Stochastic Transport in PPCD Discharges

by

You Ming Yang

A dissertation submitted in partial fulfillment of the

requirements for the degree of

Master of Science

(Physics)

at the

University of Wisconsin – Madison

2010

Contents

1 Introduction **1**

1.1 The Reversed Field Pinch 1

2 Thomson Scattering **6**

2.1 Beamline 6

2.2 Laser Power Supply Upgrade 20

3 MST Plasma Characteristics **31**

3.1 Improved Confinement Discharges 31

3.2 Mode Activity and Electron Temperature Evolution 31

3.3 Equilibrium Reconstruction using MSTFit 42

4 Summary and suggestions for future work **57**

4.1 Summary 57

4.2 Suggestions for refinements and future work 59

A Alignment **66**

B FLPSCS Interface **72**

C Mode Analysis **81**

D MSTFit Misc. **86**

D.1 Transport Calculations 86

E Calculating χ_{st} **89**

F Island width calculation **91**

Chapter 1

Introduction

1.1 The Reversed Field Pinch

The Reversed Field Pinch[1] (RFP) is a type of plasma confinement device used in the pursuit of understanding the physics that govern magnetically confined plasmas. RFP's are characterized by a toroidal shape with an often circular cross section, and a low externally applied toroidal field. In an RFP, the equilibrium field configuration is one where the toroidal magnetic field is strongest at the core, falling off radially and eventually reversing, while the poloidal field increases radially (fig 1.1). The reversal of the toroidal field gives the RFP (Reversed Field Pinch) its name.

The field configuration of the RFP arises from the solution to Taylor's[2] equilibrium in a cylinder, which can be obtained by minimizing energy with global magnetic helicity held constant, this is approximately the case in conditions where the timescale for energy dissipation is much shorter than the dissipation of global magnetic helicity. Meeting this criteria results in the requirement that $\nabla \times B = \lambda B$. One may set λ to a constant and take the curl of this equation in a cylindrical basis, to arrive at solutions to the magnetic field to be Bessel functions. These solutions are similar to the fields

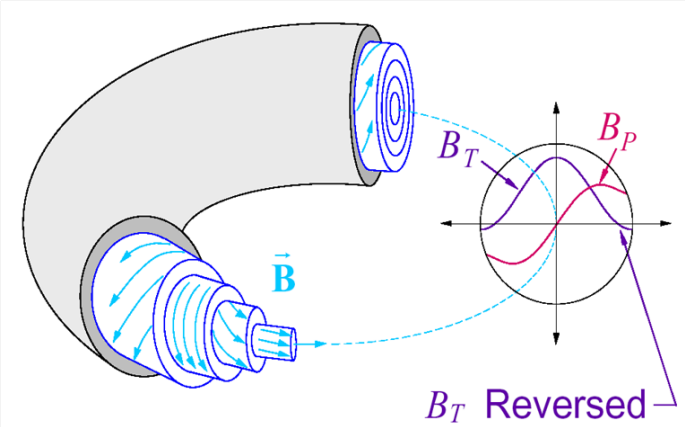

Figure 1.1: Illustration of RFP field configuration.

seen in an RFP. It is important to note that the plasma current is proportional to B with $\nabla \times B = \mu_0 J$, and as plasma pressure is related to the current and B by $J \times B = \nabla P$ we see that this requires plasmas in a Taylor equilibrium to have a flat pressure profile:

$$\nabla P = J \times B = \frac{\nabla \times B}{\mu_0} = \frac{\lambda}{\mu_0} B \times B = 0$$

As real RFP's do not have perfect cylindrical symmetry, have higher plasma pressure in the core than the edge, and are heated ohmically, they do not operate in the Taylor equilibrium configuration, instead constantly relaxing towards that state.

The Madison Symmetric Torus[3] (MST) is one such RFP, with a major radius of 1.5m, and a minor radius of 0.52m. MST plasma discharges are heated ohmically, with plasma current primarily being driven toroidally in the core. During MST operation, the plasma can be seen relaxing towards the Taylor equilibrium. The field configuration of an RFP such as MST can be described by a parameter $q(r) = \frac{rB_\phi}{RB_\theta}$. At the radial locations where the q becomes rational or $q = \frac{m}{n}$, unstable modes reside and can grow, the dominant modes being m=1 modes. These modes are driven by the

gradient in the parallel current, and form islands centered on the rational surfaces.

In a standard discharge, the overlap of these islands result in a stochastic equilibrium field throughout the entire plasma, resulting in very high energy transport. The Rechester-Rosenbluth[4] model of stochastic transport describes this stochastic energy diffusion (for a collsionless plasma) as a double-diffusion process by which particles, tied to the magnetic field lines diffusing radially proportional to the radial component of stochastic field $D_m \sim \frac{b_r^2}{B_z^2}$, collide and transfer energy at a rate proportional on the electrons' parallel thermal velocity:

$$\chi_{e,stoch} \sim D_m \frac{\lambda_{mfp}}{\tau_{collision}} = D_m v_{Te}$$

Stochastic energy transport in MST has been successfully compared this model by Biewer et al.[5].

One method of improving energy confinement on MST is to reduce the stochasticity of the plasma, thereby reducing the stochastically driven transport. This is done by reducing the plasma's own drive to reduce the gradient in the parallel current profile. To do this, we apply an external current profile control method called Pulsed Poloidal Current Drive[6] (PPCD). PPCD inductively drives poloidal current in the edge (where it is the parallel current) of the plasma. This modification of the current profile offers a dramatic improvement in plasma properties[7], including the reduction of island widths and a correspondingly lower level of stochasticity, the elimination of sawtooth crashes, healed flux surfaces, and much higher core electron temperatures as a result of a dramatic improvement in electron energy confinement. While PPCD discharges show much better performance on the metrics of energy confinement and peak

core plasma temperature, they also exhibit high shot-to-shot variability between discharges that have very similar starting conditions. This shot-to-shot variability offers an opportunity to examine the impact of stochasticity on varying confinement quality. Understanding, and predicting the causes of this disparate behavior will also benefit not only the optimization of PPCD, but more advanced current drive techniques in the future.

A diagnostic that has been crucial in helping illuminate the behavior of RFP plasmas such as those in MST is the Thomson Scattering[8] diagnostic. Thomson scattering (TS) is the process by which charged particles are accelerated by an incident electromagnetic field, and re-emit radiation due to that acceleration, scattering the incident wave. The frequency of the incident wave is shifted by the velocity of the charged particle and the direction of scatter. By comparing the spectrum of scattered photons to what one would expect with a Maxwellian distribution at a given temperature, the temperature of the scattering particles can be gleaned. In low field magnetic plasma confinement devices such as those in the Madison Symmetric Torus (MST), passive diagnostics such as Electron Cyclotron Emission (ECE) do not work as the plasma is overdense, and diagnostics such as Soft X-Ray (SXR) experience contamination from aluminum lines[9]. Therefore, Thomson scattering is an invaluable tool for accurately determining the electron temperature of the plasma in these devices. In MST, the incident electromagnetic wave is a monochromatic Laser beam at 1064nm that travels 15m through an enclosed beamline, ultimately passing through the plasma volume (see Chapter 2) and scattering off of the electrons. The scattered light from 21 radial positions is collected and from this data a temperature profile of the plasma

can be created.

This thesis will describe the implementation of various upgrades, to both the beamline and laser system, that have streamlined the day-to-day operation of the system. Additionally, a pulse-burst upgrade to the laser system has greatly increased the amount of temperature data that can be collected during each MST discharge or shot, allowing for resolution of fast plasma dynamics for fluctuation measurements (Stephens[10]), and the time-resolved evolution of the plasma electron temperature profile. The latter capability has allowed for unprecedented single-shot analysis of discharges that have a high shot-to-shot variability, such as Pulsed Parallel Current Drive (PPCD) discharges. Single-shot analysis of PPCD performance suggests a possible correlation between performance and m=1, n=8-15 magnetic fluctuations, near the reversal surface. An investigation of the effect of stochastic transport in this improved confinement regime is presented.

Chapter 2

Thomson Scattering

2.1 Beamline

The Thomson scattering diagnostic on MST currently uses two commercial Spectron[11] (SL858) lasers located in a laser room across a public hallway from the MST experiment. After exiting the laser head, the laser beam is directed underneath and across the hallway which separates Thomson scattering from MST, over and above the machine, and finally down vertically through the machine vessel, as can be seen in figure 2.1 (Reusch et al.[12]). A seven-element lens is used to focus scattered laser light onto 21 optical fiber bundles. This scattered light travels to filter polychromators[13], containing filters that select out specific wavelength regions to pass onto avalanche photodiode detectors.

The incident laser beam undergoes 5 turns before scattering off of the plasma, and the overall length of the beam path is over 15m. With this setup the beam alignment can be observed to drift throughout the course of the day. This drift is more pronounced during summers, and has been attributed to thermal expansion and contraction the building undergoes in various parts of the 15m beamline. If left

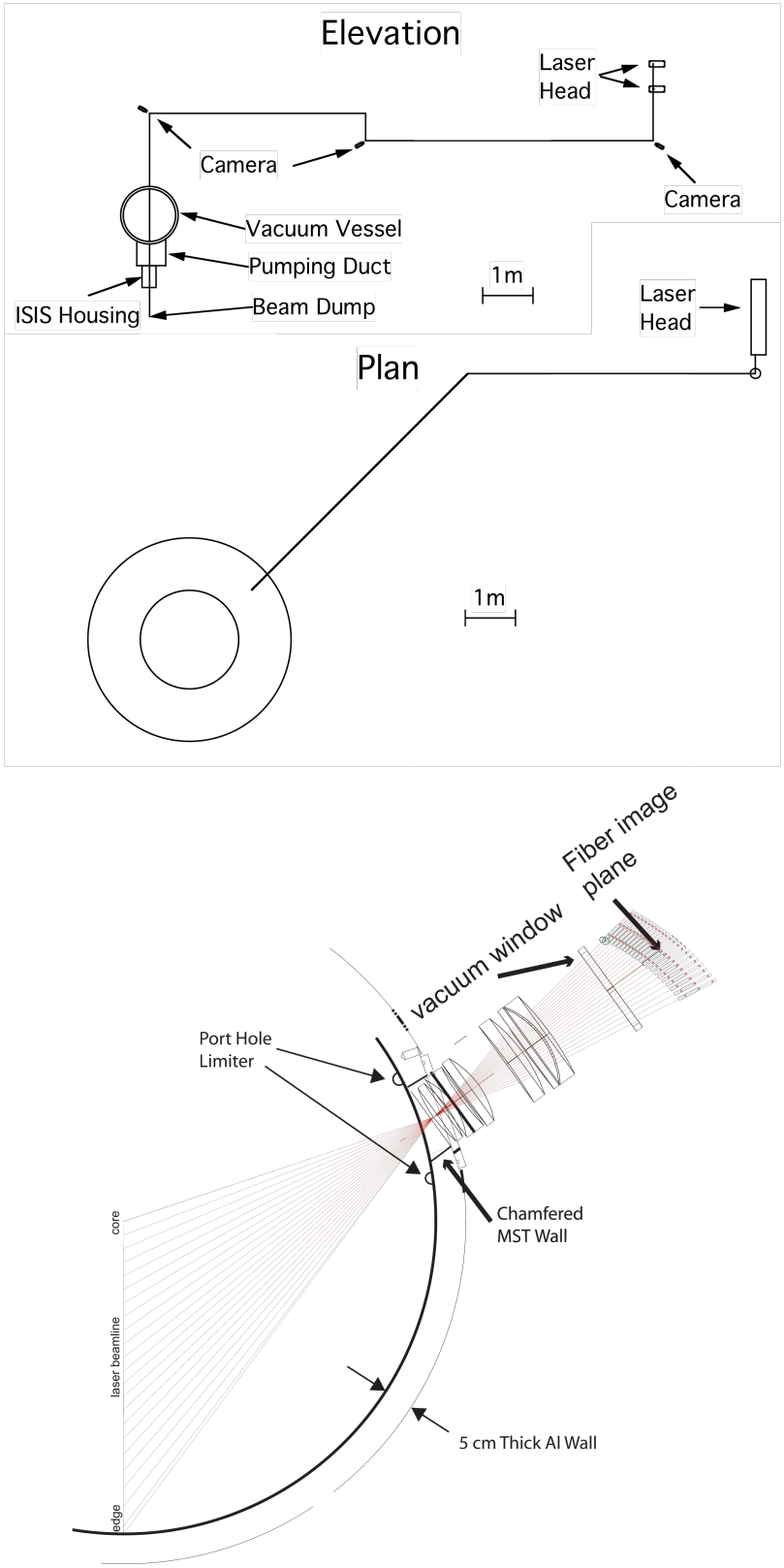

Figure 2.1: TS Beam Path from clean room to MST (above), and TS collection optics (below).

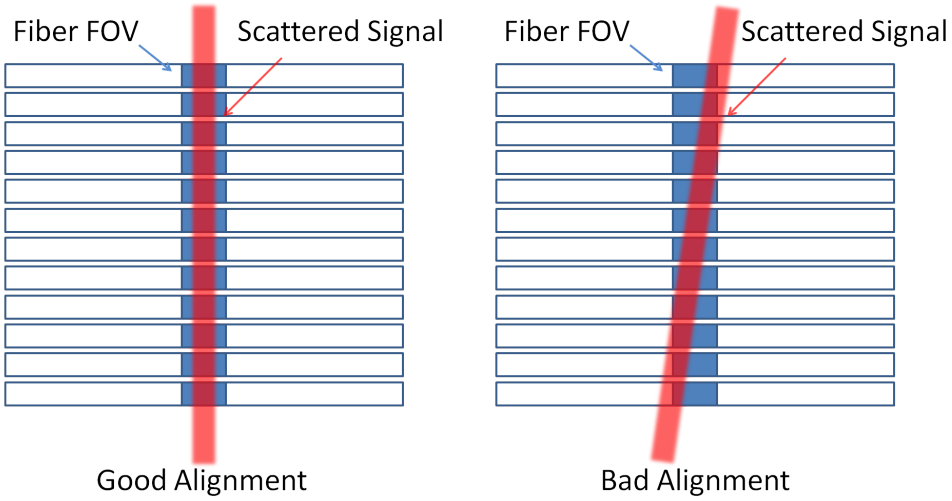

Figure 2.2: Illustrations of good alignment and poorly aligned scattered laser signal relative to optical fiber bundle.

uncorrected, the beam can drift out of the field of view of the collecting optics (2.2), resulting in reduced scattered light and poorer quality of data.

To counteract this, the beamline is equipped with remotely actuated turning mirror mounts and CCD cameras, mounted as shown in figure 2.3. The beam spots of the lasers incident on the turning mirrors are imaged by the CCD cameras.

During day-to-day operation, a Thomson Scattering operator would monitor the alignment and beam position. When the beam deviated from the center of of the turning mirror, a command could be sent to a turning mirror earlier in the beamline to adjust the position of the beam spot. Hardware limitations of the camera imaging equipment allow only one beam spot on one turning mirror can be imaged per laser fire. Additionally, the variations in orientation of mirror-mounts, motor control hysteresis, and camera positioning setups between turning mirror stages meant that the same commands sent to different turning mirror stages could have very different impacts on

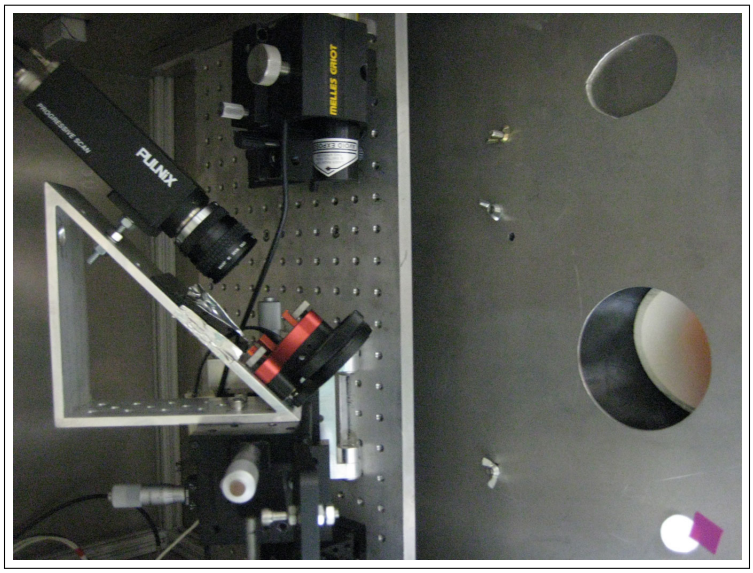

Figure 2.3: Mirror mount and Pulnix Camera in a beamline.

how they changed the beam alignment.

Due to these difficulties, ensuring proper alignment of the beamline required constant supervision by an experienced Thomson scattering operator. At the start of the day, the operator would find the ideal beam position for optimum collection of scattered photons. As the day progressed, the operator would need to correlate any deviation in beam alignment with a reduction in scattered photons, suggesting a misalignment. Then the user would input the appropriate commands to the turning mirrors to readjust the beam positions to what was optimal. Building on this system, an automation upgrade to the alignment system was implemented.

2.1.1 Calibration

In principle, the positions of the beam spots in the stage above MST (stage m in fig 1.1) and in the beam dump (d) determines the path that the beam takes through the machine. To adjust the position of the beam spot above the the machine requires adjusting the tilt of the turning mirror on the East Wall (w); adjusting turning mirror at stage w will also affect the beam's subsequent path (m) and final position in the beam dump (d), imaged by the beam dump camera. Similarly, adjusting the tilt of turning mirror m just changes the final position of the spot d. The tilt of the turning mirrors are adjusted via orthogonal directions subsequently referred to as directions A_w and C_w for the wall, and A_m and C_m for above the machine. Due to their varying orientation, adjusting just one stage in one direction (for example A_m, will have an impact on both the X and Y positions of the beam center in subsequent stages. We can write X_m and Y_m, X_d and Y_d as the coordinates (in pixels) of the beam center as recorded by the Pulnix cameras.

