if stt.PEAKS
        AddData(HitTree,'IDS_PEAKS',PEAKS);
    end
    if stt.REL_INT
        AddData(HitTree,'IDS_REL_INT',REL_INT);
    end
    if stt.PIX_SP
        AddData(HitTree,'IDS_PIX_SP',PIX_SP);
    end
    if stt.IMPACTS
        AddData(HitTree,'IDS_IMPACTS',IMPACTS);
    end
    HitTree.close()
end
end
function CleanNode(HitTree, Tag)
    disp(['Clearing ' Tag]);
    try
```

```
        current_node = HitTree.getNode(Tag);
    catch
        HitTree.addNode(Tag, 'NUMERIC')
        current_node = HitTree.getNode(Tag);
        current_node.addTag(Tag)
    end
end
function AddData(HitTree, Tag, Data)
        disp(['Modifying ' Tag]);
        current_node = HitTree.getNode(Tag);
        current_node.putData(MDSarg(Data))
        % MDSarg required for multidimensional arrays
        % to read: NATIVEvalue(node.getData())
end
```

C.2 Fitting

The following script, `Batch_Correct_2.m`, and functions load raw data, perform BD filtering and curve fitting, and save the results to a file. They can be found in the directory:

`/Users/Aaron/IDS/Fitting`

```
% UPDATED TO OBJECT ORIENTED MDSplus BY RIAN CHANDRA, WNTR 2014
% UPDATED TO NEW CINE TYPE BY RIAN CHANDRA, SPR 2014
% NB: Important Input settings Below %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all; close all; clc;
profile on;
import MDSplus.*
try
    addpath('~/IDS/Matlab/');
end
try
    addAllThePaths;
end
shots = [128580, 128581, 128586, 128587, 128588, 128593, 128595, 128596];
% 53.5 kHz standard set
nBDmodes = [];
% number of modes to save after BD filtering. Leave blank for no BD filtering
timeBound = [];
% time point bounds for BD filtering.  Leave blank to use whole movie
line = [1, 2, 3]; % spectral line number, from longest to shortest wavelength
useTree = 1;
% Use ONLY in cases where caliibration was unable to save to the tree
%Is the file/tree stucture from HIT-SI, or HIT-SI3?
hitsi3 = 0;
if (length(num2str(shots(1))) > 8)
    hitsi3 = 1;
end
% UNCOMMENT THE APPROPRIATE LABELS:
label1 = 'Impact Parameter [cm]'; % Toroidal Section
% label1 = 'Major Radius [cm]'; % Poloidal Section, fibers facing each other
```

```
label2 = 'Major Radius [cm]'; % Poloidal Section
% label1 = 'Displacement [cm]'; % Orthogonal Section
frame_sum = 1; % number of frames to sum (typically 1)
plots = 0; % makes plots for debugging
linuxDataPath = '/home/aaron/IDS/IDSdata/';
windowsDataPath = 'T:\IDS\Data Repository\';
s.sims = [];  % tag indicating type of simulation
        % 2 = NIMROD --- weighted by n^2 ---  NO temperature  --- 7x timeSum
        % 4 = NIMROD --- weighted by n^2 ---  NO temperature  --- no timeSum
        % 5 = PSI-TET --- weighted by n^2 --- with temperature --- 7x timeSum
        % 6 = NIMROD --- weighted by n^2 --- with temperature --- 7x timeSum
        % 7 = PSI-TET -- weighted by n^2 --- with temperature --- no timeSum
        % 8 = NIMROD --- weighted by n^2 --- with temperature --- no timeSum
        % LEAVE EMPTY [] IF NOT DOING SIMULATION
s.special = [99];
% tag indicating special shot number prefix, similar to simulations
s.spSum = 0; % add chords together for better statistics
s.chSum = [9:11; 12:14; 16:18; 19:21; 23:25; 26:28];
s.tSum = 0; % add frames together in a temporal pattern
s.sumPatt = [50, 12, 200]; % [time point to start from, length of pattern, end]
for n = 1:length(shots)
    tic
    if ~isempty(s.sims)
        s.sim = s.sims(n);
    else
        s.sim = [];
    end
    [out, param] = load_shot(shots(n), nBDmodes, timeBound, line,...
         frame_sum, plots, s, hitsi3, useTree);
    t = toc;
    disp(['Elapsed time is ' num2str(floor(t/60)) ' minutes, '...
         num2str(rem(t,60),2) ' seconds']);
    if ~isempty(s.sim)
        disp(['saving shot ' num2str(s.sim) num2str(shots(n))]);
    else
        disp(['saving shot ' num2str(shots(n))]);
    end
    if(~hitsi3)
        HitTree = Tree('hitsi',shots(n));
        Data = HitTree.getNode('\I_INJ_X');
        iinjx=Data.getData().data.getDoubleArray;
        %[iinjxTime, dummy, iinjx] = Data.getData();
        iinjxTime =  NATIVEvalue(Data.getDimensionAt(0).data());
        dat(1).iinjxTime = 1e3 * iinjxTime;
        dat(1).iinjx = 1e-3 * iinjx;
        Data = HitTree.getNode('\I_INJ_Y');
        iinjy = Data.getData().data.getDoubleArray;
        iinjyTime = NATIVEvalue(Data.getDimensionAt(0).data());
        %[iinjyTime, dummy, iinjy] = Data.getData();
        dat(1).iinjyTime = 1e3 * iinjyTime;
        dat(1).iinjy = 1e-3 * iinjy;
        Data = HitTree.getNode('\I_TOR_SPAAVG');
        %[ItorTime, dummy, Itor]=Data.getData();
        Itor = Data.getData().data.getDoubleArray;
```

```
        ItorTime =  NATIVEvalue(Data.getDimensionAt(0).data());
        dat(1).ItorTime = 1e3 * ItorTime;
        dat(1).Itor = 1e-3 * Itor;
        mdsclose();
    else
        HitTree = Tree('hitsi3',shots(n));
        Data = HitTree.getNode('\I_INJ_A');
        dat(1).iinja = 1e-3 * NATIVEvalue(Data.getData.data());
        dat(1).iinjaTime = 1e3 *  NATIVEvalue(Data.getDimensionAt(0).data());
        Data = HitTree.getNode('\I_INJ_B');
        dat(1).iinjb = 1e-3 *NATIVEvalue(Data.getData.data());
        dat(1).iinjbTime = 1e3 *  NATIVEvalue(Data.getDimensionAt(0).data());
        Data = HitTree.getNode('\I_INJ_C');
        dat(1).iinjc = 1e-3 *NATIVEvalue(Data.getData.data());
        dat(1).iinjcTime = 1e3 *  NATIVEvalue(Data.getDimensionAt(0).data());
        Data = HitTree.getNode('\I_TOR_SPAAVG');
        Itor = Data.getData().data.getDoubleArray;
        ItorTime =  NATIVEvalue(Data.getDimensionAt(0).data());
        dat(1).ItorTime = 1e3 * ItorTime;
        dat(1).Itor = 1e-3 * Itor;
        mdsclose();
    end
    for nn = 1:length(line)
        dat(nn).temp = out(nn).temp;
        dat(nn).vel = 1e-3 * out(nn).vel; % save in km/s
        dat(nn).int = out(nn).int;
        dat(nn).fit_par = out(nn).fit_par;
        % Curve fit parameters [time, channel, 1:6]
        dat(nn).bounds = out(nn).bounds;
        % Bounds for curve fitting grid, (time) x (channel) x (x1 x2 y1 y2)
        dat(nn).guesses = out(nn).guesses;
        % initial guess parameters, same size as fit_par
        if param.calcError
            dat(nn).dparam = out(nn).dparams; % parameter uncertainty structure
            dat(nn).stddev = out(nn).stddev;
            % standard deviation of every data point
            dat(nn).intU = out(nn).intU; % upper error bar for intensity
            dat(nn).intL = out(nn).intL;
            dat(nn).tempU = out(nn).tempU; % upper error bar for temperature
            dat(nn).tempL = out(nn).tempL;
            dat(nn).velU = 1e-3 * out(nn).velU; % upper error bar for velocity
            dat(nn).velL = 1e-3 * out(nn).velL;
        end
        if exist('out.resnorm', 'var') % DNE if using LM curve fitting method
            dat(nn).residual = sqrt(out(nn).resnorm); % residual
        end
        if exist('out.exp', 'var')
            dat(nn).exp = out(nn).exp;
            dat(nn).corrected = out(nn).corrected;
        end
    end
    dat(1).time = out(1).time;
    dat(1).raw = out(1).data; % raw data points
    dat(1).peaks = param.peaks(:, 1);
```