Using coefficients a, c to convert from units of translation to change in pixel position, and using the representation a_{xwm} to represent the change in X as a result of adjusting the turning mirror on the wall, on the stage above the machine etc., we can write the relationship between adjustments in A's and C's with X's and Y's.

For the changes in X,Y in stage above MST due to a change in the turning

mirror located at the East Wall:

$$\begin{aligned} X_m &= a_{xwm}A_w + c_{xwm}C_w \\ Y_m &= a_{ywm}A_w + c_{ywm}C_w \end{aligned} \tag{2.1}$$

The beam dump can be affected by two turning mirror stages:

$$\begin{aligned} X_d &= a_{xwd}A_w + c_{xwd}C_w + a_{xmd}A_m + c_{xmd}C_m \\ Y_d &= a_{ywd}A_w + c_{ywd}C_w + a_{ymd}A_m + c_{ymd}C_m \end{aligned}$$

Here, it is also useful to note that for the beam dump, X_d and Y_d are modified by changes in both the wall turning mirror stage, as well as the stage above MST. However, changes from the wall stage are generally unwanted, and instead corrected for by the above MST stage.

Calibration was performed by taking repeated measurements of changes in X and Y on the stages above MST and in the beam dump by adjusting A and C at the wall and above MST, then the coefficients were computed and hardcoded into the auto-alignment software. It is also important to note that there is often a fair amount of hysteresis in the turning mirror motors, such that for example, moving 1000 units in A and then -1000 units in A will not result in returning to the original position. As such, the above a's and c's must be solved for twice, once in positive moves in A and C and again for the opposite direction.

2.1.2 Alignment Algorithm

Much of the calibration and implementation of the automated alignment procedure relies on the use of the National Instruments Image Acquisition (NI-IMAQ) development package, of which the following functions were highly utilized for this purpose: a function to calculate the geometric center of intensity of a black and white image in a selection region of interest, the ability to store and retrieve custom regions of interest and data. The implementation for the automated alignment setup was then broken down into two components:

Once ideal alignment had been achieved via manually adjusting the turning mirrors, a "create-reference" program would be executed to record the beam positions on each turning mirror for each laser. A region of interest (ROI) for each turning mirror is selected to calculate the beam center (used to crop out bright scattered artifacts from mirror mounts, reflections, etc.), then the ROI and center of the beam spot is saved to a reference file.

An "autosteering" program is then used for day-to-day operation. After a data taking discharge, the camera is triggered with individual laser pulses, recording the location of the beam on each of the turning mirrors. The beam center is then calculated using the same ROIs as those stored in the reference files, and the new beam center compared to the stored reference position. If the pixel deviation (X's and Y's) is higher than 10 pixels total on any stage, the turning mirrors are adjusted via the calibration calculated above. The end result is an automated beam alignment program that requires much less user supervision than the previous iteration.

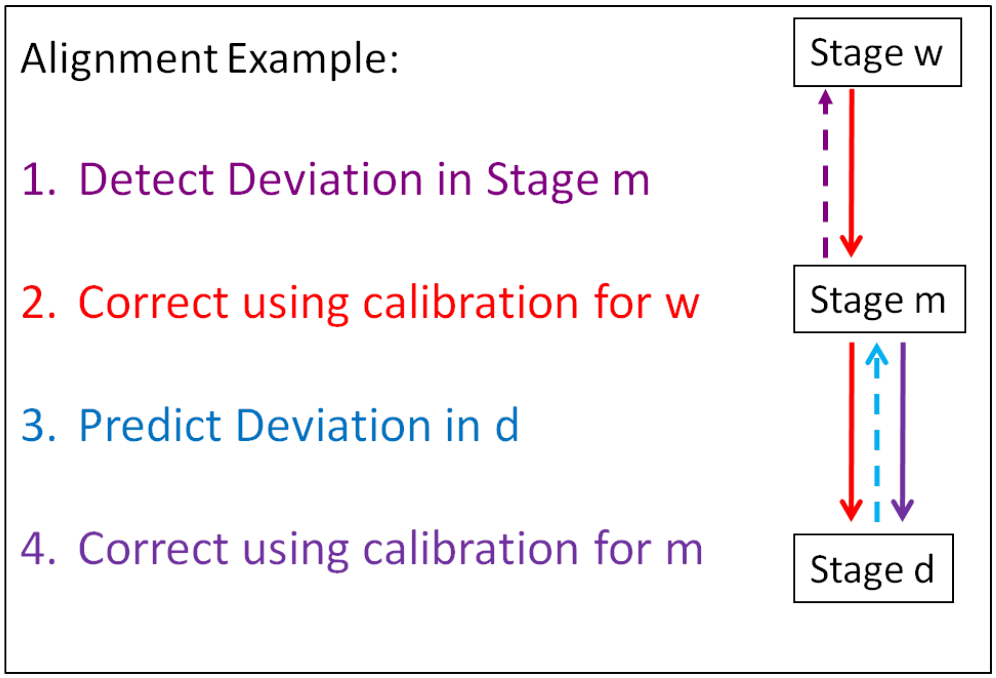

Figure 2.4: An example of using the calibrations to fix mis-alignment in the stage above MST.

2.1.3 Alignment Implementation

This section refers to the implementation of the alignment algorithm within Labview. Small Labview icons and their features will be described, and a more complete back-panel can be seen in the appendix (A).

The initial alignment of the TS system is an involved process discussed elsewhere[14], and will only be briefly outlined here. The mirrors in the Spectron beamline are replaced with dichroic mirrors that allow propagation of visible HeNe diode lasers–to which the 1064 beam is aligned–down the beamline, through MST, and into the beam dump. The bundle of collection fibers are aligned to a beam probe that is inserted into MST from the beam dump port. In general, with the incident laser beam

exiting through the same port that the beam probe is inserted through, a rough alignment can be achieved which results in at least some scattered laser light to be within the FOV of the fibers. Additional manual adjustments of the turning mirrors allows the user to maximize the number of collected photons. This last step is currently unavoidably time consuming, as it requires a plasma for the incident laser beam to scatter off of, allowing only one or two adjustments total every few minutes.

Once optimum alignment has been achieved, the next step is to create calibration files used in the automated alignment program. A Labview VI was created for such a purpose, and has a front panel which allows the user to select a specific turning mirror stage and laser to create references for. The full FOV of the camera captures the scattered light of not only the beam spot imaged on the turning mirror, but also additional artifacts due to scattered light, such as the mirror mount and sometimes reflections of the camera itself. To most effectively utilize NI-IMAQ's built-in intensity-averaging subVI, the camera image is displayed on the front panel, and the user may select an outline within the image to be cropped to a specific "region of interest (ROI)" for processing (2.5).

In the back panel, upon execution the program waits on a "camera acquire" trigger which acquires what a specific camera stage sees within a 16ms time window. This trigger is sent to the camera right before a laser of interest is fired, allowing for the selection of one laser beam on one camera stage. This acquired image is displayed on the front panel where the user selects an outline to use as an ROI. This ROI serves as an image mask, cropping out any of the image beyond the selected region. The remaining region is then input into the NI-IMAQ Centroid subVI, which does the

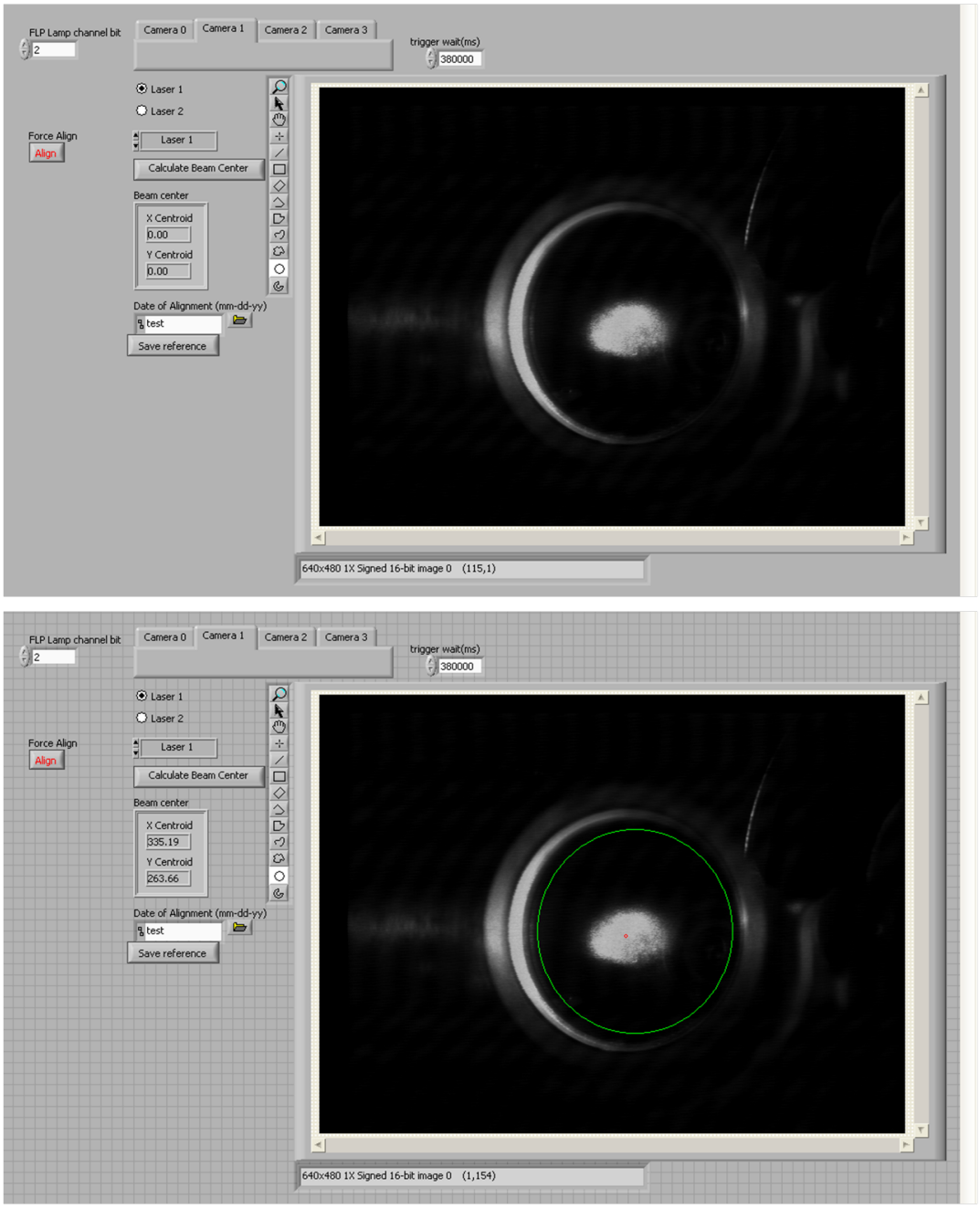

Figure 2.5: Create reference VI, before and after manual selection of ROI to crop.

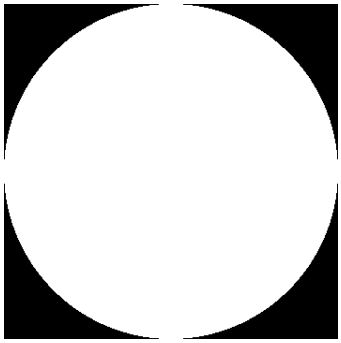

Figure 2.6: Reference image mask file used for auto-alignment.

aforementioned intensity averaging, giving the pixel locations of the intensity centroid which calculates the center of the intensity in the X and Y directions as:

$$X = \frac{\Sigma x * I(x)}{\Sigma I(x)} \tag{2.2}$$

$$\tag{2.3}$$

$$Y = \frac{\Sigma y * I(y)}{\Sigma I(y)} \tag{2.4}$$

Where x and y represent a pixel's coordinates and $I(x, y)$ is the numerical value of the intensity for a pixel at that given x or y position, and the sum is taken over all pixels. For the eight bit pixels used in the alignment code, the intensity can range from 0 to 255, where 0 is a black pixel with no energy and 255 is a white pixel with the most energy.

As the orientation of the cameras and mirrors vary between stages, a different image mask is created and used for each mirror stage.

The output of this routine is stored as a proprietary IMAQ data file format, which simply consists of the image mask (stored as a .png) and X-Y pixel coordinate

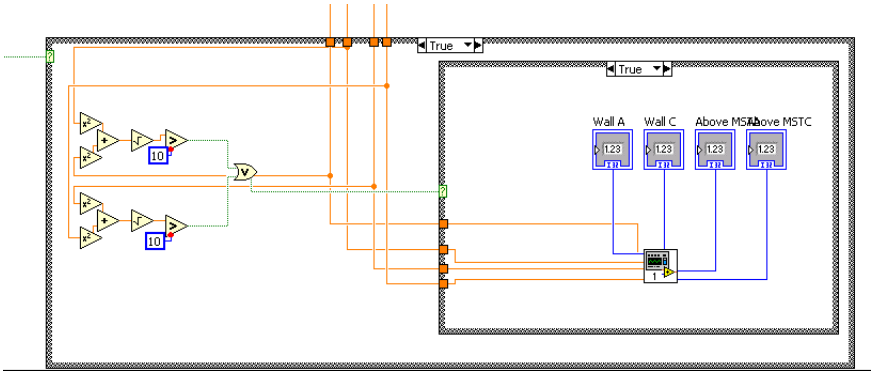

Figure 2.7: Reference call to motor-controller subVI, with X-Y pixel deviations in stages m and d taken as inputs.

of the laser beam (embedded as ASCII data within the .png file 2.6). As each file contains separate X-Y pixel information, one per each laser, there are two files stored per laser stage containing the same image mask and different center coordinates. Very often, the relative alignment of the components in the beamline–such as the orientation of the cameras relative to the turning mirrors and the fiber mount and lens setup–are undisturbed for long periods of time, and these calibration files may be used for extended periods of time without a need for recalibration.

Once the calibration files have been created, they may be used by the runtime VI for day-to-day operation. The runtime VI begins by loading the six calibration files, and proceeds to wait on six consecutive camera triggers in a specific laser firing sequence. After each camera trigger, the imaged laser pulse is cropped by its respective image mask, and the new beam center is calculated. The beam center for both lasers is calculated for each stage, and their overall deviation from their respective reference points is calculated. If this deviation is beyond 10 pixels, the XY deviations are sent to the calibrated motor-controller subVI which corrects for alignment (2.7).

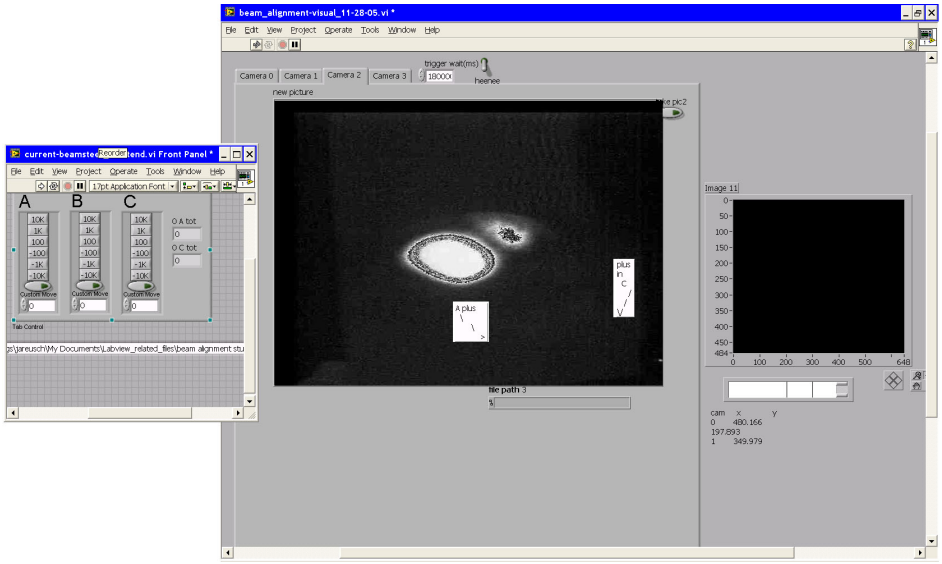

Figure 2.8: Previous beam-line alignment interface, consisting of two separate VI's. One VI images the beam spot (right), the other offers manual controls for the user to correct for beam mis-alignment (left).

In this subVI, the previously calculated movement-to-pixel calibrations have been hardcoded. The X-Y pixel deviation inputs to this VI are multiplied by their appropriate calibration factor, and sent to the motor controller for automated alignment. A copy of the back-panel along with calibration values can be found in the Appendix (A).

The end result is the automated (2.9) optimization of a manual laser alignment correction method that eliminates the need for an experienced TS operator to constantly observe and manually correct the beam alignment (2.8), and the new capability to record optimum alignment positions and automatically adjust to them on subsequent run-days.

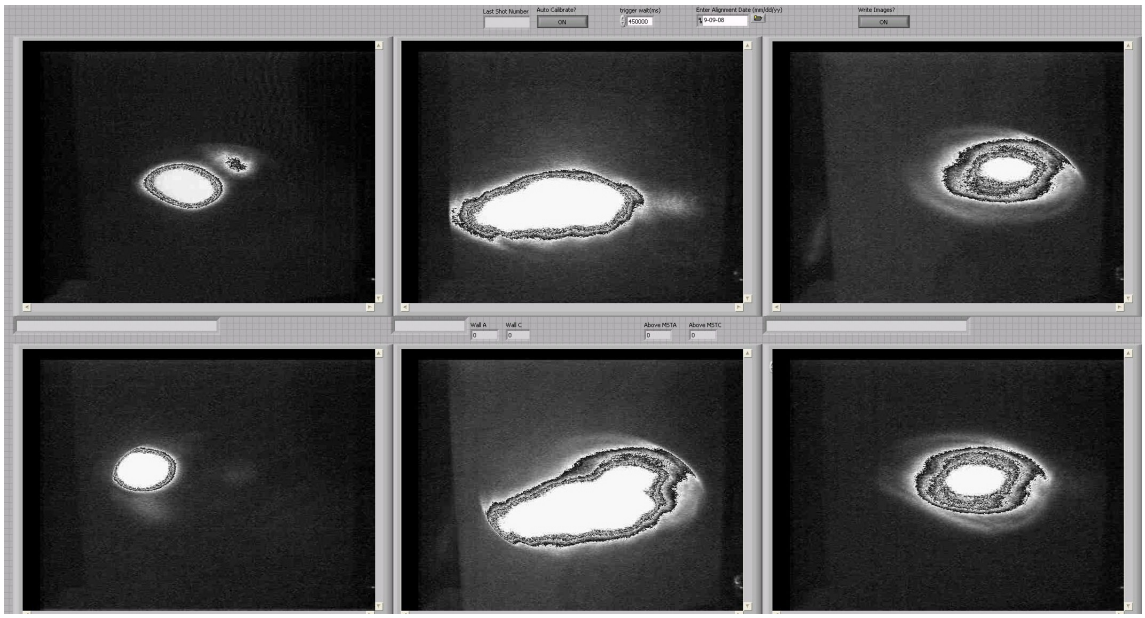

Figure 2.9: Automated beam-line alignment program with beam spot images for both lasers for stages w, m, and d.

2.2 Laser Power Supply Upgrade

As previously mentioned, a major upgrade to the Spectron laser Thomson scattering system has been the replacement of the old commercial power supplies with one made by UW-Madison's Physical Sciences Lab, allowing for improved operational versatility of the Spectron lasers over the previous commercial incarnation. The new power supply features fast IGBT switched capacitor banks that can be fired at various pulse width and duration. This system is composed of a capacitor chassis, pulse drive chassis, capacitor control box, and master control box. The heart of the control system is C/Assembly code programmed onto Rabbit microprocessor boards. These boards perform various functions such as controlling IGBT pulse duration and timing, setting capacitor charge voltage and simmer toggle, as well as performing various status and error checks. This power supply was used in conjunction with two Jorway 221-A timing modules, which are capable of generating a 16 second long sequence of pulses with 1μs resolution. The fruits of the labor that went into the Spectron TS system is the capability to operate at a variety of "pulse-burst" modes[15] where multiple laser pulses can be achieved with one flashlamp pulse, enabling laser pulse repetition rates from 1kHz-12.5kHz per laser. The two Spectron lasers were also operated in an interleaved format on a regular basis, resulting in 2kHz-25kHz data acquisition rates. Achieving these modes of operation involves direct control over both the flashlamp as well as Pockels cell timing.

2.2.1 Python Interface

The control and timing system of the power supply is referred to as the Flash Lamp Power Supply Control System (FLPSCS) and interfaces with a PC via simple serial communication for setting parameters such as pulse width, pulse delay, flashlamp voltage, etc. for each channel, as well as querying various statuses like capacitor charge and simmer status. While robust, the serial communication scheme requires the user to manually enter numerous commands and status checks per each channel. For the Spectron lasers, four channels were used for each laser (8 channels total), and for the Fast Thomson[16], all 19 channels of the power supply would ultimately be used.

The FLPSCS was programmed to take properly formatted serial commands, and send a response immediately after. Voltage programming commands are received with a confirmation, a query for previous settings returns the set value, and status or error checks give the corresponding state of the system. Aside from the solicited responses, there are additional unsolicited responses the FLPSCS is capable of sending. Between discharges the system performs a self-diagnostic error check for a potential Gate Driver Board fault, and will send an unsolicited error message if such a fault exists. The same occurs if the emergency stop is triggered. Additionally, if the system is in E-stop mode, or Run mode, it will not send any responses to the user.

This motivated the creation of a graphical user interface that could automate many of the aforementioned tasks, develop the best implementation to interface with the control system, as well as reduce the workload required of a TS operator when programming the power supply for the desired mode of operation.

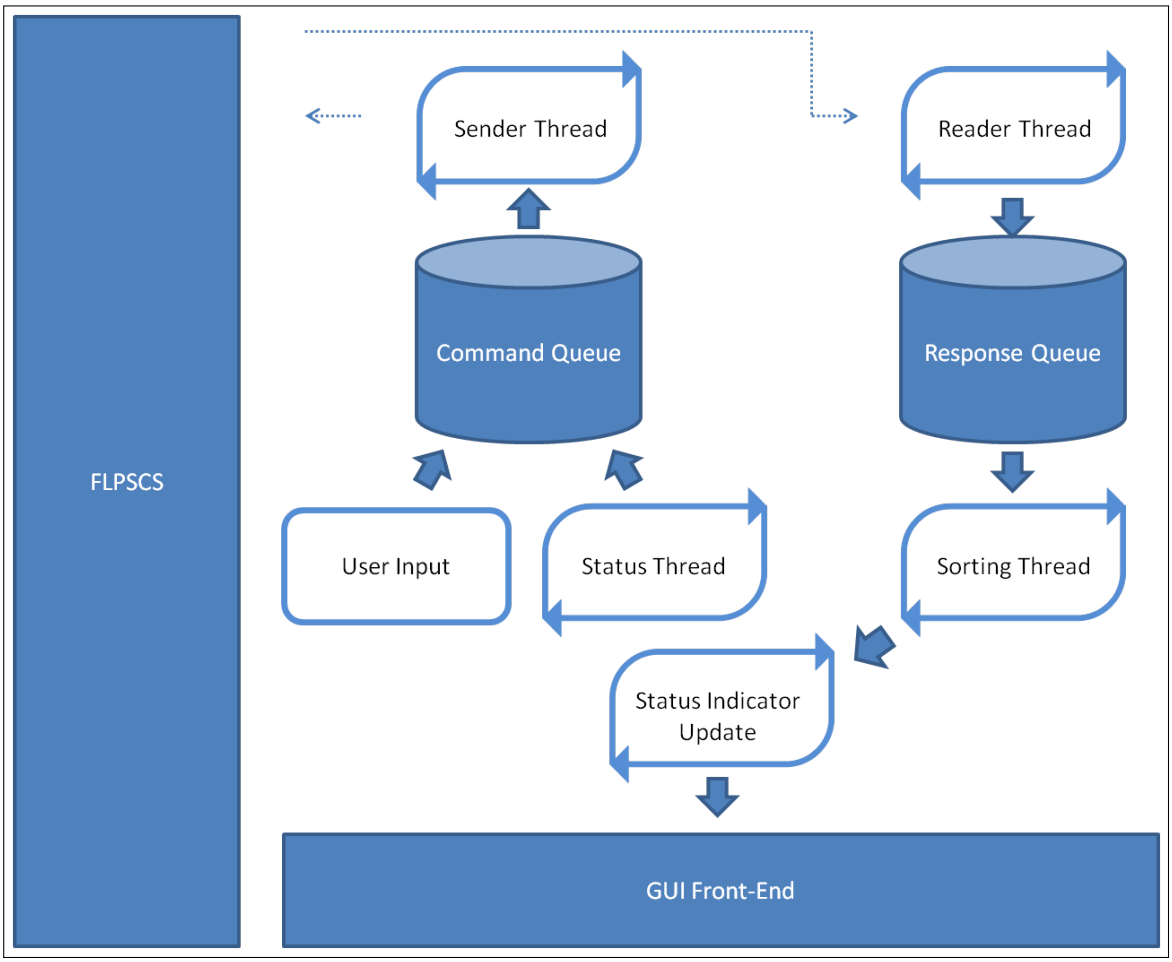

Figure 2.10: Flow chart for FLPSCS Control Interface.

For ease of use, testing and modification, a GUI interface in Python was created. The back-end was programmed in Python, and the front end utilized PyQT, a Python wrapper around the QT graphics package. Formatting of the serial commands and graphical implementation aside, the main efforts of the power supply interface went into the method of handling the serial I/O.

Taking advantage of Python's multi-threading capabilities, the I/O communication was broken down into four main parallel threads(2.11): a serial-send thread, and a serial-read thread, a sorting thread, and a status checking thread. The sending thread would monitor an internal command queue, and send any commands from the command queue to the FLPSCS, with a built in delay that allowed the FLPSCS to respond with its expected solicited response. The receiver thread would monitor inputs on serial port, and send all responses to a response queue that is then processed by a sorting thread, which identifies the response and updates the appropriate status indicator. The status-checking thread periodically fills the command queue with queries for charge voltage, simmer status, and other various updates.

When the status thread is operating, there is a "thread lock" on the status thread that ensures it has priority for populating the command queue. When a user programs the system for operation, the program thread requests the lock, waiting for the status check to finish first if necessary, and then proceeds to fill the command queue with programming commands. A similar lock is used for the serial-send thread itself to disable the sending of commands when the system is in Run mode.

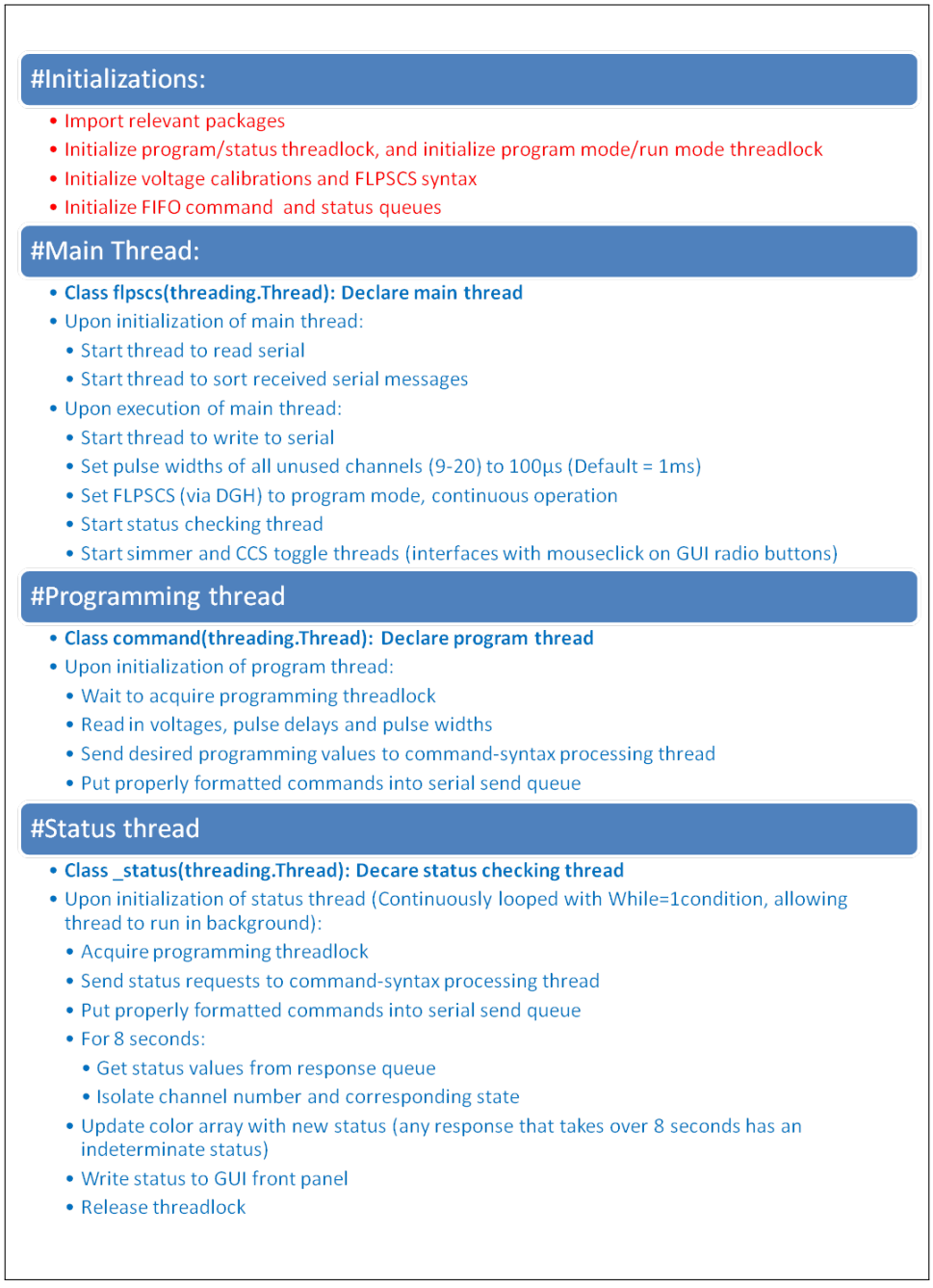

Figure 2.11: Algorithm for FLPSCS Control page 1.

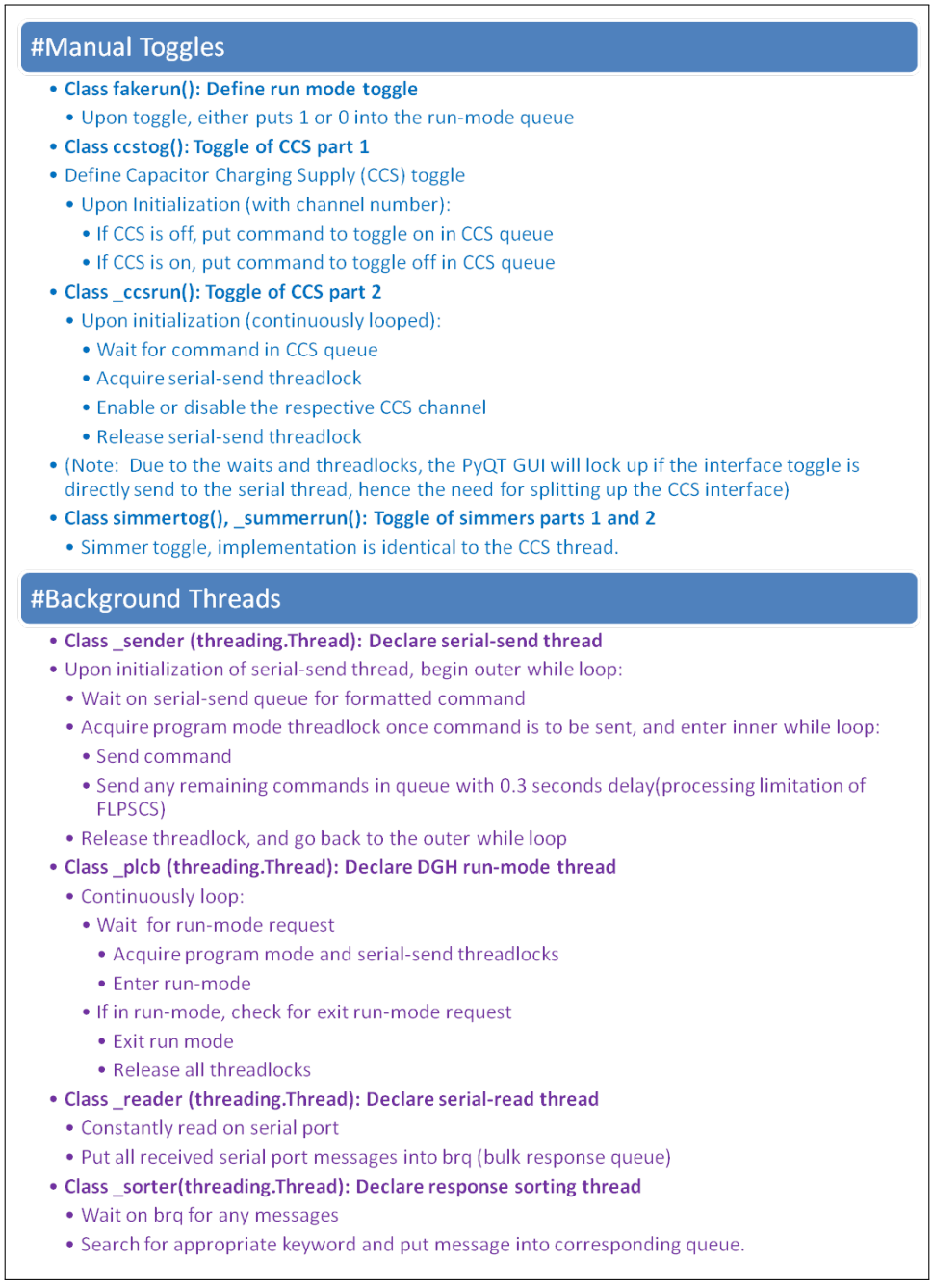

Figure 2.12: Algorithm for FLPSCS Control page 2.

2.2.2 Timing Algorithm and Implementation

The design of the upgraded power supply system already allows for repeated triggering at $\sim 1ms$ intervals, allowing for much more potential than the commercial power supplies. However, many different components of the TS system had to be coordinated together for the Spectron lasers to fully take advantage of the new power supplies. While the timing could have been implemented in a variety of ways –and is indeed in the process of being upgraded to a more compact system– a very functionally elegant solution was implemented with the hardware available at the time.