```
    dat(1).param = param; % param structure
    dat(1).shotRef = param.shotRef; % real shot number
    dat(1).impacts = param.impacts;
    dat(1).title = ['Shot ' num2str(dat(1).shotRef)];
    dat(1).label1 = label1;
    dat(1).label2 = label2;
    try
        save([linuxDataPath 'dat' num2str(s.sim) num2str(shots(n))...
            num2str(nBDmodes)], 'dat')
        % save 'dat' structure as 'dat<shot><nBDmodes>.mat'
    catch
        save([windowsDataPath 'dat' num2str(s.sim) num2str(shots(n))...
            num2str(nBDmodes)], 'dat')
        % save 'dat' structure as 'dat<shot><nBDmodes>.mat'
    end
end
```

```
function[out, param] = load_shot(shot, nBDmodes, timeBound, line,...
     frame_sum, plots, s, hitsi3, useTree)
% This code handles all the analysis of a MultiChord IDS shot.  The only
% inputs it needs are the shot number, number of frames to sum
% (typically 1), time override, and peak locations.
% If 't_override' is set to 0, the time
% to analyze will be determined by 'shotDuration.m', or 't_override' can be a
% two value vector with start and end times in ms.
%
% Load Constants and Parameters
[param, options] = loadParams(shot, line, hitsi3, useTree);
assignin('base','param',param)
%% Special Preparations
size(param.peaks)
%% BD filtering
if isempty(nBDmodes)
    if ~isempty(s.sim)
        shot = str2num([int2str(s.sim) int2str(shot)]);
    end
    if ~isempty(s.special)
        shot = str2num([int2str(s.special) int2str(shot)]);
    end
    data = importdata(['shot' int2str(shot) '.mat']);
    % [counts] (time x wavelength space x channel space)
    time = importdata(['t' int2str(shot) '.mat']); % [ms]
    newCineType = 0;
    % Unflip images to correct for flip during Python conversion
    data = data(:, end:-1:1, end:-1:1);
    data = cast(data, 'double'); % comes in as 'uint16' and ruins everything
else
    [data, time] = BDfilter(shot, nBDmodes, timeBound, param, s);
end
%% Frame averaging, if applicable
if frame_sum ~= 1
    [time, data] = frameSum(time, data, frame_sum, param);
end
```

```
%% Summing chords together
if s.spSum
    [data, param] = chordSum(data, param, s);
end
%% Adding a pattern of time points together
if s.tSum
    [data, time, param] = timeSum(data, param, s);
end
%% Calculations
% Gaussian Fitting
for nn = 1:length(line)
    tic
    [fit_par, dPar, bounds, stddev, param, guesses] = ...
        gaussFit_2D(data, param, options, nn);
    toc
    % Calculate Temperature, Velocity, Area
    [temp, vel, int, param] = calcPhysics(fit_par, param, s, nn);
    %% Error
    if param.calcError
        % Caulculate Upper Error Physics
        [tempU, velU, intU, ~] = calcPhysics(fit_par + dPar, param, s, nn);
        % Caulculate Lower Error Physics
        [tempL, velL, intL, ~] = calcPhysics(fit_par - dPar, param, s, nn);
        % Convert to Error Bars
        dintU = intU - int;
        dtempU = tempU - temp;
        dvelU = velU - vel;
        dintL = int - intL;
        dtempL = temp - tempL;
        dvelL = vel - velL;
    end
    %% Output
    % Sanitize output by discarding ridiculous data
    [temp, vel] = sanitizeOut(temp, vel, param.limits);
    % Bundle all outputs as structure
    out(nn).time = time; % (n_time)
    out(nn).temp = temp; % (n_time) x (param.n_chan)
    out(nn).vel = vel; % (n_time) x (param.n_chan)
    out(nn).int = int;
    % intensity, relatively calibrated (n_time) x (param.n_chan)
    % out.resnorm = resnorm; % (n_time) x (param.n_chan)
    out(nn).fit_par = fit_par;
    % (n_time) x (param.n_chan) x (6 (number of parameters))
    out(nn).data = data;
    % (n_time) x (pixels in wavelength space) x (pixels in real space)
    out(nn).bounds = bounds; % (n_time) x (param.n_chan) x (4: x1, x2, y1, y2)
    out(nn).guesses = guesses;
    % (n_time) x (param.n_chan) x (6 (number of parameters))
    if param.calcError
        % Added when switched to Levenberg-Marquardt error method
        out(nn).dparams = dPar;
        % same size as 'out.params'- standard error in params
        out(nn).stddev = stddev; % standard deviation for every data point
        out(nn).intU = dintU; % upper error bar for intensity
```

```
        out(nn).intL = dintL; % lower error bar for intensity
        out(nn).tempU = dtempU; % upper error bar for temperature
        out(nn).tempL = dtempL; % lower error bar for temperature
        out(nn).velU = dvelU; % upper error bar for velocity
        out(nn).velL = dvelL; % lower error bar for velocity
    end
end
assignin('base','out',out);
end
```

```
function[param, options] = loadParams(shot, line, hitsi3, useTree)
    if useTree % try using the tree, if that doesnt work,
        % try importing callibration from file
        import MDSplus.*;
        if(~hitsi3)
        % I ASSUME THAT ALL THE NODES I WANT EXIST ON HITSI AND HITSI3
            HitTree = Tree('hitsi', shot);
        else
            HitTree = Tree('hitsi3', shot);
        end
        HitTree.setDefault(HitTree.getNode('.analysis.spectroscopy.ids'));
        Data = HitTree.getNode('IDS_PEAKS');
        param.peaks = NATIVEvalue(Data.getData());
        % [pixels] (n_channels x 5) chan number, x, y, sig_x, sig_y
        Data = HitTree.getNode('IDS_MASS');
        IonMass = (1.66e-27) * NATIVEvalue(Data.getData()); % [AMU] => [kg]
        param.IonMass = IonMass(line); % select mass of ion of interest
        Data = HitTree.getNode('CAL_LAMBDA');
        param.CalLam = NATIVEvalue(Data.getData()); % [m]
        Data = HitTree.getNode('IDS_LAMBDA');
        LineLam = NATIVEvalue(Data.getData()); % [m]
        param.LineLam = LineLam;
        % for doublet, this automatically selects both lines
        % Used to be LineLam(line)
        Data = HitTree.getNode('IDS_PIX_SP');
        param.PIX_SP = NATIVEvalue(Data.getData());
        % [m] (1 x n_channels) wavelength spacing per channel
        Data = HitTree.getNode('IDS_REL_INT');
        param.REL_INT = NATIVEvalue(Data.getData());
        % INVERSE relative intensity,
        % ie: multiply by fit area to calibrate
        Data = HitTree.getNode('IDS_IMPACTS');
        param.impacts = NATIVEvalue(Data.getData());
        mdsclose();
    else
        temp = importdata(['T:\IDS\Calibration\Calibration repository\'...
             int2str(shot) '.mat'] ); % load in params from file
        assignin('base','temp',temp);
        param.peaks = temp.PEAKS;
        param.IonMass = (1.66e-27)*temp.MASS(line(1));
        param.CalLam = temp.CAL_LAMBDA;
        param.LineLam = temp.LAMBDA(line);
        param.PIX_SP = temp.PIX_SP;
```