The main focus of proper timing implementation is the capability of reliably and consistently generating 2J laser pulses, while at the same time minimizing the stress on the flashlamps. As previously mentioned, the timing scheme for the Fast Spectron is controlled by two Jorway J221-A modules, and the FLPSCS Master Control chassis. An MST trigger to begin TS data collection is sent to one of the two Jorway modules. This module is used to control the "macro" ($\sim ms$) timing and determines the frequency of the flashlamp burst pulses. Once the macro Jorway triggers the master control chassis of the power supply, there is a $\sim 10\mu s$ jitter for the drive system to respond before initiating the flashlamp pulses. Once the flashlamp bursts begin, a sync pulse, with a $\sim 1\mu s$ jitter relative to the flashlamps, is sent to the the second "fast" J221 module, which then with triggers the Q-switch and digitizers on order $\sim \mu s$ timing, generating a laser pulse and digitizing any data from the scattered signal. This method allows the lasers to be Q-switched within $\sim 1\mu s$ of their optimal charge time and digitize at the proper time with respect to the flashlamp bursts, minimizing the

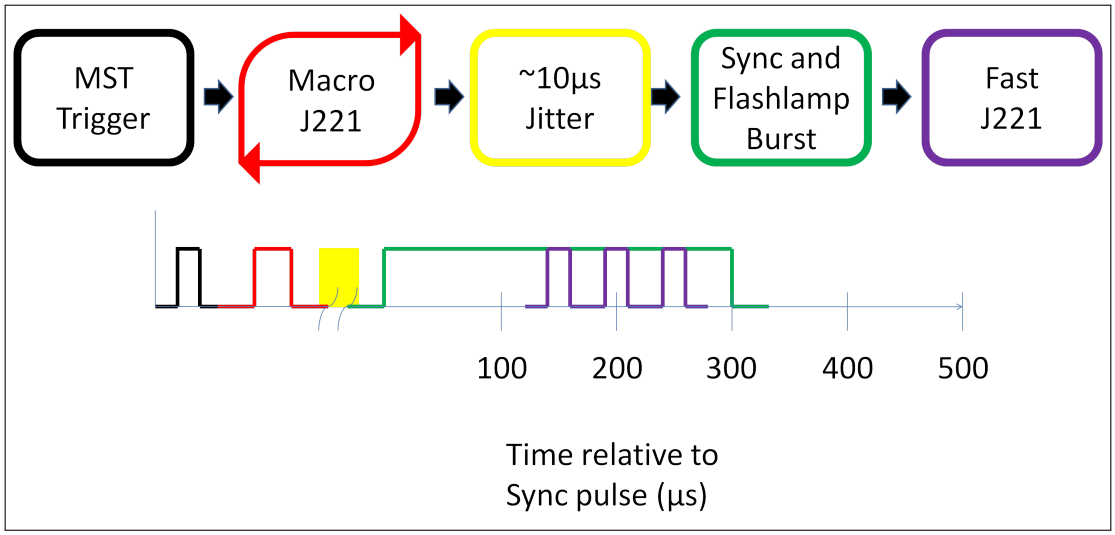

Figure 2.13: Flow chart for FLP timing of a pulse-burst sequence.

time that the flashlamps are on, while ensuring optimal pumping of the rods before lasing. This implementation allowed for the capability of taking data at rates ranging from 2kHz, to 25kHz in "pulse-burst" mode (fig 2.13).

The Jorway J221-A modules function by receiving two input arrays, one for time (μs) after an input trigger, and one to turn any number of channels (referenced via bit-number+1) high-low corresponding to that time. For example, two arrays consisting of: (0,5,10) and (6,2,0) would have the Jorway turn on channels 3 and 2 on at t=0, and then leave only Ch. 2 on after t=5μs, then turning everything off at t=10μs. The Jorway modules are interfaced with GPIB via low-level Labview drivers.

2.2.3 Timing Schemes

By using the basic pulse-burst hardware and timing scheme above, the simple modification of certain parameters such as flashlamp pulse widths and delays, and fast J221 triggering, allows for a variety of TS operation modes.

Single Pulse Operation(fig 2.14): The most basic and commonly used mode of operation is the interleaved mode of two lasers firing in Single-Pulse mode. In this mode after the MST trigger, the macro J221 sends master triggers spaced $1ms$ apart to the FLPSCS. For each master trigger received by the FLPSCS –after a $10\mu s$ jitter– at t=0 the four channels driving the first laser's flashlamps fire for a $150\mu s$ duration pulse. At t=$140\mu s$ (to allow for an optimum population inversion in the rods) the fast J221 triggers the Q-switch for the first laser and digitizers. After a $500\mu s$ delay, the four channels driving the second laser's flashlamps fire for a $150\mu s$ duration pulse. $140\mu s$ into the second pulse, at t=$640\mu s$, the fast J221 triggers the digitizers and Q-switch for the second laser. This results in two pulses spaced $500\mu s$ apart repeating every $1ms$ for 2kHz operation.

Triple Pulse Operation(fig 2.15): An excellent demonstration of the pulse-burst mode of operation is the interleaved mode of two lasers firing in Triple-Pulse mode. In this mode after the MST trigger, the macro J221 sends master triggers spaced $1ms$ apart to the FLPSCS. For each master trigger received by the FLPSCS (after a $10\mu s$ jitter) at t=0 the four channels driving the first laser's flashlamps fire for a $310\mu s$ duration pulse. At t=$140\mu s$ (to allow for an optimum population inversion in the rods) the fast J221 triggers the Q-switch for the first laser and digitizers. Each laser

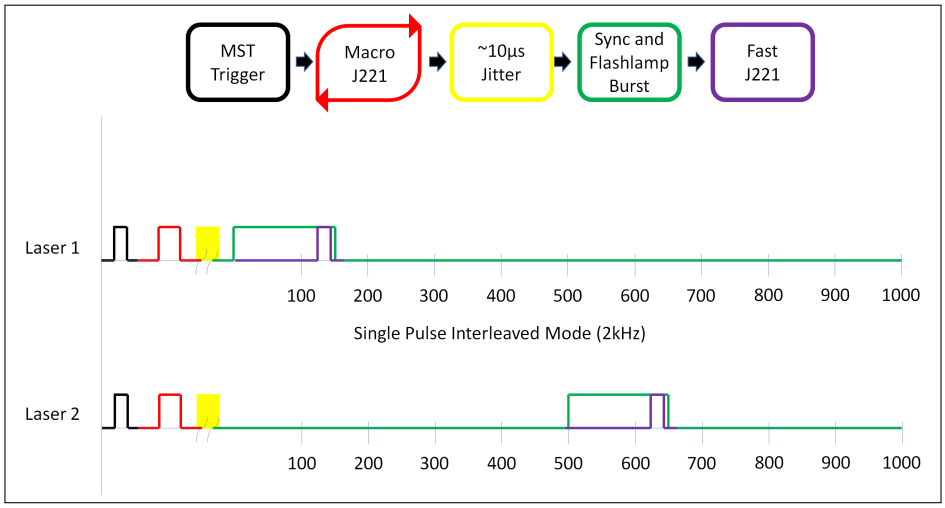

Figure 2.14: Two lasers interleaved in Single-pulse mode capable of 2kHz operation for 15ms.

pulse only partially depopulates the rod, and after a short 80μs repopulation period another laser pulse of 2J energy can be fired again. The Q-switch and digitizers are triggered again at t=220μs and t=300μs. For the second laser, a pulse delay of 40μs is used. The second laser's lamps are fired for the same 310μs duration pulse. The fast J221 triggers the digitizers and Q-switch for the second laser at t=180μs, 260μs, 340μs. This results in two pulses spaced 40μs apart repeating every 1ms, allowing 25kHz burst operation for 6 pulses, every 1ms.

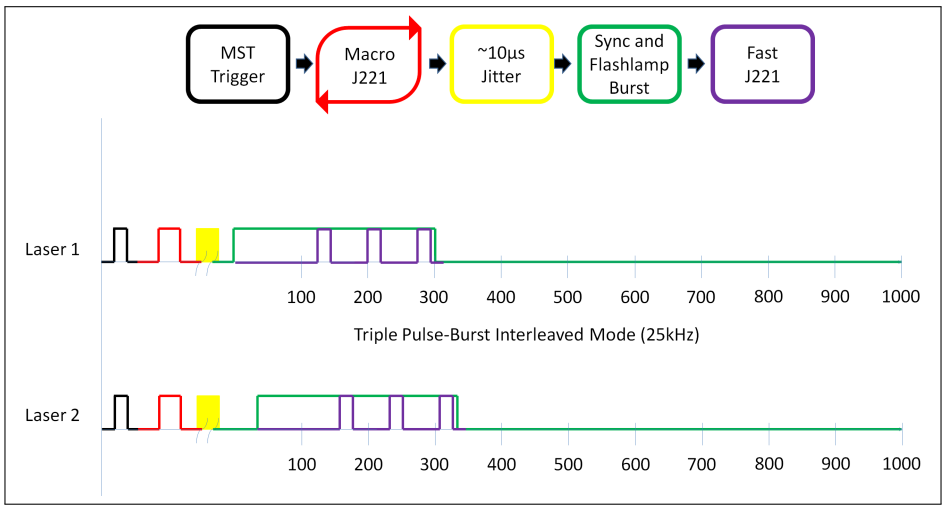

Figure 2.15: Two lasers interleaved in Triple pulse pulse-burst mode capable 25kHz operation for 6 pulses every $1ms$.

Chapter 3

MST Plasma Characteristics

3.1 Improved Confinement Discharges

As previously mentioned, the gradient in parallel current can be a drive for the instabilities that degrade energy confinement. One means of improving confinement is therefore to reduce the gradient which drives the instabilities by techniques such as Pulsed Poloidal Current Drive (PPCD). By externally driving parallel current along the edge of the plasma(3.1), PPCD attempts to flatten the parallel current profile. While extremely effective[17], PPCD discharges have high shot-to-shot variability. The same plasma current, density, and PPCD programming can often result in disparate temperature evolution.

3.2 Mode Activity and Electron Temperature Evolution

The reduced fluctuations levels from PPCD result in much smaller island widths, and a correspondingly lower level of stochasticity throughout the majority of the plasma, as can be seen in the results of a field-line tracing simulation using inputs from PPCD discharges (3.2). Thermal transport in the core of the plasma is much

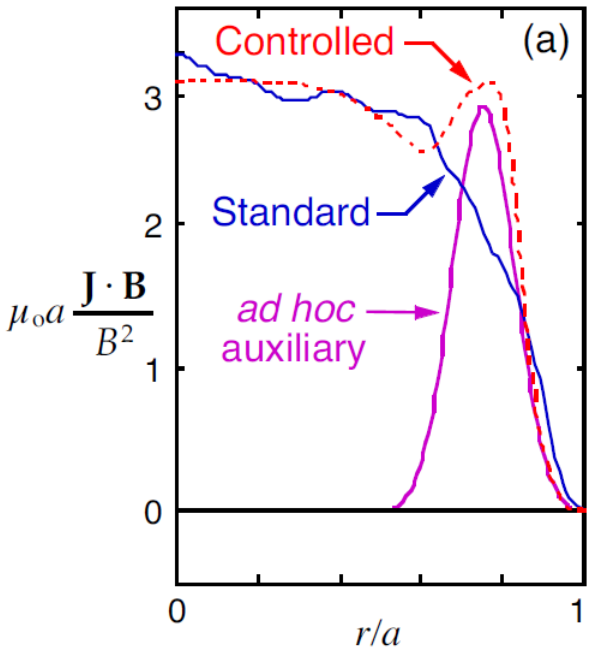

Figure 3.1: Cartoon of current profile control from 2009 MST proposal.

higher than what is predicted by the stochastic model, and is assumed to be via another mechanism (possibly electrostatic[18]), but its discussion is beyond the scope of this work. There is however a region radially inward from the reversal surface where the calculated stochastic transport is of the order of the measured transport[5]. It can be seen that in this region, the shape of the q-profile is such that even when the fluctuations are greatly reduced, many small amplitude fluctuations may still overlap, creating a stochastic region (3.3). The focus of this thesis will be on comparing the thermal transport in this region with the stochastic model for high current PPCD discharges.

Because of the reduction of fluctuations in PPCD discharges, and the high n-number of the modes in the stochastic region, the modes of interest have very small

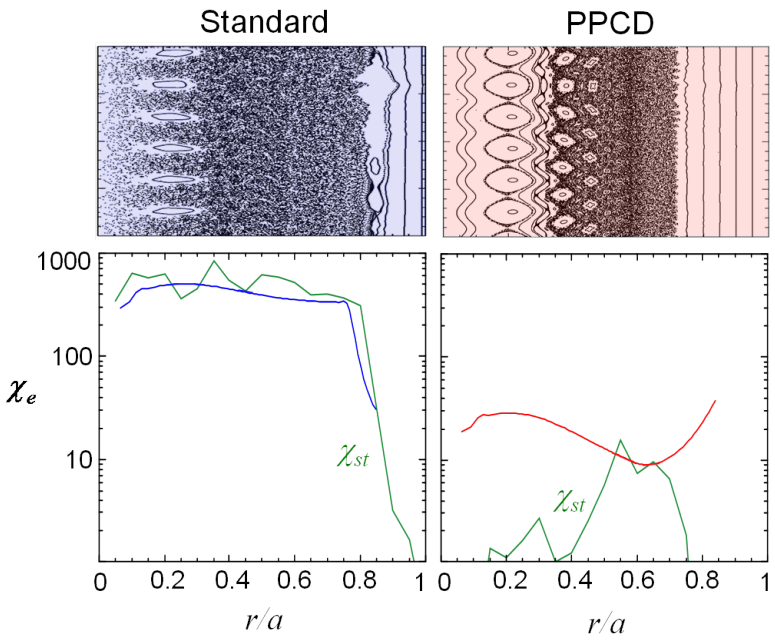

Figure 3.2: A comparison of transport in Standard vs. PPCD discharges taken from 2009 CMPD and CMSO Winter School Lecture by Dr. Sarff. Green line is calculated stochastic transport, vs. measured transport.

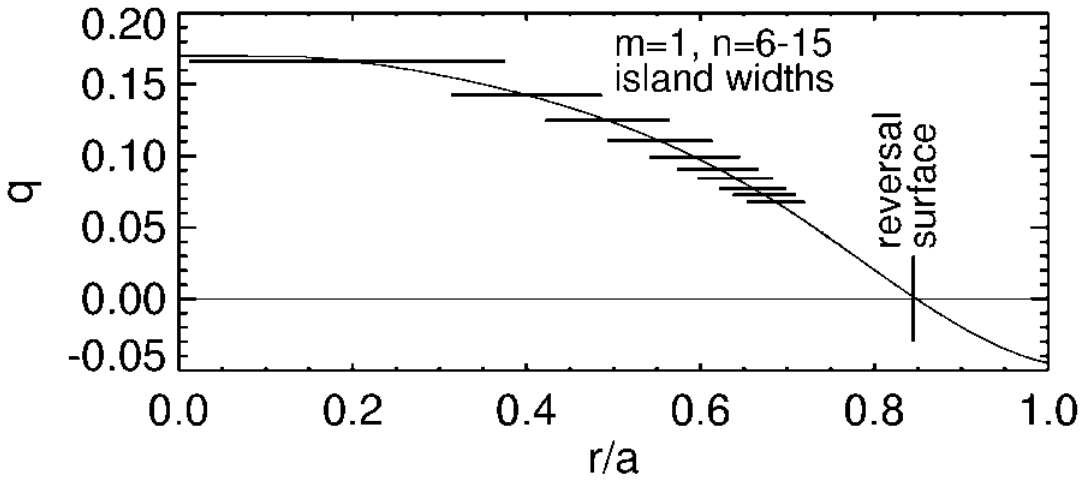

Figure 3.3: A q-profile of a standard discharge, showing the calculated widths of magnetic islands centered in their rational surface. The island widths are greatly reduced in PPCD discharges, and core stochasticity is small, however stochasticity near the reversal surface can still dominate transport.

amplitudes. The smallness of these mode amplitudes mean that the mode data requires a more careful treatment of the measured magnetic signals to arrive at the proper fluctuation amplitudes. In this regime, small error fields that arise during a plasma discharge are often on the order of the measured fluctuations, and correcting for this contribution is crucial to the understanding of transport in this regime.

3.2.1 The toroidal Array on MST

The magnetic field in MST, as measured by the toroidal array, can be written as an infinite sum of its Fourier components (see Appendix C):

$$B(\phi) = \sum_{n=0}^{\infty} c_n cos(n\phi - \delta_{cn})$$

The field is measured unintrusively by an array of 64 passive magnetic coils[19] that are spaced evenly along the toroidal direction of MST, at poloidal angle 241, allowing it only to resolve the n-number of the mode. The finite number of physical coils used to measure the field also limit the resolution of the n number. By spatially Fourier decomposing the signal, 64 coefficients from the summation can be determined[20], allowing modes up to n-31 to be resolved. Modes with n-number greater than 31 will have their fluctuation amplitudes and energy aliased into the resolved n modes, however the small amplitudes of the higher-n modes are on the order of digitizer bit noise[21], and contribute little to the overall mode amplitude.

In PPCD discharges the fluctuation amplitudes can be reduced to a point where variations in the coil-to-coil calibrations require a more careful and specific treatment of the mode amplitudes, as stationary error fields can obscure the amplitude of a measured fluctuation. While real effects of stationary fields on fluctuations (such as causing mode locking) can occur and have a real physical impact on the plasma, examining the impact of these modes on PPCD plasma discharges is beyond the scope of this work.

When stationary and slowly varying error fields (x_n, y_n or $z_n = \sqrt{x_n^2 + y_n^2}$)are present in the array (See appendix C.0.3), the calculated mode amplitude becomes:

$$B_n^2 = [a_n^2(t) + b_n^2(t)] + (x_n^2 + y_n)^2 + 2[a_n(t)x_n + b_n(t)y_n]d$$

or

$$B_n^2 = C_n^2 + Z_n^2 + 2C_n Z_n cos(\delta_Z - \delta_C)$$

It can be seen that an error-field can change the amplitude of the calculated mode amplitude from the database value to what is actually present, as well as altering the time dependence due to its non-rotating nature. In low fluctuation level situations, a careful analysis of the fluctuating signal is then very beneficial. A mode analysis script by John Sarff performs such a treatment.

We can write the field measured by the coils, including the error field, as:

$$B(\phi) = \sum c_n cos(n\phi - \delta_c) + z_n cos(n\phi - \delta_z)$$

We choose a time-window t_1 to t_2 (order $\sim$1ms) to perform the mode analysis, and average the total field measured during this period.

$$< B(\phi) >= \frac{\int_{t_1}^{t_2} \sum c_n cos(n\phi - \delta_c) + z_n cos(n\phi - \delta_z) dt}{t_2 - t_1}$$

Because the fluctuations of interest are rotating quickly past the coils in this time window (it is noted if they are found not to be rotating later in the analysis, and the specific discharge being analyzed is omitted from the data set), their amplitude

averages out to zero, leaving just the stationary error-field amplitude measured by that coil.

$$< B(\phi) >= \frac{\int_{t_1}^{t_2} \sum Z_n cos(n\phi - \delta_z) dt}{t_2 - t_1} = \sum Z_n cos(n\phi - \delta_z)$$

By subtracting this component from the raw measured signal, we are left with just the fluctuating component of the signal:

$$\begin{aligned} B(\phi,t) - < B(\phi) > &= \sum C_n(t) cos(n\phi - \delta_c) + Z_n cos(n\phi - \delta_z) - < B(\phi) > \\ &= \sum C_n(t) cos(n\phi - \delta_c) + Z_n cos(n\phi - \delta_z) - Z_n cos(n\phi - \delta_z) \\ B(\phi,t) - < B(\phi) > &= \sum C_n(t) cos(n\phi - \delta_c) \end{aligned}$$

The result is an overall lower fluctuation level that is less influenced by the presence of stationary error fields (3.4).

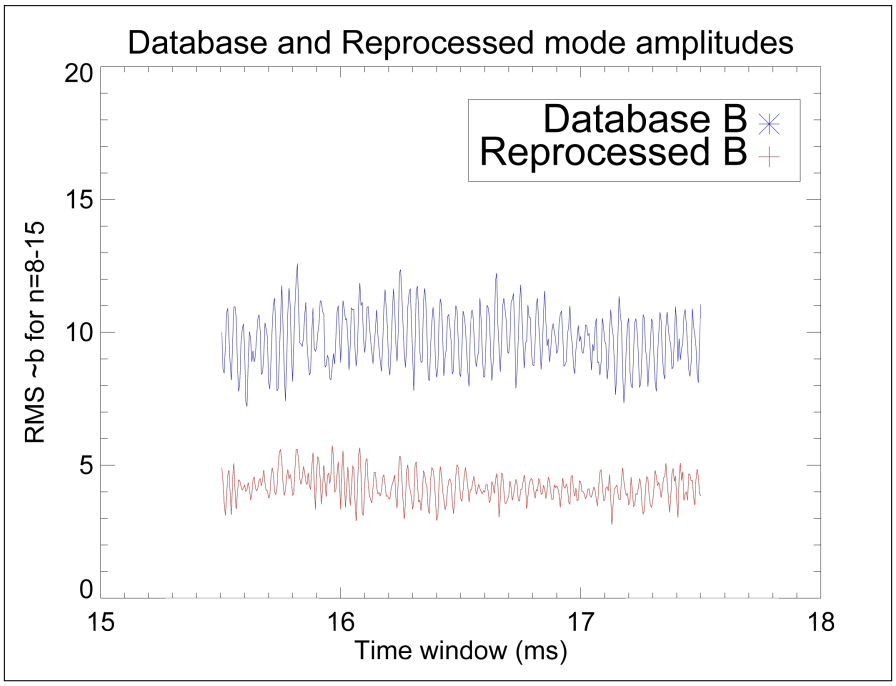

Figure 3.4: RMS mode amplitudes of n=8-16 tearing modes, comparing database fluctuation amplitudes vs. fluctuation amplitudes corrected for stationary fields.

3.2.2 Core Temperature and mode activity

The T_e profiles of PPCD discharges exhibit a flat region from the core to mid-radius, suggesting high transport that ties core T_e behavior to the mid radius. This connection between behavior in the core and mid-radius offers a strong motivation to examine the impact of transport in the stochastic mid-radius region directly on the behavior of the core electron temperature. It should be noted here that the proper calculation of stochastic transport, such as performed in Biewer et al.[5] , is an involved process not within the scope of this work. However, because stochastic transport relies on the the diffusion of magnetic field lines as given by[4]: $D_m \approx mR\frac{b_r^2}{B_z^2}$, much information could still be gained by a proper examination of the measured fluctuations in the T_e gradient region.

To examine the potential impact of stochastic transport on PPCD performance, core temperatures from high current PPCD shots from June 30, 2009 were plotted against the RMS sum of their n=8-16 mode amplitudes. All of the discharges from the set had the same voltage and timing programming, although the start and end times for good PPCD would vary. Specific shots of moderate to good quality PPCD during times starting 12-14ms and ending 17-20ms were chosen. Core electron temperature was plotted with the RMS n=8-16 mode amplitudes, and showed a clear positive correlation between a high core temperature of a specific discharge and a low RMS mode amplitude in the stochastic region (3.5): Plotting the core Te vs. the core n=5-7 modes (3.6) shows very little correlation between core Te and fluctuation amplitude.

The next step in the investigation involved a local transport analysis in the region

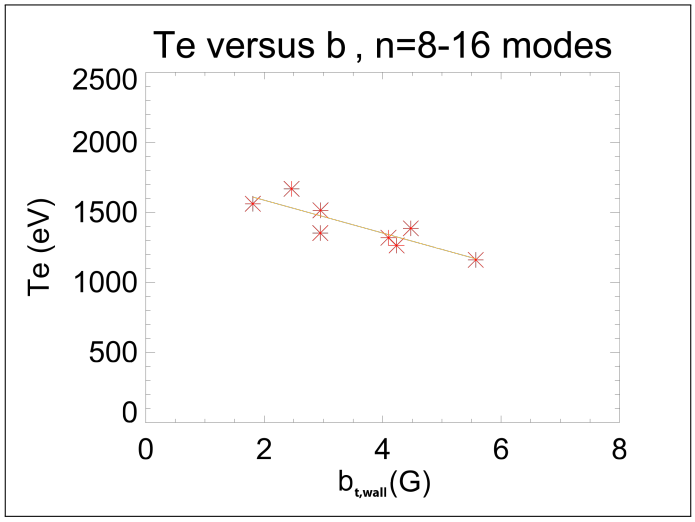

Figure 3.5: Core temperature of various PPCD discharges vs n=8-16 RMS fluctuation amplitude.

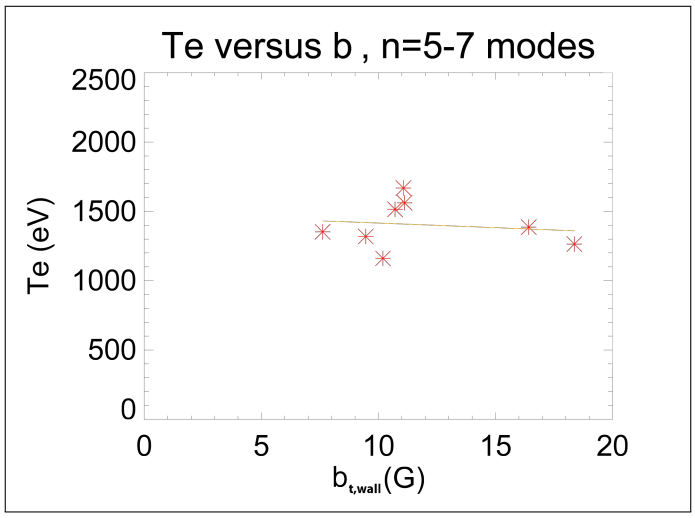

Figure 3.6: Core temperature of various PPCD discharges vs n=5-7 RMS fluctuation amplitude.

of interest (the stochastic region where the n=8-16 modes reside). This process relied on the use of a 2D equilibrium reconstruction code called MSTFit, which is outlined below.

3.3 Equilibrium Reconstruction using MSTFit

MSTFit is a 2D equilibrium reconstruction code which solves the Grad Shafranov equation in a torus:

$$\Delta^* \psi = -\mu_0 R^2 \frac{dp}{d\psi} - \frac{1}{2}\frac{dF^2}{d\psi}$$

Where

$$F(\psi) = RB_\phi$$

$$\Delta^* \psi = R\frac{\partial}{\partial R}\left(\frac{1}{R}\frac{\partial \psi}{\partial R}\right) + \frac{\partial^2 \psi}{\partial Z^2}$$

The plasma current, pressure, and reversal and pinch parameters are taken as inputs, and a current profile is created from this input. If there are additional diagnostics such as MSE[22] or probe measurements, the data is compared to expected values for the MSTFit generated equilibrium, and used in a χ^2 minimization routine to additionally constrain the fit. The pressure input is the straightforward product of electron (from Thomson scattering) and ion temperature and density. A full write-up of how MSTFit works is presented in Anderson's thesis[23].

3.3.1 Fitting

Due to the high core electron temperatures associated with high current PPCD discharges, and current limitations with TS hardware's ability to properly resolve those temperatures, the T_e profile in the core of a high current PPCD discharge often has quite a bit of structure, and large error bars. It was beyond the scope of this work to determine whether this structure was real and occurring in the plasma, or systematic and an artifact of the diagnostic. Therefore, rather than using a spline fit (one that offers more freedom in fitting) for the electron temperature profile, a polynomial fit – which makes a relatively good assumption that the Te profile is flat from the core to the mid radius, and still captures the characteristics of the temperature gradient in the mid-radius – was used for the T_e profile allowing for a consistent and reliable fit between different sets of data (3.7)

$$T_e(\psi) = (T_{e,core} - T_{e,edge})(1 - \psi^a)^b + T_{e,edge}$$

The model generates the best fit possible by adjusting the domain ψ, and parameters a, and b. Varying ψ adjusts the position of the gradient, while a (~ 4)and b (~ 1.2) adjust the steepness of the falloff.

While robust, the polynomial model can only fit monotonically decreasing functions with one gradient region. As a result, many PPCD discharges that appear to have two flat top regions–a large flat core and a smaller flat region located near the reversal surface(3.8)–were omitted from the dataset due to the inaccuracy of the tem-

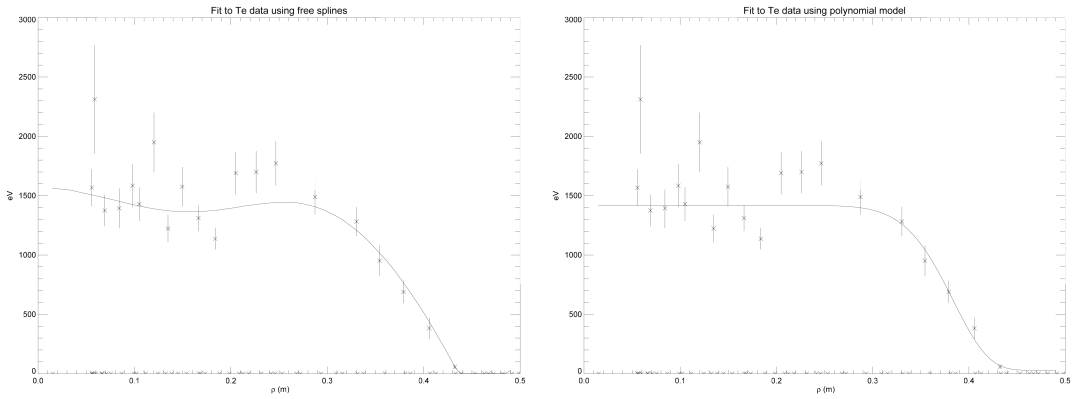

Figure 3.7: Example of a free spline fit on the left versus the polynomial temperature fit on the right.

perature fit in this region. It is worth mentioning that there is strong motivation for a capability to better resolve and fit this region, as cursory observations suggest that possibly all PPCD discharges show this behavior to some extent. Additionally, the highest performing PPCD discharges seem to strongly correlate to a pronounced double flattening of the profile, suggesting the possibility of two radially separated transport barriers resulting in better core energy confinement.

Once the magnetic equilibrium that best fits the data is generated, a local transport analysis can be performed.

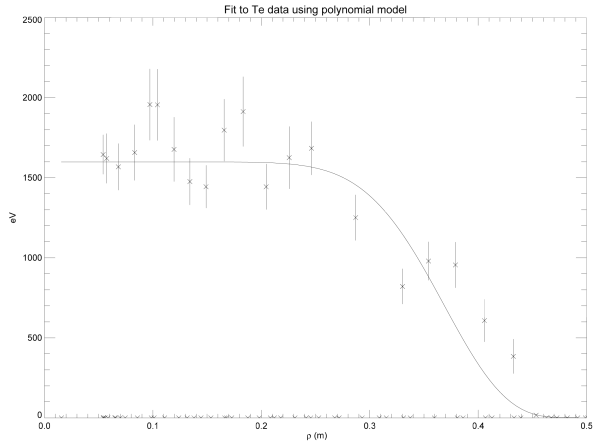

Figure 3.8: An electron temperature profile displaying two pronounced gradient regions (centered around r 0.3m and r 0.4m), resulting in a poor temperature profile fit.

3.3.2 Transport

The transport of thermal energy within the plasma can be broken down into conductive and convective components:

$$Q_e = \chi_e n_e \nabla T_e - \frac{5}{2}\Gamma_e T_e$$

The energy transport resulting from particle flux is generally a small contribution to the overall transport, additionally, particle flux measurement data from MST's D_α array was unreliable from the day the 500kA PPCD discharge data set was taken. To verify the smallness of the convective contribution to energy transport, an RFX particle transport reconstruction code was applied to data from one time point in one specific discharge to estimate the convective transport contribution to the overall χ_e,

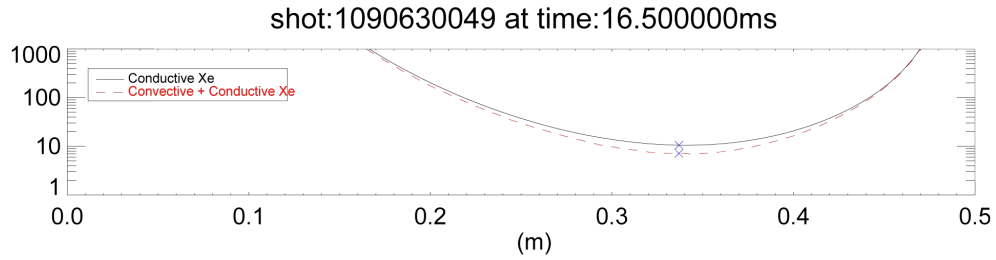

Figure 3.9: A plot of χ_e with and without particle transport contributions.

and can be seen to be very small (3.9). Therefore, convective transport contributions have been omitted from the following transport calculations, and the the task becomes to properly calculate the local heating and loss terms in order to solve for:

$$\chi_e = \frac{-Q_e}{n_e \nabla T_e}$$

Where Q_e is the total electron heat flux:

$$Q_e = \frac{P_\Omega - \dot{W}_e - P_{etoi} - P_{rad}}{4\pi^2 r \rho}$$

Where the terms in the numerator are the volume-integrated values of: ohmic heating power, change in electron thermal energy, electron-to-ion collisional losses, and radiative losses, and the denominator is the surface area of the torus enclosing the volume. With the current profile generated by MSTFit, it is possible to calculate the local ohmic heating in the plasma by $P_\Omega = \int \eta J^2 \, dV$. The change in thermal energy $\dot{W}_e$ was set to zero for this calculation, as all of the data points for the PPCD data set were either from periods of constant or increasing T_e. Setting $\dot{W}_e = 0$ in these situations at most results in an overestimation of the value of χ_e. Alternatives

to the omission of $\dot{W}_e$ are discussed in the summary section. Electron to ion loss P_{etoi} is based on a simple collisional model, and the radiated power P_{rad} is empirically set to be proportional to the ohmic heating power. More detail is offered in the appendix (C).

In order to properly identify the region of interest in the plasma to examine the local transport with respect to fluctuation-induced stochasticity, we examine the q-profile generated by MSTFit. The rational surfaces, radial locations at which the q-profile becomes rational $q = \frac{m}{n}$, give the positions at which the fluctuations of interest reside. The m=1, n=8-16 tearing modes would be centered on radial locations where q=1/8, 1/9, etc. As any stochastic transport would be the result of an overlap of the islands centered on these locations, χ_e is evaluated at the geometric mean of the n=8-15 rational surface locations. A plot of the local χ_e vs. the n=8-15 mode amplitudes shows that χ_e scales linearly with the squared sum of the fluctuation amplitudes (3.10).

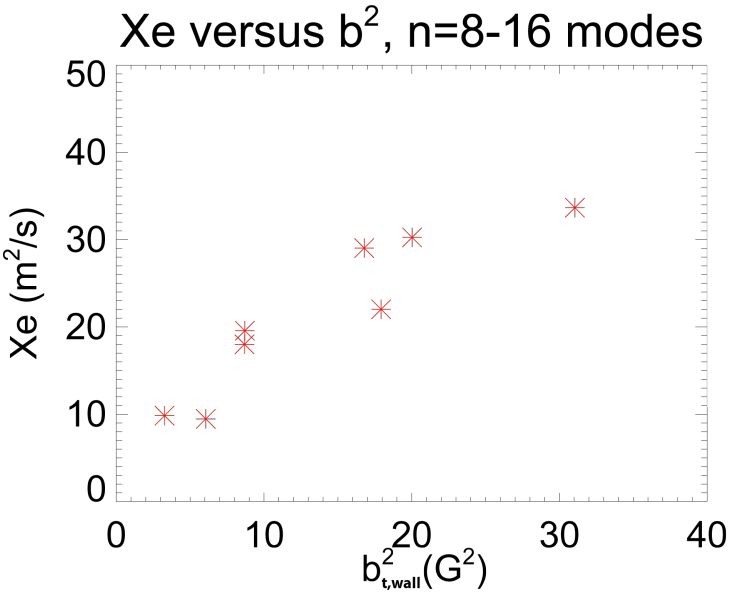

Figure 3.10: A plot of local χ_e vs. rms fluctuation amplitudes of n=8-16 modes.