```
        param.REL_INT = temp.REL_INT;
        param.Impacts = temp.IMPACTS;
        assignin('base','param',param)
    end
    param.peaks(8,2)=71.75;param.peaks(22,2)=136.5;
    param.dbl = 0; % flag indicating single line- not a doublet
    assignin('base','param',param);
    param.shotRef = shot;
    % Adjust Center from calibration line to line of interest- NB: for DOUBLET,
    % center is arbitrarily chosen to be the smaller wavelength line ***
    for n = 1:length(line)
        param.Center(:, n) = param.peaks(:, 3) - ...
            (param.CalLam - param.LineLam(line(n))) ./ param.PIX_SP;
    end
    % Establish Constants
    param.limits = [0 100; -40e3 40e3];
        % limits for sanitizing final output- throws out data  that exceeds the
        % above limits for [temp; vel]
    param.n_chan = size(param.peaks, 1);
    param.calcError = 1; % calculate uncertainty by LM method - presently broken
    param.ampThresh = 5; % [pixels], amplitude threshold TEMP SET TO ONE
    param.xTol = 0.5;
    % [pixels], tolerance of recentering on line in spatial dimension
    param.yTol = 4.5; % [pixels], tolerance for velocity offset
    param.xWing = 2; % real-space domain for fitting. 1 = 3 wide, 2 = 5 wide
    param.yWing = 7; % wavelength-space domain. 5 = 11 wide, etc...
    try
        options = optimsetv61('lsqcurvefit'); % set options for curve fitting
    catch
        options = optimset('lsqcurvefit'); % set options for curve fitting
    end
    param.kBoltz = 1.3807e-23; % Boltzmann's Constant [SI]
    param.c = 2.99792458e8; % speed of light [SI]

end
```

```
function [data, time] = BDfilter(shot, nBDmodes, timeBound, param, s)
% The '3' version does a better job- now, tight bounds are set around the
% toroidal and poloidal data, they are individually BD'd, and then placed
% onto an array of zeros.  Thus, the movie is the same size as the original
% so the channel numbers match.
plotFits = 0;
BDwing = param.yWing + 1; % number of points to either side of center
% in wavelength space to include in BD domain
modes = 1:nBDmodes; % array of mode numbers to recombine and add together
first = param.peaks(1, 2);
last = param.peaks(find(param.peaks(:, 1) <= 36, 1, 'last'), 2);
chan_bound_t = floor(first):ceil(last);
first = param.peaks(find(param.peaks(:, 1) >= 37, 1), 2);
last = param.peaks(end, 2);
chan_bound_p = floor(first):ceil(last);
lam_bound = round(param.Center(1, end)) - BDwing : ...
    round(param.Center(1, 1)) + BDwing;
```

```
% Since 'Center' is ordered longes to shortest the indexing is a little
% weird.
%% Load Data Array
if ~isempty(s.sim)
    shot = str2num([int2str(s.sim) int2str(shot)]); % FUCK YEAH
end
data = importdata(['shot' int2str(shot) '.mat']);
% [counts] (time x wavelength space x channel space)
time = importdata(['t' int2str(shot) '.mat']); % [ms]
data = data(:, end:-1:1, end:-1:1); % removed x and y flip
newCineType = 0;
[n_time_o, n_pix_o, n_chan_o] = size(data);
% Account for the possibility of no wavelength bounds
if isempty(lam_bound)
    lam_bound = 1:n_pix_o;
end
% Account for empty channel bound
if isempty(chan_bound_t)
    chan_bound_t = 1:n_chan_o;
end
if isempty(chan_bound_p)
    chan_bound_p = 1:n_chan_o;
end
% BD specific time bound
if isempty(timeBound)
   timeBound = 1:length(time)-1;
end
time = time(timeBound);
for m = 1:2
    if m == 1 % toroidal fiber
        chan_bound = chan_bound_t;
    else % poloidal fiber
        chan_bound = chan_bound_p;
    end
    %isolate time in new/old filetypes
    if(~newCineType)
        % Record size for recombining
        [n_time_o, n_pix_o, n_chan_o] = size(data)
        data2 = data(timeBound, lam_bound, chan_bound);
        % Arrange in vertical columns
        [n_time, n_pix, n_chan] = size(data2);
        BDdat = NaN*zeros(n_chan * n_pix, n_time);
        for n = 1:n_time
            BDdat(:, n) = reshape(squeeze(data2(n, :, :)), n_chan * n_pix, 1);
        end
    else
        [ n_pix_o, n_chan_o, n_time_o] = size(data);
        % DONT MODEFY DATA IN THE LOOP:
        % BECAUSE IT GOES TWICE, IT WILL TRY TO TRIM IT AGAIN, WHICH IS BAD
        data2=data(lam_bound,chan_bound,timeBound);
        % Arrange in vertical columns
        [n_pix, n_chan, n_time] = size(data2);
        BDdat = NaN * zeros(n_chan * n_pix, n_time);
        for n = 1:n_time
```

```
            BDdat(:, n) = reshape(squeeze(data2(:, :, n)), n_chan * n_pix, 1);
        end
    end
    % Subtract off minimum value to reduce noise floor
    minVal = min(BDdat(:));
    if plotFits
        disp(['Min = ' num2str(minVal)]);
        disp(['Median = ' num2str(median(BDdat(:)))]);
        disp(['Mean = ' num2str(mean(BDdat(:)))]);
        disp(['Std = ' num2str(std(BDdat(:)))]);
    end
    BDdat = BDdat - minVal; % subtract minimum value in data
    %% Perform SVD ----------------------
    [U, S, V] = svd(BDdat, 'econ');
    Ak = diag(S);
    % Rearrange data as it was before
    %% NOTE: LEAVING TOPOS IN (T,X,Y) FORMAT FOR CONSISTANCY.
    topos = zeros(n_time, n_pix, n_chan);
    for n = 1:n_time % actually a loop over topo modes
        topos(n, :, :) = reshape(U(:, n), 1, n_pix, n_chan);
    end
    %% Plotting
    if plotFits
        [X, Y] = meshgrid(chan_bound, lam_bound);
        S1 = get(0,'ScreenSize');
        lnwdth = 2;
        fntsz = 20;
        h1 = figure('Visible','on','Name','Topo 1','Position',...
            [S1(3)/4, S1(4)/6, S1(3)/2 2*S1(4)/3], 'Color', [1 1 1]);
        h2 = axes('Parent', h1, 'Position', [.1 .1 .8 .8], 'FontSize', fntsz);
        h3 = surf(X, Y, Ak(1)*squeeze(topos(1, :, :)));
        hold on;
        shading interp;
        colormap jet;
        colorbar;
        grid on;
        view([0 90]);
        title(['First Topo, shot ' num2str(shot)]);
        xlabel('Pixel Number (real space)');
        ylabel('Pixel Number (wavelength space)');
        set(h3, 'LineWidth', lnwdth);
        set(gca, 'FontSize', fntsz, 'XLim', [1 n_chan], 'YLim', [1 n_pix]);
    end
    %% Save Data for Recombining
    if m == 1
        Ak_t = Ak(modes); % save weights
        V_t = V(:, modes); % save chronos
        topos_t = topos(modes, :, :); % save topos
    else
        Ak_p = Ak(modes); % save weights
        V_p = V(:, modes); % save chronos
        topos_p = topos(modes, :, :); % save topos
    end
    clear Ak V topos
```