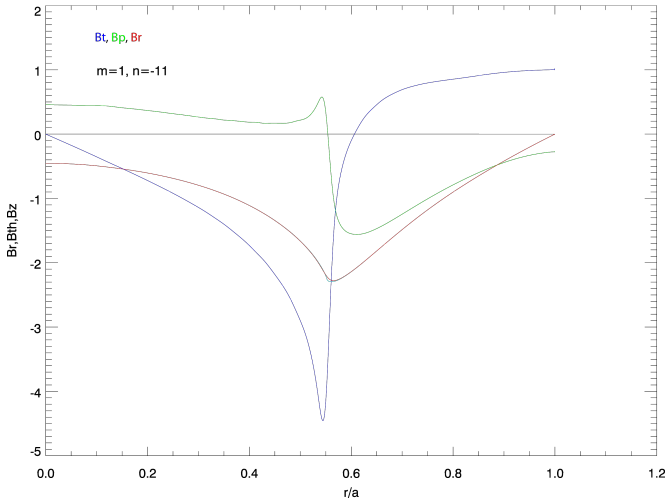

Figure 3.11: A plot of the m=1, n=-11 eigenmode for Br, Bt, and Bp.

3.3.3 χ_e and stochasticity limits

This is a good moment to compare this result with an estimate of the expected transport due to a stochastic field. The stochastic energy transport in a collisionless plasma can be expressed as: $\chi_{e,stoch} = v_{Te} D_m$ where v_{Te} is the thermal velocity, and the magnetic diffusion D_m of a stochastic field is given by $D_m = L_{ac} \frac{<\tilde{b_r^2}>}{B_0^2}$. And L_{ac} is the auto-correlation length of the stochastic field, and can be approximated as $L_{ac} \approx \frac{\pi}{\Delta k_\parallel}$ where $\Delta k_\parallel$ is the spread in the mode spectrum and $k_\parallel = \vec{k} \cdot \vec{B}$.

As the values for the equilibrium and perturbed field must be evaluated in the stochastic region of the plasma, we make use of the equilibrium generated by MSTFit, as well as calculated linear eigenmodes of the m=1 n=8-15 fluctuations, allowing us to locate the region of interest for our calculations and convert from the measured $\tilde{b_t}$ at the wall to the $\tilde{b_r}$ local to the region of interest (3.11).

The spread in the mode spectrum[24] (appendix E) $\Delta k_{||} = \frac{n}{R}\Delta r \left| \frac{\partial}{\partial r}\left[\frac{m}{n}\frac{RB_\theta}{r|B|} + \frac{B_\phi}{|B|}\right]\right|_{rs}$ can be numerically evaluated using the q-profile and equilibrium B_θ, B_ϕ profiles generated by MSTFit. Δr is defined to be a fluctuation's mode radial width. The interpretation of this value is expanded upon later.

The MSTFit equilibrium allows us to directly evaluate the field in the gradient region (∼0.45T). From the eigenmode plots previously calculated by Sovinec[25], it is possible to convert the toroidal fluctuation amplitude measured at the wall to that of the radial amplitude in the stochastic region. We see that the $\tilde{b_r}$ of a specific eigenmode at its rational surface within the plasma is on the order of 2-3 times the $\tilde{b_t}$ measured at the wall. The thermal velocity is a simple conversion from the measured electron temperature T_e.

The one parameter that is not well constrained by data in the calculation for stochastic transport is the mode radial width Δr. The mode radial width can be interpreted as a variety of values, from the full width half maximum of the fluctuation's radial eigenfunction, to a tearing mode's island width. The radial component of fluctuations on MST, while peaked at their specific rational surfaces, also span the minor radius of the plasma. A half-width of the fluctuation for the n=8-16 fluctuations would have $\Delta r \sim 0.2m$. The Δr term can also be interpreted as the island width associated with the fluctuation.. As $\chi_{e,stoch}$ scales directly with what we choose for this value, it is important to note its effects. For an initial estimate, a calculation with Δr taken to be $\sim$ 0.1m, on the order of an n $\sim$ 10 island, results in a reasonable autocorrelation length of $L_{ac} \approx$ 1m. The calculated stochastic transport for this value of Δr gives a $\chi_{e,stoch}$ approximately twice that which is measured (3.12).

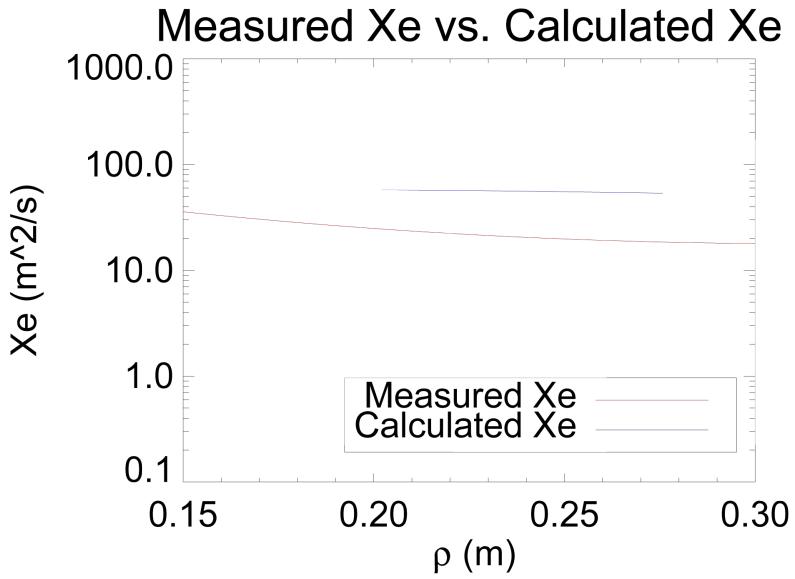

Figure 3.12: A plot of measured χ_e (red) vs. χ_e obtained from the previous calculation.

Instead the mode radial width was adjusted for one data point (one time point in, one discharge) such that the experimental and calculated χ_e agreed; this resulted in $\Delta r \sim .14$m, and an autocorrelation length $L_{ac} \sim 0.5$m. This value of Δr was used for all of the data points. Plotting the calculated $\chi_{e,stoch}$ and measured $\chi_{e,meas}$ against $\frac{b_r^2}{B^2}$ data (3.13) clearly shows strong agreement between measured and calculated stochastic transport.

As this analysis uses local values for all of the parameters in the transport calculations, it was possible to expand the calculation beyond the data set where the core T_e tracked with n=8-16 modes. The same calculations for the measured and expected stochastic transport was applied to all of the properly fit data points available. The inclusion of these additional data points show more scatter between the core T_e relative to RMS fluctuation amplitude (3.14), however the agreement between measured

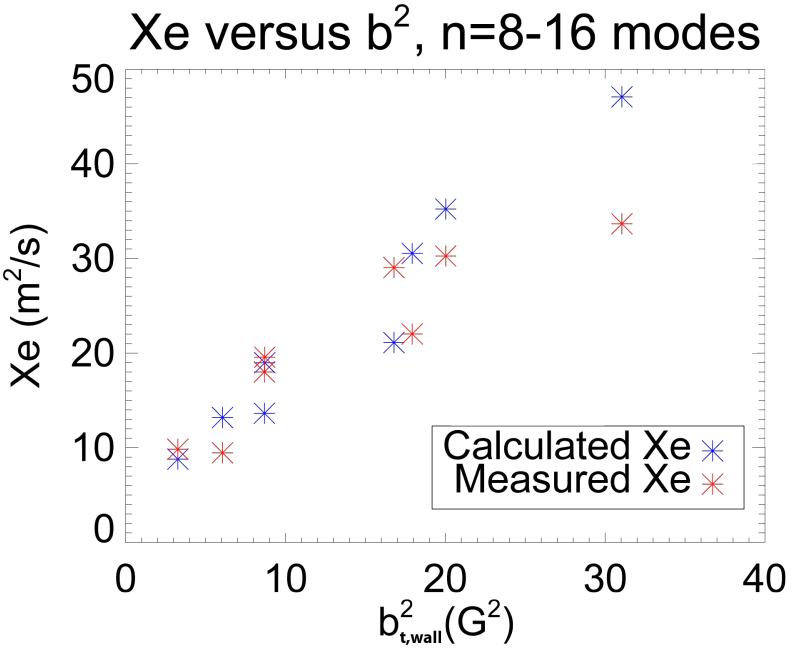

Figure 3.13: A plot of measured χ_e (red) vs. calculated stochastic χ_e for select data points.

transport and expected stochastic transport remains clear (3.15).

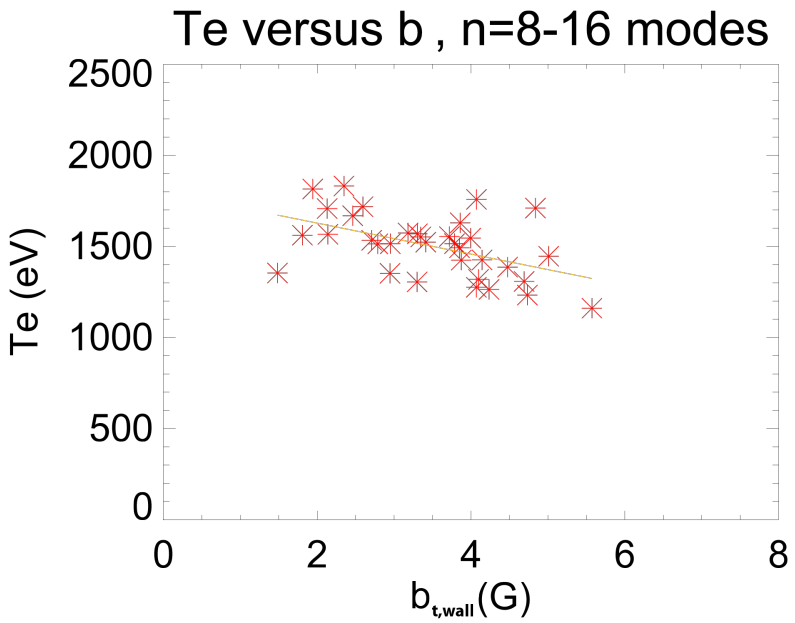

Figure 3.14: Core temperature of various PPCD discharges vs n=8-16 RMS fluctuation amplitude for all data points.

This demonstrates a strong argument for the presence of stochastic thermal transport in the mid-radius region of PPCD discharges, even in the case of very low relative fluctuation amplitudes.

Because the choice of Δr allowed the calculation to fit the measured data so well, it is important examine the implications of this value. Generally, the autocorrelation length is of the order of a field line's transit length. At the core where the field is completely toroidal, the autocorrelation length is on the order of the major radius $L_{ac} \sim \pi R \sim 5m$. At the reversal surface where the field is compeltely poloidal, we have $L_{ac} \sim \pi a \sim 1.5m$. The small D_m that results from our choice of Δr and the resulting small L_{ac} suggests that our calculation for stochastic transport was applied in in a regime of very low field stochasticity. A calculation of the island overlap

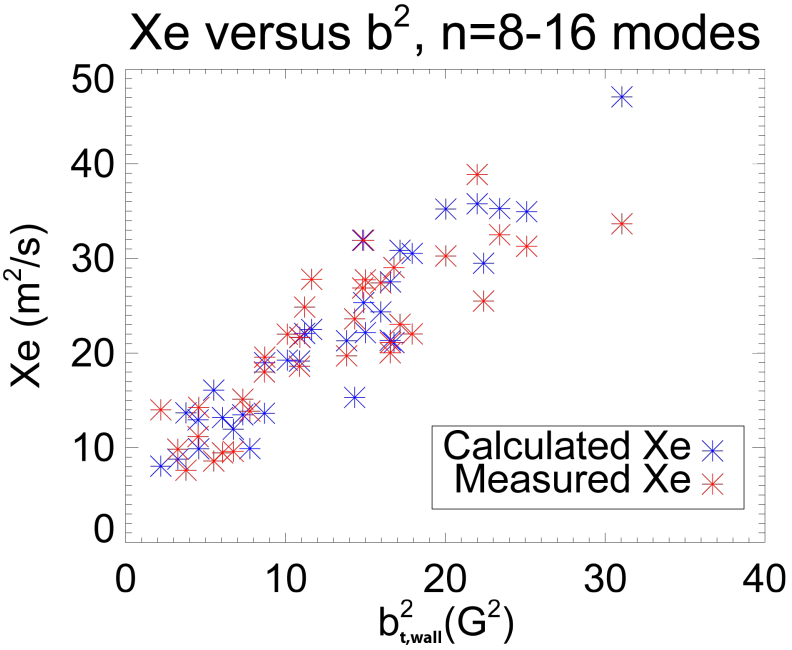

Figure 3.15: A plot of measured χ_e (red) vs. calculated stochastic χ_e for all data points.

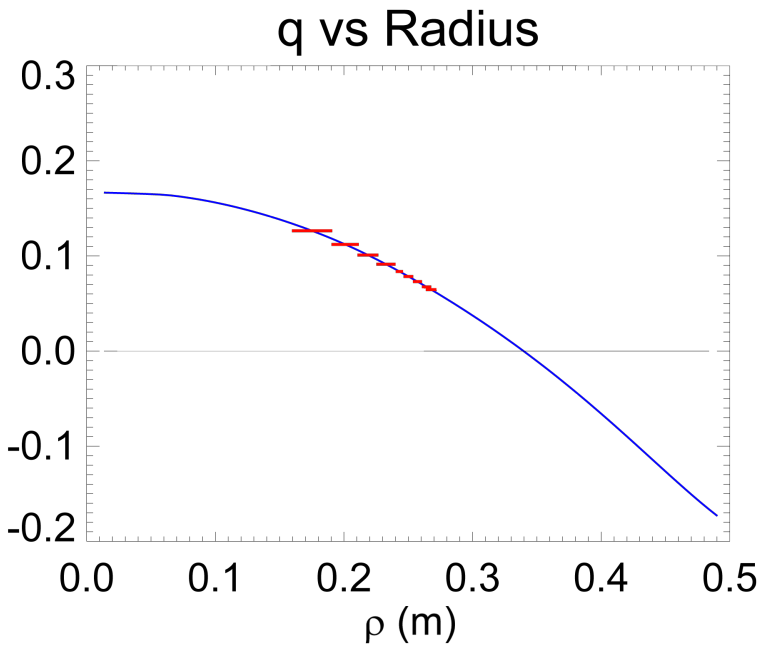

Figure 3.16: The q-profile from a representative data point with calculated island widths for n=8-16 fluctuations.

(Appendix F) using the q-profile from a representative data point (3.16) confirms that the islands are small, while there is still an overlap between neighboring islands. The calculated Chirikov[26] stochasticity parameter (Appendix F) for this region is almost exactly unity.

Chapter 4

Summary and suggestions for future work

4.1 Summary

While many approximations were made in the previous calculations to arrive at these results, the clear $\chi_e \sim {b_r}^2 B$ scaling characteristic of stochastic transport clearly showed through. And although stochastic energy transport is the dominant energy transport mechanism in standard discharges in MST due to the large fluctuation amplitudes, it is evident from this work that stochastic energy transport is still capable of describing the behavior seen in the mid-radius of the plasma in PPCD discharges, a regime of very low fluctuation amplitudes. The agreement between experimental and calculated stochastic transport is seen in all of the data points in the data set, and for discharges where high core transport results in a flat T_e profile to the mid-radius, the stochastic transport at the mid-radius becomes a strong determinant in the evolution of the core electron temperature.

It is very interesting to note that stochastic transport in MST is generally associated with poor energy confinement[27] due to the large fluctuation amplitudes in standard discharges. However in this situation the stochastic transport mechanism

plays a significant role even in regimes of very low magnetic fluctuation amplitudes; where very small transport ($\chi_e \sim 10\frac{m^2}{s}$) still scales following what is predicted by the Rechester-Rosenbluth model for stochastic transport. The result is that the regions of stochastic transport in these PPCD discharges exhibit some of the lowest transport in the plasma.

In line with what Biewer[5] saw in his work, the stochastic transport mechanism is applicable in regimes of very low fluctuation amplitudes, as long as the field is still stochastic. What is interesting in this situation is that estimates of the island widths for the n=8-16 regions in our 500kA PPCD data set, and the spacing between rational surfaces, suggest that the island overlap and resulting stochasticity is actually quite small. This is reflected in the value which we set our mode radial width as well. The strong agreement between our data and the model for stochastic transport that should be at its limit of applicability greatly encourages additional investigation.

Some further refinements are suggested in the following pages for a more detailed investigation into probing the limits of the stochastic transport mechanism.

4.2 Suggestions for refinements and future work

4.2.1 Measured transport calculation refinements

Two approximations were made when calculating the measured transport: setting the change in electron thermal energy $\dot{W}_e = 0$, and omission of the convective term in the power balance equation.

One of the reasons setting $\dot{W}_e = 0$ in the power balance equation was necessary because large errors result from taking the time derivative of T_e data points, which already have inherently large error bars due to the high plasma temperatures. As previously mentioned, for the PPCD data set, this results in an over-estimate of χ_e as the stored thermal energy is either constant or increasing throughout the enhanced confinement period. However, short of hardware improvements to the TS system itself, there still exist a few approximations that can still be made to either minimize the effect of omitting $\dot{W}_e$, or offer a more reasonable lower bound to $\dot{W}_e$.

One method is to take advantage of the characteristic that is prominent in many PPCD discharges: the plateauing of core T_e. This can occur at any point in the discharge, and generally the temperature profile remains mostly unchanged for the remaining duration of the enhanced confinement period. Barring significant changes in the density profile between time points, selecting data points from only this plateau period would lend much more accuracy in omitting the change in thermal energy term. Unfortunately this method automatically excludes using the best performing PPCD discharges from the data set, as the core temperature and stored electron energy for

those discharges steadily increase until the end of PPCD.

For certain PPCD discharges, there also exist periods of steady and smooth Te increase throughout the core and mid-radius of the plasma. For discharges where the density profile does not vary much during this period, it is possible to arrive at an estimate of the $\dot{W}_e$ term by taking a linear time derivative between data points.

For the particle balance, as previously noted, there exists a particle transport reconstruction code ported from RFX that is capable of utilizing MSTFit's D_α array data to more accurately correct for the convective term in the transport calculations. Still, this contribution was seen to be small in the handful of data points it was previously applied to.

4.2.2 Stochastic transport calculation

For a more rigorous calculation of the expected stochastic transport, a direct numerical simulation of select PPCD discharges via field line tracing is required. By using the linear eigenmodes calculated via DEBS, which solves the nonlinear MHD equations in a cylindrical geometry. The magnetic field configuration can be calculated by the field-line tracing routine MAL, using the DEBS results and measured fields as inputs. Previously[5], this has allowed for the direct calculation of the radial excursion Δr a field line undergoes, allowing for the calculation $\chi_{e,MAL} = v_{T_e} \left\langle \frac{\Delta r^2}{2\Delta L} \right\rangle$ as a direct comparison to $\chi_{e,stoch}$ given by the Rechester-Rosenbluth model. However, the above equation for χ_e is only assumed valid for regions of high stochasticity. A numerical calculation of the of $\chi_{e,MAL}$ compared with $\chi_{e,meas}$ and $\chi_{e,stoch}$ in this case would give insight into the the applicability of stochastic transport in this regime, as well as allow for a better estimate of the actual stochasticity of the field in the mid-radius.

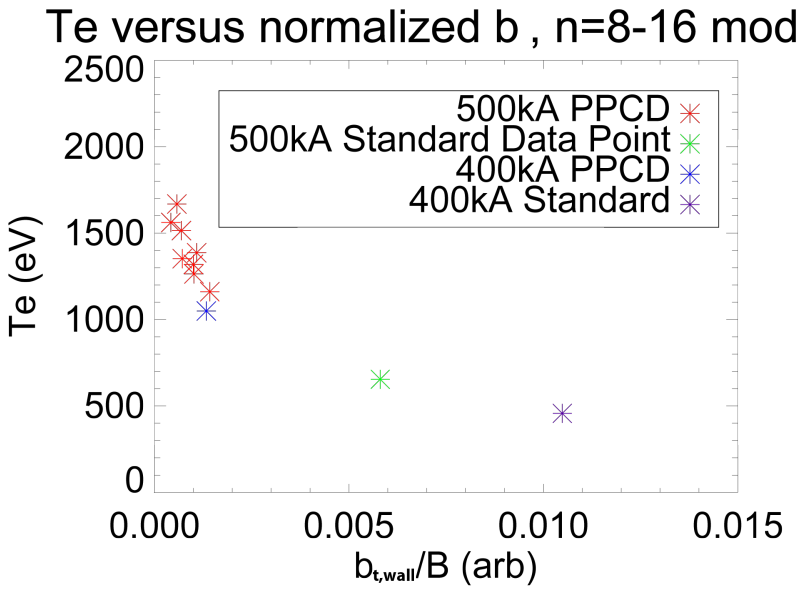

Figure 4.1: tbpp.

Probing the limits of stochastic energy transport

The previous refinements would improve the accuracy the results seen for the 500kA PPCD discharges, and also be useful in the many other operating regimes MST is capable of. An examination of peak core T_e with respect RMS fluctuation amplitudes for four different operating modes–500kA PPCD, 500kA standard, 400kA PPCD, and 400kA standard discharges–shows a clear continuation of relationship seen previously (4.1). The same analysis done in this thesis could easily be applied to those different operating regimes, and would greatly increase our understanding of the scope of applicability of the stochastic energy transport mechanism.

Bibliography

1. J.S. Sarff, J.C. Sprott, and L. Turner. Equilibrium studies of a poloidal divertor pinch with a reversed toroidal field. *Phys. Fluids*, 30(7):2155–62, March 1987.

2. S. Ortolani and D. Schnack. Magnetohydrodynamics of plasma relaxation. 1993.

3. R. N. Dexter, D. W. Kerst, T. W. Lovell, Prager. S. C., and J. C. Sprott. The madison symmetric torus. *Fusion Technol.*, 19(1):131–139, 1991.

4. A. B. Rechester and M. N. Rosenbluth. Electron heat transport in a tokamak with destroyed magnetic surfaces. *Physical Review Letters*, 40(1):38–41, Jan 1978.

5. T. M. Biewer, C. B. Forest, J. K. Anderson, G. Fiksel, B. Hudson, S. C. Prager, J. S. Sarff, J. C. Wright, D. L. Brower, W. X. Ding, and S. D. Terry. Electron heat transport measured in a stochastic magnetic field. *Phys. Rev. Lett.*, 91(4):045004, Jul 2003.

6. J.S. Sarff, S.A. Hokin, H. Ji, S.C. Prager, and C.R. Sovinec. Fluctuation and transport reduction in a reversed field pinch by inductive poloidal current drive. *Phys. Rev. Lett.*, 72(23):3670–73, June 1994.

7. B.E. et al. Chapman. Reduced edge instability and improved confinement in the mst reversed-field pinch. *Phys. Rev. Lett.*, 87(20):205001, Nov 2001.

8. D J Den Hartog and M Cekic. *Meas. Sci. Technol.*, 5:1115–1123, 1994.

9. N.E. Lanier, S.P. Gerhardt, and D.J. Den Hartog. Low-cost, robust, filtered spectrometer for absolute intensity measurements in the soft x-ray region. *Rev. Sci. Instrum.*, 72(1):1188–91, Jan 2001.

10. H.D. Stephens. *Electron Temperature Structures Associated With Magnetic Tearing Modes in the Madison Symmetric Torus*. PhD thesis, University of Wisconsin at Madison, 2010.

11. SpectronLaser. User manual for sl858g nd:yag laser system.

12. J.A. Reusch, M.T. Borchardt, D.J. Den Hartog, A.F. Falkowski, D.J. Holly, R. O'Connell, and H.D. Stephens. Multipoint thomson scattering diagnostic for the madison symmetric torus reversed-field pinch. *Review of Scientific Instruments*, 79(10):10E733 (5 pp.), Oct. 2008.

13. H. D. Stephens, M. T. Borchardt, D. J. Den Hartog, A. F. Falkowski, D. J. Holly, R. O'Connell, and J. A. Reusch. Calibration of a thomson scattering diagnostic for fluctuation measurements. *Review of Scientific Instruments*, 79(10):10E734 (5 pp.), Oct. 2008.

14. TS alignment section of MST plasma wiki. `http://hackserver.physics.wisc.edu/wiki/index.php/TS_Alignment`.

15. D. J. Den Hartog, J. R. Ambuel, M. T. Borchardt, J. A. Reusch, P. E. Robl, and Y. M. Yang. Pulse-burst operation of standard nd:yag lasers. *to be published in Journal of Physics: Conference Series*, 2010.

16. W.S. Harris, D.J. Den Hartog, and N.C. Hurst. Initial operation of a pulse-burst laser system for high-repetition-rate thomson scattering. *Rev. Sci. Instrum.*, 81:10D505, 2010.

17. J.S. Sarff, N.E. Lanier, S.C. Prager, and M.R. Stoneking. Increased confinement and β by inductive poloidal current drive in the reversed field pinch. *Phys. Rev. Lett.*, 79(1):62–5, Jan 1997.

18. J. Lei, P.M. Schoch, D.R. Demers, U. Shah, and K.A. Connor. Core electrostatic fluctuations and particle transport in a reversed-field pinch. *Phys. Rev. Lett.*, 89(27):275001, Dec 2002.

19. A.F. Almagri, S. Assadi, S.C. Prager, J.S. Sarff, and D.W. Kerst. Equilibrium studies of a poloidal divertor pinch with a reversed toroidal field. *Phys. Fluids B*, 4(12):4080, Aug 1992.

20. D. Craig. Magnetic mode analysis in MST, 2005.

21. T. M. Biewer. *ELECTRON THERMAL TRANSPORT IN THE MADISON SYMMETRIC TORUS*. PhD thesis, University of Wisconsin at Madison, 2002.

22. D. Craig, D. J. Den Hartog, G. Fiksel, V. I. Davydenko, and A. A. Ivanov. First charge exchange recombination spectroscopy and motional stark effect results from

the madison symmetric torus reversed field pinch. volume 72, pages 1008–1011. AIP, 2001.

23. J. K. Anderson. *Measurement of the Electrical Resistivity Profile in the Madison Symmetric Torus*. PhD thesis, University of Wisconsin at Madison, 2001.

24. J.A. Krommes, C. Oberman, and R.G. Kleva.

25. C.R. Sovinec. *Magnetohydronynamic Simulations of Noninductive Helicity Injection in the Reversed-Field Pinch and Tokamak*. PhD thesis, University of Wisconsin at Madison, 1995.

26. G.M. Zaslavsky and B.V. Chirikov. *Sov. Phys. Usp.*

27. J.S. Sarff, S.A. Hokin, H. Ji, S.C. Prager, and C.R. Sovinec. Fluctuation and transport reduction in a reversed field pinch by inductive poloidal current drive. *Phys. Rev. Lett.*, 72(23):3670, 1994.

28. B.E. et al. Chapman. Generation and confinement of hot ions and electrons in a reversed-field pinch plasma. *Plas. Phys. Controll. Fusion*, 52:12048, 2010.

Appendix A

Alignment

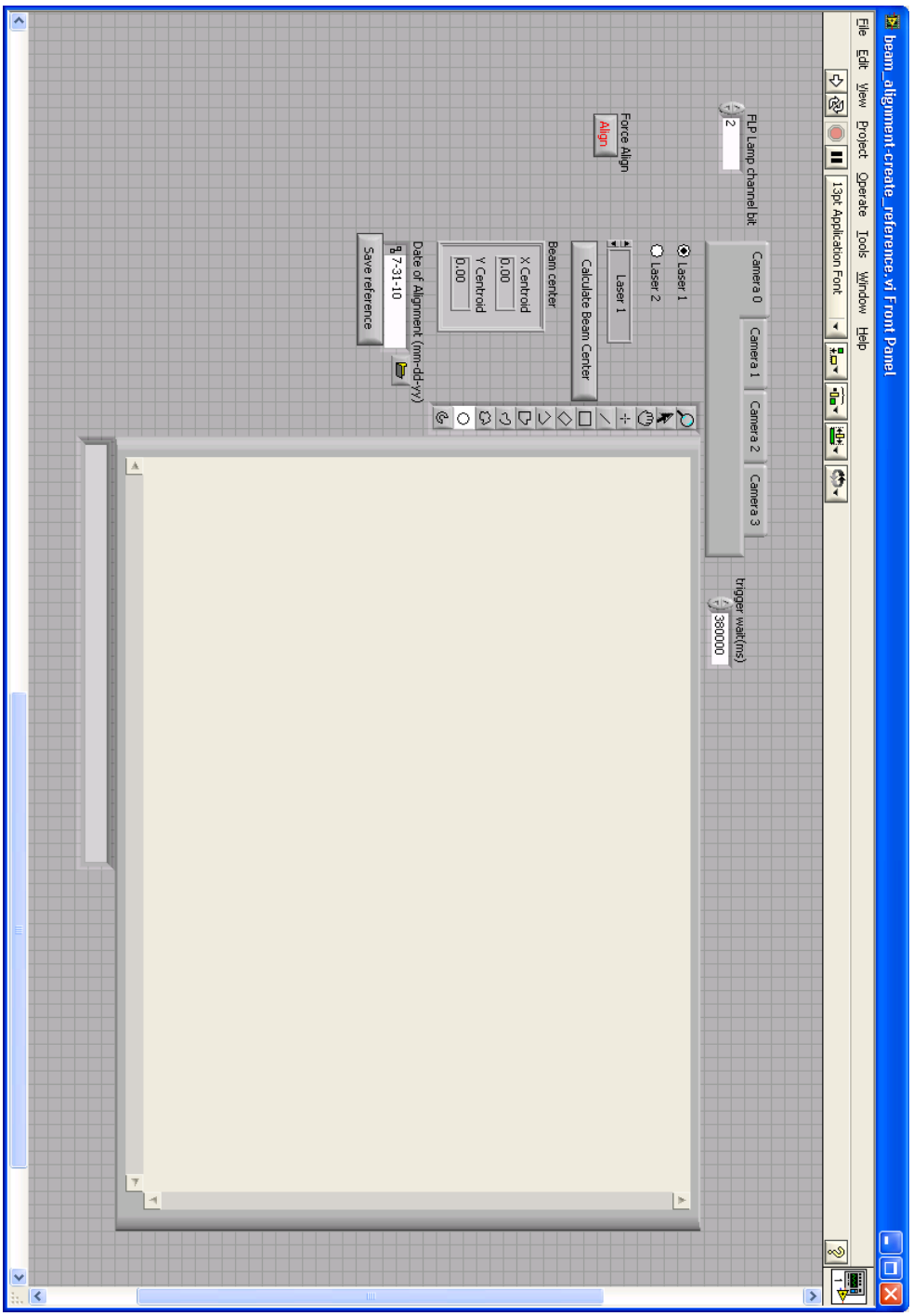

Figure A.1: align-create-front.

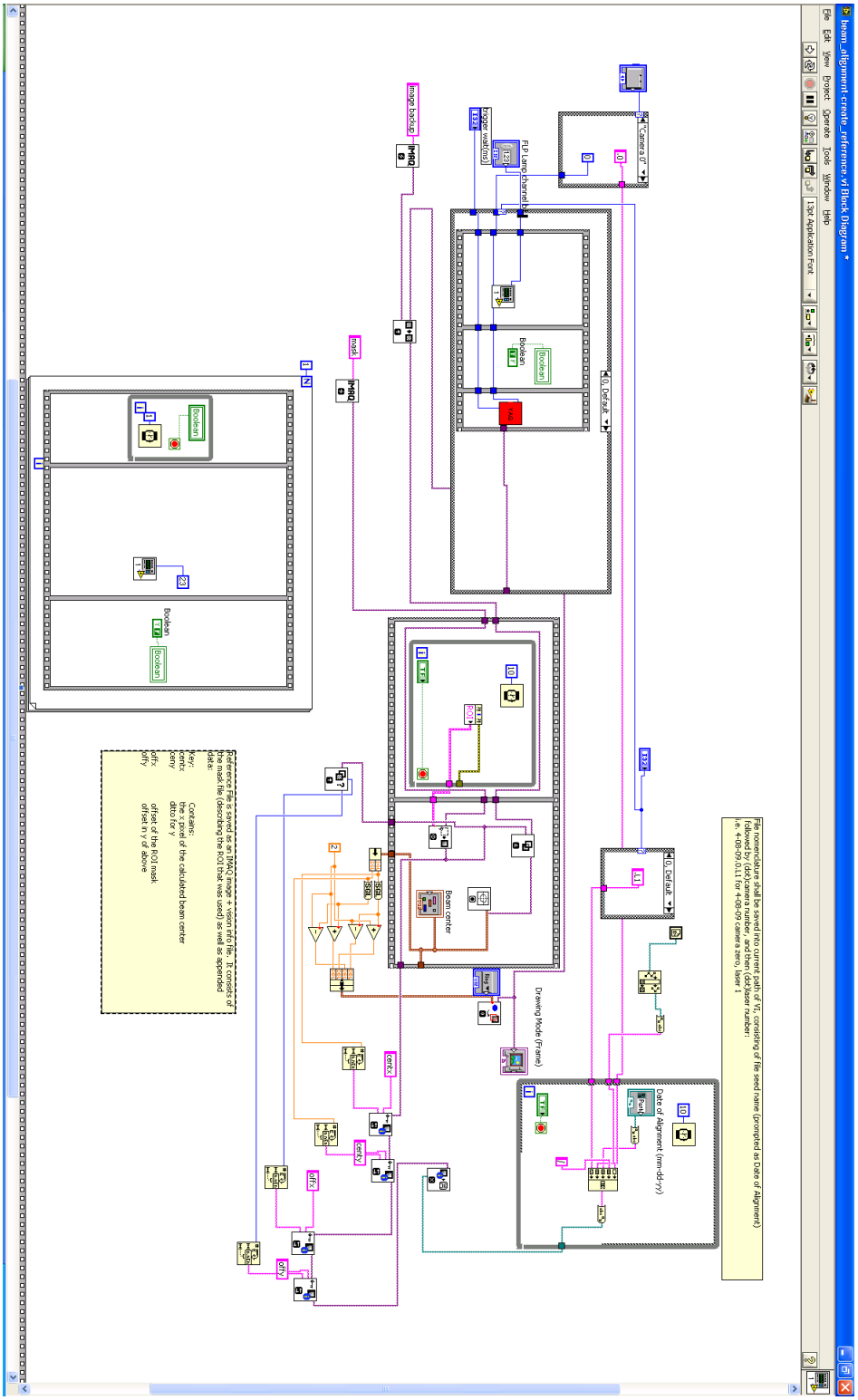

Figure A.2: align-create-back.