```
end
%% Recombine Selected Modes into False Shot ASSUME RECOMBINE IS 1
assignin('base','data2',data2)
% NOTE: CHANGED N_TIME_O TO N_TIME:
% THIS SHOULD ONLY MATTER WHEN TIMEBOUND IS ACTIVE
data = zeros(n_time, n_pix_o, n_chan_o);
% THIS SHOULD BE COMPATABLE WITH NEW CINE ARQUETECTURE
data2 = data; % initialize both to zeros
for n = 1:length(modes)
    for m = 1:size(V_t, 1) % loop over time
        data2(m, lam_bound, chan_bound_t) = ...
            Ak_t(n) * V_t(m, n) * topos_t(n, :, :);
        data2(m, lam_bound, chan_bound_p) = ...
            Ak_p(n) * V_p(m, n) * topos_p(n, :, :);
    end
    data = data + data2;
end
toc
clear data2 Ak_t V_t topos_t Ak_p V_tp topos_p;
```

```
function[fit_par, dPar, bounds, stddev, param, guesses] = ...
    gaussFit_2D(data, param, options, nn)
% Revamped 6-4-13 by ACH to handle 2D fitting
[n_time, ~, ~] = size(data); % time, wavelength, spatial
fit_par = NaN * zeros(n_time, param.n_chan, 6); % preallocate fit parameters
guesses = fit_par; % preallocate array of initial guesses
bounds = NaN * zeros(n_time, param.n_chan, 4); % preallocate grid bounds array
n_pts = (2 * param.xWing + 1) * (2 * param.yWing + 1);
blank1 = NaN * ones(1, 6); % overwrite parameters if found to be bad
if param.calcError
    blank2 = NaN * ones(1, n_pts); % overwrite stddev data if bad
    stddev = NaN * zeros(n_time, param.n_chan, n_pts);
    % preallocate standard deviation
    % array
    dPar = fit_par; % uncertainty in fit parameters
    dp = [0.001, 0.001, 0.001, 0.001, 0.001, 0.001];
    % fractional increment of 'p' for numerical derivatives
else
    stddev = NaN; dPar = NaN;
end
f1 = @singletGauss2D; % function handle for singlet gaussian function
f2 = @singletGauss2DLM;
% LM code must have slightly different form
% of function
c = NaN; % not sure if necessary...
for n = 1:n_time
    for m = 1:param.n_chan
        % Make Zero-Velocity Bounds ---------------------------------------
        x0 = round(param.peaks(m, 2));
        y0 = round(param.Center(m, nn));
        xBound = x0 - param.xWing : x0 + param.xWing;
        yBound = y0 - param.yWing : y0 + param.yWing;
        % Save Bounds for Plotting Later ----------------------------------
```

```
        bounds(n, m, :) = [xBound(1), xBound(end), yBound(1), yBound(end)];
        % Make Grid ------------------------------------------------------
        [X, Y] = meshgrid(xBound, yBound);
        Z = squeeze(data(n, yBound, xBound));
        % Initial Guesses ------------------------------------------------
        amp = max(Z(:)) - min(Z(:)); % brightest pixel in domain - dimmest
        if amp > param.ampThresh
        % difference between any two pixels is above threshold
            sigx = param.peaks(m, 4);
            sigy = param.peaks(m, 5);
            vol = 6 * sigx * sigy * amp;
            guess = [vol, param.peaks(m, 2), y0, sigx, sigy, min(Z(:))];
            guesses(n, m, :) = guess; % save for post processing
            lb = [0, guess(2)-param.xTol, guess(3)-param.yTol, 0.8*guess(4),...
                 0.9*guess(5), 0];
            ub = [3*vol, guess(2)+param.xTol, guess(3)+param.yTol, ...
                1.2*guess(4), 4*guess(5), max(Z(:))];
            % Give the temperature guess a little boost to start
            guess(5) = 1.5 * guess(5);
            % Reshape Data and Grid ------------------------------------------
            x(:, 1) = X(:);
            x(:, 2) = Y(:);
            z = Z(:);
            % Curve Fit ------------------------------------------------------
            [fit_par(n, m, :), resnorm, residual, exitflag] = ...
                lsqcurvefit(f1, guess, x, z, lb, ub, options);
            % Exclude Bad Data -----------------------------------------------
            % Find Amplitude
            amp = fit_par(n, m, 1) / (2*pi * fit_par(n, m, 4) * ...
                    fit_par(n, m, 5));
            if amp < param.ampThresh
                fit_par(n, m, :) = blank1;
                if param.calcError
                    dPar(n, m, :) = blank1;
                    stddev(n, m, :) = blank2;
                end
            elseif param.calcError
                fit_par(n, m, 1) = fit_par(n, m, 1) - 3;
                [p_fit, Chi_sq, dPar(n, m, :), stddev(n, m, :), corr, ...
                    R2, cvg_hst] = lm(@singletGauss2DLM, fit_par(n, m, :),...
                         x, z, 0.001, dp);
            end
        else % Do not attempt to fit
            fit_par(n, m, :) = blank1;
            if param.calcError
                dPar(n, m, :) = blank1;
                stddev(n, m, :) = blank2;
            end
        end % Attempt the fit at all
    end % Channel Loop
end % Time Loop
end
```

```
function z = singletGauss2D(par, x)
% par(1) = "volume" (the function is a normal distribution)
% par(2) = x0
% par(3) = y0
% par(4) = sigx
% par(5) = sigy
% par(6) = offset
% x is a 2 column array where x(:, 1) is mesh X and x(:, 2) is mesh Y
z = par(1)/(2*pi*par(4)*par(5)) * exp(-0.5 * (((x(:,1) - par(2)) ./ ...
        par(4)).^2 + ((x(:,2) - par(3)) ./ par(5)).^2)) + par(6);
end
```

```
function z = singletGauss2DLM(x, par, c)
z = par(1)/(2*pi*par(4)*par(5)) * exp(-0.5 * (((x(:,1) - par(2)) ./ ...
        par(4)).^2 + ((x(:,2) - par(3)) ./ par(5)).^2)) + par(6);
end
```

```
function[temp, vel, int, param] = calcPhysics(fit_par, param, s, nn)
[n_time, n_chan, n_par] = size(fit_par); % determine size of data array
%% Temperature
% Calculate Instrument Temperatures
sigym = param.PIX_SP .* param.peaks(:, 5);
param.Inst_Temp(:, nn) = sigym.^2 * param.c^2 * param.IonMass(nn) / ...
    (param.CalLam^2 * param.kBoltz * 11605);
% Convert Instrument Temperature to dimensions of data
Inst_Temp = meshgrid(param.Inst_Temp(:, nn), 1:n_time);
% Expand 'PIX_SP' into same dimensions as Guassian width data
PIX_SP = meshgrid(param.PIX_SP, 1:n_time);
% Calculate Temperature
UnCorr_Temp = squeeze(fit_par(:, :, 5)).^2 .* PIX_SP.^2 .* param.c^2 .* ...
    param.IonMass(nn) ./ (mean(param.LineLam(nn))^2 * param.kBoltz * 11605);
temp = UnCorr_Temp - Inst_Temp; % subtract instrument temperature
%% Velocity
% Expand 'Center' into same dimensions as Gaussian offset data
center = meshgrid(param.Center(:, nn), 1:n_time);
% calculate velocity
vel = param.c * (squeeze(fit_par(:, :, 3)) - center) .* ...
    PIX_SP ./ param.LineLam(nn);
%% Intensity
% convert volume to amplitude
amp = squeeze(fit_par(:, :, 1)) ./ (2*pi * squeeze(fit_par(:, :, 4)) .* ...
    squeeze(fit_par(:, :, 5)));
% convert amplitude to area
area = sqrt(2*pi) * squeeze(fit_par(:, :, 5)) .* amp;
if isempty(s.sim) % real IDS data
    [rel_int, dummy] = meshgrid(param.REL_INT, 1:n_time);
    % expand REL_INT onto data sized grid
    int = rel_int .* area;
    disp('Intensity corrected because data is real data from IDS');
else % Simulation Data - DO NOT correct for fiber throughput
```

```
    int = area;
    disp('Intensity NOT corrected because data is from simulation');
end
end
```

```
function[temp, vel] = sanitizeOut(temp, vel, limits)
[n_time, n_chan] = size(temp);
for n = 1:n_time
    for m = 1:n_chan
        if or(temp(n, m) > limits(1, 2), temp(n, m) < limits(1, 1))
            temp(n, m) = NaN;
        end
        if or(vel(n, m) > limits(2, 2), vel(n, m) < limits(2, 1))
            vel(n, m) = NaN;
        end
    end
end
end
```

C.3 Synthetic Diagnostic

The following python script extracts parameters along chords in simulation output. The script and location are:

`/Users/Aaron/IDS/Visit/launchVisit.py`

```
# -*- coding: utf-8 -*-
from numpy import *
from scipy import *
import scipy.io as sio
import time
import sys
sys.path.append("/home/aaron/Software/visit2_7_1.linux-x86_64/2.7.1/linux-\
    x86_64/lib/site-packages")
import vtk
reader = vtk.vtkUnstructuredGridReader()
from visit import *
"""
Toroidal midplane, 129499
"""
vScale = 3e-6 # Vector scaling
nVectors = 2000 # number of vectors to plot
pMin = -5e4 # Pseudocolor velocity min
pMax = 5e4 # Pseudocolor velocity max
plotNim = 0
nimPath = "fin_beta_ramp"
nimTimes = range(110, 294)
#nimTimes = range(0, 294)
plotTet = 1
```

```
tetPath = "aaronData_141022" # First Dynamic PSI-TET run
#tetTimes = tuple(map(tuple, range(1, 75))) # all times from this
#tetTimes = range(193, 209)
tetTimes = range(0, 209)
plotIds1 = 0
idsShot1 = "12949910" # string
idsTP1 = "t" # "t" for toroidal fiber, "p" for poloidal fiber
plotIds2 = 0
idsShot2 = "12949910" # string
idsTP2 = "p" # "t" for toroidal fiber, "p" for poloidal fiber
idsTimes = (167, 170) # shift down by 1 from vtk file number
idsvectorColor = (255, 0, 0, 0) # "firebrick" red
idslineWidth = 0
# Window Information
x0 = 2600
y0 = 50
width = 2400
height = 1000
# SliceAtttributes
torPlane = 1 # true for midplane slice
polPlane = 0 # true for poloidal plane slice
theta = 135
boxAtt = (-0.6, 0.6, -0.6, 0.6, -0.6, 0.6)
viewportCoords = (0.25, 0.95, 0.15, 0.85)
windowCoords = (-0.6, 0.7, -0.6, 0.7)
# Line Out Settings
makeLineOut = 1
#nSamples = 1000
lineOutV = 1 # make Velocity LineOut
lineOutn = 1 # make density LineOut
lineOutT = 1 # make temperature LineOut
# Geometry settings, related to 'makeLineOutGeom' Matlab code
tor = 1
config = 2
databaseInfoFlag = 0
timeInfoFlag = 1
""" ---------------------------------------------------
Begin actual Scrip
--------------------------------------------------- """
def idsVectors(idsShot, idsTP):
    OpenDatabase("~/IDS/Visit/IDSvtk/ids_" + idsShot + idsTP + \
        "_*.vtk database")
    idsTS = GetActiveTimeSlider()
    SetTimeSliderState(idsTimes[0])
    # Get number of points
    reader.SetFileName("/home/aaron/IDS/Visit/IDSvtk/ids_" + \
        idsShot + idsTP + "_1.vtk")
    reader.ReadAllScalarsOn()  # Activate the reading of all scalars
    reader.Update()
    data = reader.GetOutput()
    AddPlot("Vector", "V")
    v = VectorAttributes()
    v.nVectors = data.GetNumberOfPoints() / 2
    v.useLegend = 0
```

```
    v.autoScale = 0
    v.scale = vScale
    v.colorByMag = 0
    v.glyphLocation = 0
    v.vectorColor = idsvectorColor
    v.lineWidth = idslineWidth
    SetPlotOptions(v)
    SetDefaultPlotOptions(v) # ensures future instances use these settings
    return idsTS
Launch()
SetWindowArea(x0, y0, width, height)
a = GetAnnotationAttributes()
a.userInfoFlag = 0 # disable user info
a.databaseInfoFlag = databaseInfoFlag
a.timeInfoFlag = timeInfoFlag
if polPlane:
    a.axes2D.xAxis.title.userTitle = 1
    a.axes2D.xAxis.title.title = xAxistitle
    a.axes2D.yAxis.title.userTitle = 1
    a.axes2D.yAxis.title.title = yAxistitle
SetAnnotationAttributes(a)
if plotNim:
    OpenDatabase("~/IDS/NimrodData/" + nimPath + "/DUMP/dump_*_b.vtk database")
    nimTS = GetActiveTimeSlider()
    SetTimeSliderState(nimTimes[0])
    simTimes = nimTimes # used for saving LineOut files
    simCode = 'nim'
    simFile = '/home/aaron/IDS/NimrodData/' + nimPath + '/LO/'
    DefineScalarExpression("Vx", "ve[0]")
    DefineScalarExpression("Vy", "ve[1]")
    DefineScalarExpression("Vz", "ve[2]")
    DefineVectorExpression("V", "ve")
    if makeLineOut and lineOutn:
        DefineScalarExpression("n", "nd")
    if makeLineOut and lineOutT:
        DefineScalarExpression("T", "tion")
if plotTet:
    OpenDatabase("~/IDS/PsitetData/" + tetPath + "/DUMP/out_*.xmf database")
    tetTS = GetActiveTimeSlider()
    SetTimeSliderState(tetTimes[0])
    simTimes = tetTimes # used for saving LineOut files
    simCode = 'tet'
    simFile = '/home/aaron/IDS/PsitetData/' + tetPath + '/LO/'
    DefineScalarExpression("Vx", "V[0]")
    DefineScalarExpression("Vy", "V[1]")
    DefineScalarExpression("Vz", "V[2]")
    if makeLineOut and lineOutn:
        DefineScalarExpression("n", "N")
    # 'T' already exists as a variable in PSI-TET
if plotNim or plotTet:
    # Calculate component of V normal to slice
    if torPlane:
        DefineScalarExpression("Vn", "Vz")
    if polPlane:
```

```
        thetaRad = deg2rad(theta)
        xHat = -sin(thetaRad)
        yHat = cos(thetaRad)
        DefineScalarExpression("Vn", "V[0] * " + str(xHat) + " + V[1] * " +
            str(yHat))
    AddPlot("Pseudocolor", "Vn")
    p = PseudocolorAttributes()
    p.minFlag = 1
    p.min = pMin
    p.maxFlag = 1
    p.max = pMax
    SetPlotOptions(p)
    SetDefaultPlotOptions(p) # ensures future instances use these settings
    plotName = GetPlotList().GetPlots(0).plotName
    l = GetAnnotationObject(plotName)
    l.xScale = 0.4
    l.yScale = 0.7
    l.fontHeight = 0.05
    l.numberFormat = "%+5.f"
    l.drawTitle = 0
    l.drawMinMax = 0
    AddPlot("Vector", "V")
    v = VectorAttributes()
    v.nVectors = nVectors
    v.useLegend = 0
    v.autoScale = 0
    v.scale = vScale
    v.colorByMag = 0
    v.glyphLocation = 1
    SetPlotOptions(v)
    if makeLineOut:
        # load start and end points for LineOut
        contents = sio.loadmat('/home/aaron/IDS/Geometry/coords' + \
            str(tor) + str(config))
        originArr = contents['origin']
        origin = tuple(map(tuple, originArr))
        # must convert numpy array to tuple
        origin = origin[0] # this is so stupid
        originArr = originArr[0] # this is so stupid
        ptsArr = contents['pts']
        pts = tuple(map(tuple, ptsArr))
        chan_range = contents['chan_range'] # can be left as an array
        # Calculate length of each chord
        dl = zeros((ptsArr.shape[0], 1)) # preallocate
        nSamples = zeros((1, ptsArr.shape[0])) # preallocate
        for (m, ptsx) in enumerate(ptsArr[:, 0]):
            dl[m, 0] = 1e3 * math.sqrt((ptsArr[m, 0] - originArr[0]) ** 2 +
                (ptsArr[m, 1] - originArr[1]) ** 2 +
                (ptsArr[m, 2] - originArr[2]) ** 2)
            nSamples[0, m] = dl[m, 0]
        nSamples = nSamples.astype(int64)
        DrawPlots()
        if lineOutV and not (lineOutn and lineOutT):
            lineMode = "V"
```

```
            nVars = 3
            Lineout(origin, pts[0], ("Vx", "Vy", "Vz"), nSamples[0,0])
        if lineOutV and lineOutn and not lineOutT:
            lineMode = "Vn"
            nVars = 4
            Lineout(origin, pts[0], ("Vx", "Vy", "Vz", "n"), nSamples[0,0])
        if lineOutV and lineOutT and not lineOutn:
            lineMode = "VT"
            nVars = 4
            Lineout(origin, pts[0], ("Vx", "Vy", "Vz", "T"), nSamples[0,0])
        if lineOutV and lineOutn and lineOutT:
            lineMode = "VnT"
            nVars = 5
            Lineout(origin, pts[0], ("Vx", "Vy", "Vz", "n", "T"), nSamples[0,0])
if plotIds1:
    idsTS1 = idsVectors(idsShot1, idsTP1)
if plotIds2:
    idsTS2 = idsVectors(idsShot2, idsTP2)
if not makeLineOut: # refine the plots for display
    AddOperator("Slice", 1) # "1" applies to all plots
    s = SliceAttributes()
    s.project2d = 1
    #s.originPoint = (0, 0, 1e-4)
    s.originIntercept = 1e-6
    if torPlane:
        s.normal = (0, 0, 1)
    if polPlane:
        s.theta = theta
        s.axisType = 4
    SetOperatorOptions(s, 0, 1)
    SetDefaultOperatorOptions(s) # ensures future instances use these settings
    va = View2DAttributes()
    va.viewportCoords = viewportCoords
    va.windowCoords = windowCoords
    SetView2D(va)
    AddOperator("Box", 1)
    b = BoxAttributes()
    b.amount = 1
    b.minx = boxAtt[0]
    b.maxx = boxAtt[1]
    b.miny = boxAtt[2]
    b.maxy = boxAtt[3]
    b.minz = boxAtt[4]
    b.maxz = boxAtt[5]
    SetOperatorOptions(b, 0, 1)
for m in range(0, len(simTimes)): # Time Advance
    if plotNim:
        SetActiveTimeSlider(nimTS)
        SetTimeSliderState(nimTimes[m])
    if plotTet:
        SetActiveTimeSlider(tetTS)
        SetTimeSliderState(tetTimes[m])
    if plotIds1:
        SetActiveTimeSlider(idsTS1)
```

```
        SetTimeSliderState(idsTimes[m])
    if plotIds2:
        SetActiveTimeSlider(idsTS2)
        SetTimeSliderState(idsTimes[m])
    if makeLineOut:
        SetActiveWindow(2)
        lineVals = []
        for (i, pt) in enumerate(pts):
            for k in range(nVars):
                SetActivePlots(k)
                t = GetOperatorOptions(0)
                t.point2 = pt
                t.numberOfSamplePoints = nSamples[0, i]
                SetOperatorOptions(t)
            SetActivePlots(0)
            vx = GetPlotInformation()["Curve"]
            SetActivePlots(1)
            vy = GetPlotInformation()["Curve"]
            SetActivePlots(2)
            vz = GetPlotInformation()["Curve"]
            if lineMode is "V":
                lineVals.append({'vx': vx, 'vy': vy, 'vz': vz})
            if lineMode is "Vn":
                SetActivePlots(3)
                n = GetPlotInformation()["Curve"]
                lineVals.append({'vx': vx, 'vy': vy, 'vz': vz, 'n': n})
            if lineMode is "VT":
                SetActivePlots(3)
                T = GetPlotInformation()["Curve"]
                lineVals.append({'vx': vx, 'vy': vy, 'vz': vz, 'T': T})
            if lineMode is "VnT":
                SetActivePlots(3)
                n = GetPlotInformation()["Curve"]
                SetActivePlots(4)
                T = GetPlotInformation()["Curve"]
                lineVals.append({'vx': vx, 'vy': vy, 'vz': vz, 'n': n, 'T': T})
        sio.savemat(simFile + 'LO' + simCode + str(tor) + '_' + str(config)
                + '_' + str(simTimes[m]).zfill(4) + '.mat',
                {'lineVals': lineVals, 'chan_range': chan_range})
    DrawPlots()
    time.sleep(5)
```

The following Matlab script converts the synthetic chord data into a movie equivalent to raw IDS data. The script and location are:

/Users/Aaron/IDS/Visit/Lineout_2_Movie.m

```
% This script takes the lineout files generated in VisIt/Python in the
% format 'LO <tet/nim> <tor=1/pol=0> _ <config> _ <time point> .mat', ie:
% LO1_3_63.mat and generates a false movie as if it had been created by the
% fast camera.
% The script loops through all times and both fibers, if applicable.
% 'lineVals{p, 1}.vx' contains the pth channel and velocity in the x
```

```
% direction. Channel numbers are matched to expermental channel numbers in
% 'chan_range'
% CONFIG REFERENCE:
% 1 toroidal, 71 degree port
% 2 toroidal, mohawk port in midplane
% 3 toroidal, axial port at 135 degrees
% 4 toroidal, mohawk port perp.
% 1 axial fiber @ toroidal angle 45
% 2 axial fiber @ toroidal angle 270
% 3 axial fiber @ toroidal angle 135
clear all; close all; clc;
addpath('~/IDS/Matlab/');
addAllThePaths;
%% INPUT SETTINGS
testing = 0; % make lots of plots along the way
saveTemp = 0;
% special setting for outputting weighted, chord averaged temperature.
refShot = 12949910; % real IDS shot including BD number added on
refLine = 3; % corresponds to 'line' setting in Batch_Correct and the tree,
%typically 3 for C III triplet
refLineDat = 3; % the index of the above line, ie: dat(#).temp,
% typically 3 for C III triplet
sim = 'nim'; % 'tet' or 'nim'
nimDataSet = 'fin_beta_ramp';
% folder containing everything related to a particular run
nimMatFile = 'post_process_output'; % for creating time base
tetDataSet = 'aaronData_141022';
% folder containing everything related to a particular run
tetMatFile = 'PSITET_14kHz_xMHD'; % for creating time base
tor = 1; % tor = 1, pol = 0 (refers to fiber)
config = 2; % config of the primary dataset
tor2 = []; % leave blank '[]' if only using one fiber
config2 = 1; % config of the secondary dataset
newShotNum = '6129499'; % new shot number,
% leading number indicates type of simulation, followed by real shot number
% 2 = NIMROD --- weighted by n^2 ---  NO temperature  --- 7x timeSum
% 3 = PSI-TET --- weighted by n^2 ---  NO temperature  --- 7x timeSum
% 4 = NIMROD --- weighted by n^2 ---  NO temperature  --- no timeSum
% 5 = PSI-TET --- weighted by n^2 --- with temperature --- 7x timeSum
% 6 = NIMROD --- weighted by n^2 --- with temperature --- 7x timeSum
% 7 = PSI-TET -- weighted by n^2 --- with temperature --- no timeSum
% 8 = NIMROD --- weighted by n^2 --- with temperature --- no timeSum
timeSum = [7]; % number of points to add together
% leave blank if none
chWeight = 1; % 0 = no chord weighting
% 1 = n^2
tempBroaden = 1; % 0 = do not use temperature
% 1 = broaden lines as a function of temperature
denFac = 1e-19; % factor to multiply by density to keep numbers down
dataFac = 0.07;
% scale factor for final data array to make values similar to IDS
%% LOAD IDS DATA, EXTRACT INFO
load(['dat' int2str(refShot) '.mat']);
% Extract settings from IDS 'mat' file
```

```
[~, yPix, xPix] = size(dat(1).raw); % size of original IDS movie
% Extract info for "binning" velocity data
% first, find velocity per pixel:
vPerPix = dat(1).param.c / dat(1).param.LineLam(refLine) * dat(1).param.PIX_SP;
%% LOAD LINEOUTS
if strcmp(sim, 'tet')
    simFolder = 'PsitetData';
    simDataSet = tetDataSet;
    simMatFile = tetMatFile;
elseif strcmp(sim, 'nim')
    simFolder = 'NimrodData';
    simDataSet = nimDataSet;
    simMatFile = nimMatFile;
end
cd(['/home/aaron/IDS/' simFolder '/' simDataSet '/LO/']);
matList = dir(['LO' sim int2str(tor) '_' int2str(config) '_*.mat']);
% all times for this fiber and configuration
%% Calculate Normal Vectors Along Chords
load(matList(1, 1).name); % loads 'lineVals' and 'chan_range'
simDat.peaks = chan_range;
% for feeding the channels to 'findIDSforVisIt' which it interprets as dat.peaks
ptShift = 0;
lengthBeyond = 0.1;
[~, vec1] = findIDSforVisIt(tor, config, simDat, ptShift, lengthBeyond);
% first fiber
%% Second Fiber
if ~isempty(tor2)
    matList2 = dir(['LO' sim int2str(tor2) '_' int2str(config2) '_*.mat']);
    % all times for other fiber and configuration
    matList = horzcat(matList, matList2);
    load(matList(1, 2).name); % loads 'lineVals' and 'chan_range'
    simDat.peaks = chan_range; % for feeding the channels to
    % 'findIDSforVisIt' which it interprets as dat.peaks
    [~, vec2] = findIDSforVisIt(tor2, config2, simDat, ptShift, lengthBeyond);
    % optional second fiber
end
%% PREPARE SETTINGS
nTimes = size(matList, 1);
data = zeros(nTimes, yPix, xPix); % initialize 'movie' data array
if saveTemp
    avgTemp = zeros(nTimes, length(chan_range));
end
for m = 1:nTimes
    for k = 1:size(matList, 2) % 1 or 2 if using both fibers
        load(matList(m, k).name); % loads 'lineVals' and 'chan_range'
        %% Calculate v along the chords
        if k == 1
            vec = vec1;
        else
            vec = vec2;
        end
        for p = 1:length(chan_range)
            % dot velocity vetor onto unit chord vector
            % !!! minus sign put in because of velocity convention. 'vec'
```

```
        % is a unit vector pointing toward the telescope, but the
        % prevailing theory is positive velocity points away.
        vdl(p).v = -[lineVals{1, p}.vx(2:2:end)', ...
            lineVals{1, p}.vy(2:2:end)', lineVals{1, p}.vz(2:2:end)']...
                * vec(p, :)';
    end
    if testing
        figure(eval([num2str(m) num2str(k) '1']))
        hold all;
        for p = 1:length(chan_range)
            plot(vdl(p).v);
        end
        title('v \cdot dl');
        xlabel('Index');
        ylabel('[m/s]');
    end
    %% Chord Weighting
    switch chWeight
        case 0 % no weighting
            for p = 1:length(chan_range)
                wVec(p).w = ones(size(vdl(p).v, 1), size(vdl(p).v, 2));
            end
        case 1 % n^2
            for p = 1:length(chan_range) % normalize density by a factor
                wVec(p).w = (denFac * lineVals{1, p}.n(2:2:end)).^2;
                % (density, n, scaled) squared
            end
    end
    for p = 1:length(chan_range)
        % find real channel index corresponding to 'chan_range(p)'
        dataInd = find(dat(1).peaks == chan_range(p));
        if ~isempty(dataInd) % I have calibration data for this channel
            %% Create 2D Grid Identical to Real Data
            x0 = round(dat(1).param.peaks(dataInd, 2));
            y0 = round(dat(1).param.Center(dataInd, refLineDat));
            xBound = x0 - dat(1).param.xWing : x0 + dat(1).param.xWing;
            yBound = y0 - dat(1).param.yWing : y0 + dat(1).param.yWing;
            [X, Y] = meshgrid(xBound, yBound);
            % Reshape Grid
            x(:, 1) = X(:);
            x(:, 2) = Y(:);
            % Evaluate the 2D temperature
            % Gaussian at every velocity point independently along
            % the chord. Sum all the Gaussians with optional
            % density weighting to get total distribution.
            % NB: this covers temperature AND velocity, so only
            % need to convolve this with inst. function.
            if tempBroaden
                root_T = sqrt(lineVals{1, p}.T(2:2:end)); % prepare sqrt(T)
                % All non-changing constants to convert sqrt(T) into
                % sig_y. sig_y = root_T * sig_coeff.
                sig_coeff = sqrt(dat(1).param.LineLam(refLine)^2 * ...
                        dat(1).param.kBoltz * 11605 / ...
                    (dat(1).param.c^2 * dat(1).param.IonMass(refLineDat)...
```

```
                        * dat(1).param.PIX_SP(dataInd)^2));
                    par_5 = sqrt((sig_coeff * root_T).^2 + ...
                        dat(1).param.peaks(dataInd, 5).^2); % [vector]
                else
                    par_5 = dat(1).param.peaks(dataInd, 5) * ...
                        ones(1, length(vdl(p).v)); % [vector]
                end
                % calculate "volume" for all points ahead of time
                % still need to do: par(1) = par_1(w) * sig_y
                par_1 = 2 * pi * dat(1).param.peaks(dataInd, 4) * ...
                    wVec(p).w .* par_5; % [vector]
                par_2 = dat(1).param.peaks(dataInd, 2); % x_0, [scalar]
                % par 3 (y_0) is every velocity point along the chord.
                % this converts velocity to pixels,
                % offsets so 0 matches experimental zero.
                par_3 = dat(1).param.Center(dataInd, refLineDat) + ...
                    (1 / vPerPix(dataInd)) * vdl(p).v; % [vector]
                par_4 = dat(1).param.peaks(dataInd, 4); % sig_x [scalar]
                par_6 = 0; % no offset
                z = zeros(size(x, 1), 1); % initialize empty array
                for q = 1:length(vdl(p).v)
                % loop over all velocity points for this chord/channel
                z = z + singletGauss2D([par_1(q), par_2, par_3(q), ...
                    par_4, par_5(q), par_6], x); % evaluate function
                end
                Z = reshape(z, size(X, 1), size(X, 2));
                if testing
                    figure(eval([num2str(m) num2str(k) '8']))
                    clf;
                    surf(X, Y, Z);
                    hold on;
                    view([0 90]);
                    colorbar;
                    shading interp;
                    title('Convolved Inst. Func., Vel. Dist. and Temp.');
                end
                if saveTemp
                    avgTemp(m, p) = sum(wVec(p).w .* ...
                        lineVals{1, p}.T(2:2:end)) / sum(wVec(p).w);
                end
            else
            end
            %% Add New Channel Data to 'data' array
            data(m, yBound, xBound) = Z;
        end % channel loop
    end % fiber loop
end % time loop
%% Create Time Base
cd(['/home/aaron/IDS/' simFolder '/' simDataSet '/DUMP/']);
t = []; % initialize
if strcmp(sim, 'tet')
    list = dir('out*.xmf');
    for n = 1:length(list)
        text = fileread(list(n).name);
```

```
        tmp = regexp(regexp(text,'Time Value="[0-9.]*"','match'),...
            '[0-9.]*','match');
        time = str2num(char(tmp{1}));
        t = horzcat(t, time);
    end
elseif strcmp(sim, 'nim')
    load(['/home/aaron/IDS/' simFolder '/' simDataSet '/' simMatFile]);
    for n = 1:length(nimsave.times)
        if exist(['dump_' sprintf('%05d', nimsave.times(n)) '_b.vtk'], 'file')
            t = [t, nimsave.time(n)];
        end
    end
end
t = 1e3 * t; % convert from s to ms
%% Time Averaging
if ~isempty(timeSum)
    n_time2 = floor(size(data, 1) / timeSum);
    data2 = zeros(n_time2, size(data, 2), size(data, 3));
    t2 = zeros(1, n_time2);
    for n = 1:n_time2
        data2(n, :, :) = sum(data((n-1) * timeSum + 1 : n * timeSum, :, :), 1);
        t2(n) = mean(t((n-1) * timeSum + 1 : n * timeSum));
    end
    data = data2;
    t = t2;
    dataFac = dataFac / timeSum; % rescale so data is still a reasonable value
end
%% Save Movie
% Throw in scale factor to make lines more closely match HIT-SI data pixel
% values
data = dataFac * data;
% Reflip Data so everything in consistent
data = data(:, end:-1:1, end:-1:1);
if ~testing
    cd(['/home/aaron/IDS/' simFolder '/' simDataSet]);
    save(['shot' newShotNum '.mat'], 'data');
    save(['t' newShotNum '.mat'], 't');
    if saveTemp
        save(['temp' newShotNum '.mat'], 'avgTemp');
    end
end
```

C.4 List of Other Codes

The following codes are used to generate data and figures in this work. Unless otherwise noted they reside on the 27-inch iMac in the HIT-SI3 post-doc office.

C.4.1 Chapter 3

- Figure 3.2 was generated with data supplied by Eagle Harbor Technologies, Inc. Jim Prager estimated an error upper-bound of 30%. The code is:

`/Users/Aaron/Sdrive/HIT_SI/Papers/RSI Helicon/Plotting_Langmuir.m`

- Figures 3.4 and 3.5 were generated with the codes:

  `/Users/Aaron/Sdrive/HIT_SI/Papers/RSI Helicon/Plotting_Main.m`
  `/Users/Aaron/Sdrive/HIT_SI/Papers/RSI Helicon/Plotting_Main_D.m`

- Figures 3.7 and 3.8 were generated with the code:

  `/Users/Aaron/Sdrive/Gas Puff Valve/Flow_Rate_Calc11.m`

- Figure 3.10 was generated with the code:

  `/Users/Aaron/Valves/Flow_Rate_HITSI3.m`

  and the raw data figure (A.1) was generated with:

  `/Users/Aaron/Valves/Fig_raw.m`

C.4.2 Chapter 4

- Figure 4.3 was generated by the calibration code.

- Figures 4.4, 4.5, and 4.6 were generated by:

  `/Users/Aaron/IDS/Analysis/BD/BD_Analysis_4.m`

C.4.3 Chapter 5

- Figures 5.1, 7.1, and 7.6 were generated with the code:

  `/Users/Aaron/Thesis/Matlab/Plot_HIT_general_traces.m`

- Figure 5.2 was generated from the code:

  `/Users/Aaron/IDS/Display/Play_Movie.m`

- Figures 5.3 and 5.4 were generated with the code:

  `/Users/Aaron/Thesis/Matlab/Error_Analysis2.m`

- Figures 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.20, 5.21, 5.22, and 5.23 are all generated by the code:

  `/Users/Aaron/IDS/Display/Multiplot.m`

  To use the code, uncomment the relevant section at the beginning where all input and plotting settings are preserved.

- Figures 5.17, 5.18, 5.24, and 5.25 were generated by the code:

  `/Users/Aaron/Thesis/Matlab/Plot_Data_Phase_Displace.m`

  To use the code, uncomment the relevant section at the beginning.

- Figure 5.19 was generated with the script:

  `U:\saracen\bvictor\Modified_ACH\bd_n1_inj_sub.m`

  Note that this code is on the "Alfven" server in the "users" drive and requires pre-object-oriented MDSplus, ie: Brian Victor's old computer.

- The Ohmic heating model is calculated by the script:

  `/Users/Aaron/Thesis/Matlab/Evolve_T.m`

C.4.4 Chapter 6

- Figure 6.1 was generated with the code:

  `/Users/Aaron/Thesis/Matlab/Disp_Currents_NIM_TET_IDS.m`

- Figures 6.2, 6.3, 6.4, 6.12, 6.13, 6.14, 6.15, 6.16, 6.19, 6.17, 6.18, 6.20, 6.22, 6.23, and 6.24 were all generated by:

  `/Users/Aaron/IDS/Display/Multiplot.m`

  To use the code, uncomment the relevant section at the beginning where all input and plotting settings are preserved.

- Figures 6.5 and 6.6 were generated with the code:

`/Users/Aaron/Thesis/Matlab/Plot_Data_Phase_Displace.m`

To use the code, uncomment the relevant section at the beginning.

- Figures 6.7, 6.10, and 6.25 were created with the VisIt GUI and the sessions are saved in:

  `/Users/Aaron/Thesis/VisIt/vel_nim_ids1727.session`
  `/Users/Aaron/Thesis/VisIt/vel_nim_j_1727.session`
  `/Users/Aaron/Thesis/VisIt/vel_nim_n.session`

- Figures 6.8 and 6.9 were created with the script:

  `/Users/Aaron/Thesis/Matlab/Lineout_Vel_Plot.m`

- Figure 6.11 was created with the script:

  `/Users/Aaron/Thesis/Matlab/Helicity_Canonical2.m`

- The "velocity-subtracted" data in figures 6.15, 6.16, 6.19, 6.17, 6.18, and 6.20 were generated with the script:

  `/Users/Aaron/IDS/Analysis/Subtract_Vels_from_Each_Other.m`

- Figure 6.21 was generated with the script:

  `/Users/Aaron/Thesis/Matlab/Temp_Profile.m`

C.4.5 Chapter 7

- Figures 7.2 and 7.7 were generated with the script:

  `/Users/Aaron/Thesis/Matlab/Plot_HF_Phase.m`

- Figure 7.3 was generated with the script:

  `/Users/Aaron/Thesis/Matlab/Raw_Wavelength_HF.m`

- Figures 7.4, 7.5, 7.8, 7.10, and 7.11 were generated with the script:

`/Users/Aaron/IDS/Display/Multiplot.m`

To use the code, uncomment the relevant section at the beginning where all input and plotting settings are preserved.

- Figures 7.18 and 7.19 were made with the code:

  `/Users/Aaron/Thesis/Matlab/Plot_Compare_Freqs.m`

- Figure 7.20 was made with the code:

  `/Users/Aaron/Thesis/Matlab/Plot_Compare_Freqs2.m`

- The calculations regarding β in section 7.5.4 were performed with the script:

  `/Users/Aaron/Thesis/Matlab/Beta_tau_E.m`