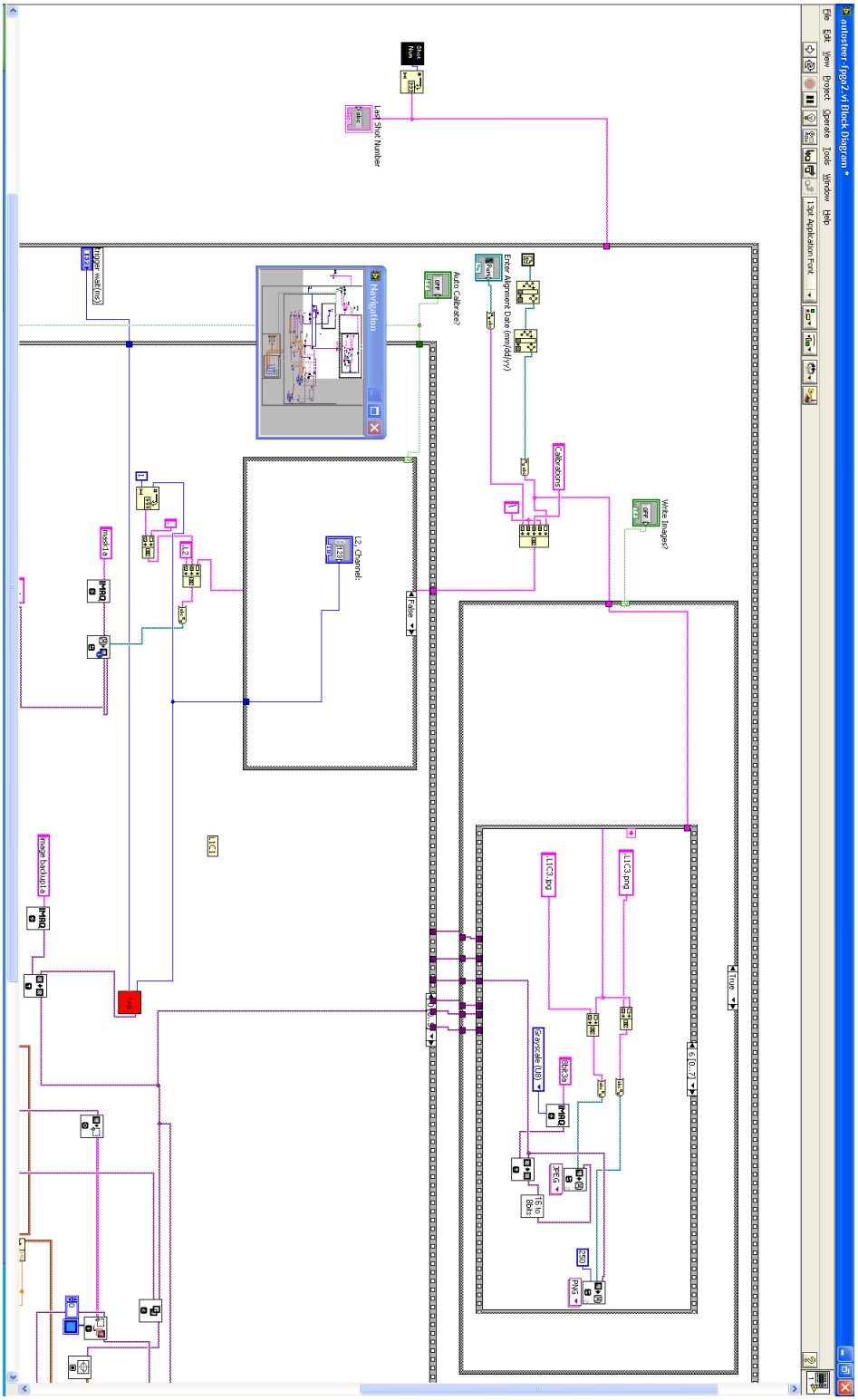

Figure A.3: align-runtime-backtop.

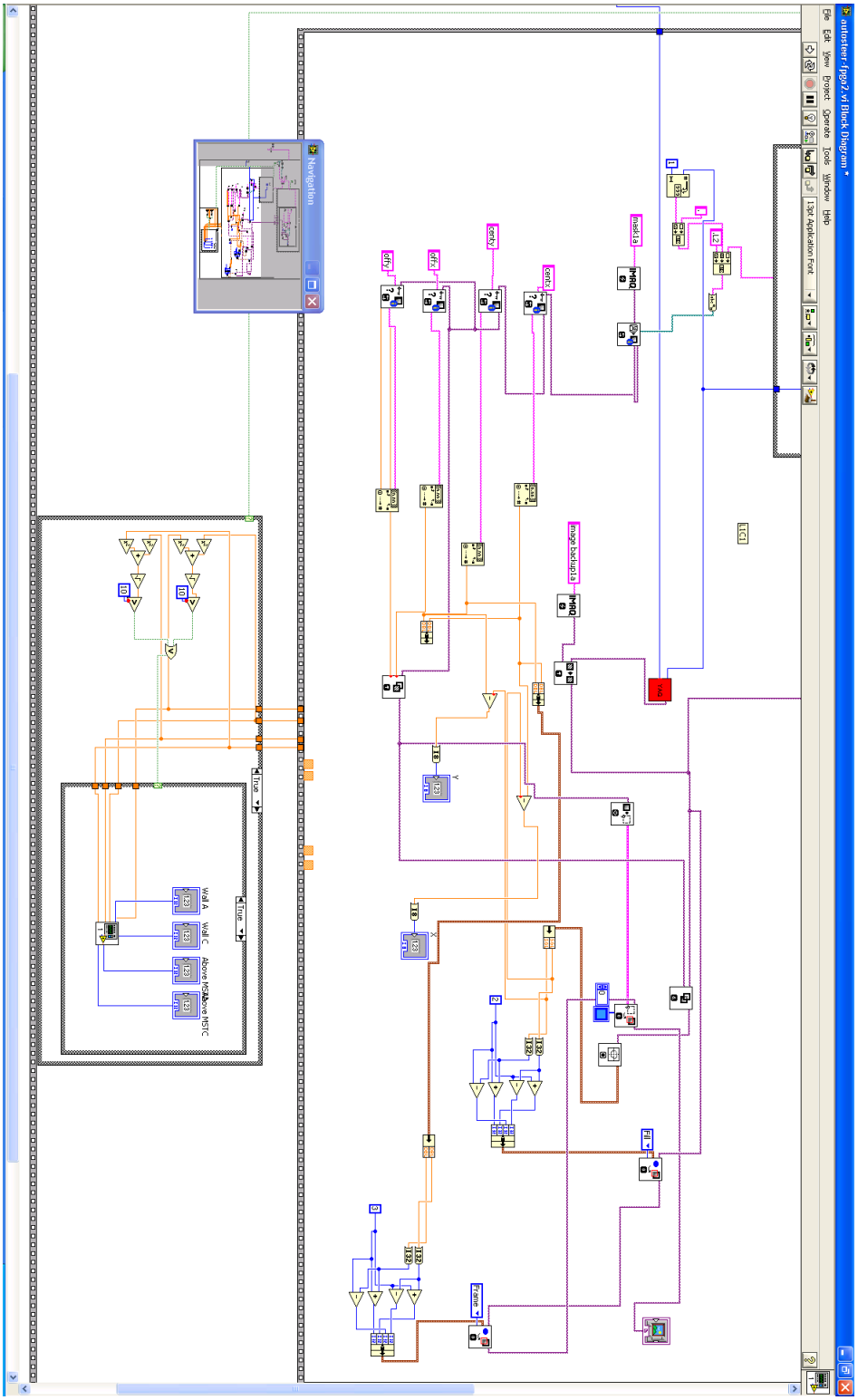

Figure A.4: align-runtime-backbot.

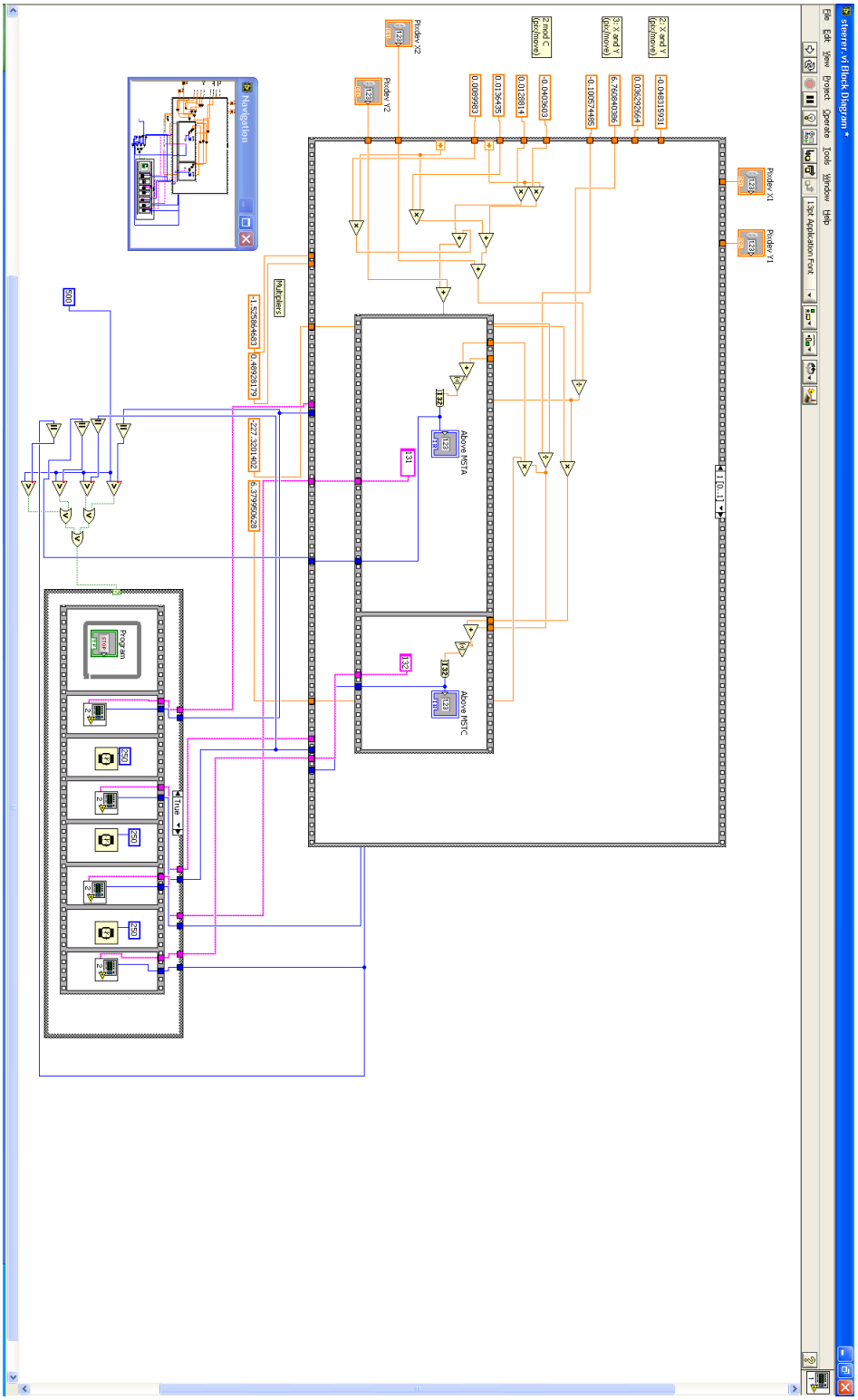

Figure A.5: align-calib-back.

Appendix B

FLPSCS Interface

```
#Initializations

"""This is the code used to manage the FLPSCS power supply, it will be able to
address all of the FLPSCS functions, however it is intended to be imported into
another script, or GUI, and fed inputs from there.Note that all timing is
somewhat order of magnitude, as time.clock goes by the cpu clock which varies
between computers, and also time.sleep has ~1ms accuracy"""

import threading, serial, Queue, time, ctypes
_mainbaton = threading.Lock()
_sendbaton = threading.Lock()
dwps=ctypes.c_ulong(0)
dwpc=ctypes.c_ulong(1)
pb=ctypes.c_byte()
qb=ctypes.pointer(pb)

#modi
#load the device driver for the USB D-i/o board
a=ctypes.windll.LoadLibrary('adsapi32')
pshort=ctypes.c_ushort()
i=ctypes.c_ulong(0)
dhandle=ctypes.c_long()
dhandle2=ctypes.pointer(dhandle)
c=a.DRV_DeviceOpen(i,dhandle2)

#Create lists of commands for programming, checking status, and on/off:
#null at position zero shifts indices chanlist [1] and [2] to match lasers
chanlist=['null']
chanlist.append(range(2,6)) #list of channels for iterating through
chanlist.append(range(6,10))
proglist=['SetPulseWidth','SetPulseDelay']
statuslist=['GetEOC','GetSS','GetGDBFault']
setlist=['GetEOC','GetSS']

caliblist=[[],[3836,3831,3835,3803],[3845,3852,3822,3868],\
          [.082,.08,.11,.1],[.124,.073,.068,.02]]
l1cal = []
l2cal = []
for i in range(4):
  l1cal.append((10-caliblist[3][i])/caliblist[1][i])
  l2cal.append((10-caliblist[4][i])/caliblist[2][i])
```

```
slopes=[[],l1cal,l2cal]
"""Create all of the internal variables we will need.
csq = commands to be sent
brq = bulk responses
srq = status responses
crq = command responses
erq = error responses (currently not implemented)
drq = dgh responses
sq = simmer queue
RT = used to test the run mode toggle
"""
_csq = Queue.Queue()
_brq = Queue.Queue()
_srq = Queue.Queue()
_crq = Queue.Queue()
_erq = Queue.Queue()
_drq = Queue.Queue()
st   = Queue.Queue()
ccst = Queue.Queue()
rt   = Queue.Queue()

#modi
pser = serial.Serial('COM1',19200)
dghser = serial.Serial('COM2',19200)
```

#Main thread

```
class flpscs (threading.Thread):
  """We begin by creating a class for the FLPSCS, which will contain all of our
  I/O functions
  Starting this class will start background inner classes that check for received
  messages, and the PLC trigger this class will also have functions that call
  threads to perform solicited actions (set programming, run mode)"""
  def __init__ (self,status='',settings=''):
    #Connect to the FLPSCS, DGH module, and PLC trigger
    #serial specs: 32 bytes. 19200 bps, 8 data bits, 1 stop bit, no parity.
    #Everything except bps and portname are defaults in pyserial.
    #Automatically opens pser if port is not none

    self.status  = status
    self.settings = settings
    threading.Thread.__init__ ( self )
    print 'FLPSCS backend initialized'
    _reader().start()
    _sorter().start()
    #OUTPUT QUEUE:
    #csq will be the list of commands to be sent, either for queries or programming
    #INPUT QUEUES:
    #brq is the bulk serial received queue, which takes in all serial data from the flpscs
    #srq is the status received queue, sorted from brq, this data will update the UI
    #crq is the command received queue, sorted from brq
    #_sendbaton will be a lock that prevents run-mode toggle while sending commands
    #_mainbaton prevents queueing up of commands when in run mode.

  def run (self):
    _sender().start()
    #_plca().start()
    _plcb(self.status).start()
    #set unused delays to 100us
    for i in range(9,20):
      cs=comsyn('SetPulseWidth','0',i,'',100)
      _csq.put(cs)
    cs=comsyn('SetPulseWidth','0',1,'',100)
```

```
    _csq.put(cs)
    #Set continuous mode
    #modi
    dghser.write('$1DO0400\r\n')
    time.sleep(.1)
    dghser.write('$1SB0B\r\n')
    time.sleep(4)
    #modi
    _status(self.status).start()
    _simmerrun().start()
    _ccsrun().start()
```

#Programming thread

```
class command (threading.Thread):
  def __init__(self, laser,input):
    threading.Thread.__init__(self)
    self.input = input
    self.laser = laser
  def run(self):
    _mainbaton.acquire()
    for i in range(4):
      #creating input: * 0 setoutput voltage nn vvvv
      cs=comsyn('SetOutputVoltage','0',chanlist[self.laser][i], \
      abs((self.input[0][i]/10.-caliblist[self.laser+2][i])/slopes[self.laser][i]))
      _csq.put(cs)
      #create input: * 0 set width/delay nn vvvv
      for ii in range(2):
        if self.input[1][i*2+ii]<450:
          cs=comsyn(proglist[ii],'0',chanlist[self.laser][i],'',self.input[1][i*2+ii])
        else
          cs=comsyn(proglist[ii],'0',chanlist[self.laser][i],'','150')
        _csq.put(cs)

    time.sleep(4)

    #cs=comsyn('GetPulseWidth','0',chanlist[self.laser][i],'',self.input[1][ii])
    #_csq.put(cs)
    #time.sleep(.2)
    tic1=time.clock()
    """
    while time.clock()-tic1<5:
      try:
        c=_crq.get(0)
        ind=c.find('PulseWidth ')
        if ind!=-1:
          print c[ind+11:ind+13]
          print c[ind+14:ind+20]
        else:
          pass
      except:
        pass
    """
    _mainbaton.release()
```

#Status thread

```
class _status (threading.Thread):
#thread which will send "query for status" commands into the _csq, unless in run mode
  def __init__(self, status=''):
    self.status = status
    threading.Thread.__init__(self)
```

```
def run (self):
  while 1:
    _mainbaton.acquire()
    self.tic1=time.clock()
    for i in range(4):
      for ii in statuslist:
        cs=comsyn(ii,'0',chanlist[1][i])
        _csq.put(cs)
        cs=comsyn(ii,'0',chanlist[2][i])
        _csq.put(cs)
    #send out serial commands, but for now just print
    tic1=time.clock()
    _mainbaton.release()

    """
    dlist is default list of channels and colors
    clist is list of channels that receive responses
    slist is list of statuses, to be parsed as faults, and converted into colors
    """
    dlist=[]
    clist=[]
    slist=[]
    clist2=[]
    slist2=[]
    colors=[]
    colors2=[]

    while time.clock()-tic1<8:
      echan=0
      stat=''
      voltage=0
      try:
        time.sleep(.1)
        s=_srq.get(0)
        if s.find('EOC')!=-1:
          ss=s.find('EOC')
          chan=s[ss+10:ss+12]
          stat=s[ss+13:ss+14]
          chan2=int(chan)*2-4
          chan=int(chan)*3-6
          clist.append(chan)
          slist.append(stat)
          clist2.append(chan2)
          slist2.append(stat)
        elif s.find('SS')!=-1:
          ss=s.find('SS')
          chan=s[ss+9:ss+11]
          stat=s[ss+12:ss+13]
          chan2=int(chan)*2-3
          chan=int(chan)*3-5
          clist.append(chan)
          slist.append(stat)
          clist2.append(chan2)
          slist2.append(stat)
        elif s.find('GDB')!=-1:
          ss=s.find('GDB')
          chan=s[ss+15:ss+17]
          stat=s[ss+18:ss+19]
          chan=int(chan)*3-4
          clist.append(chan)
          slist.append(stat)
        else:
          print 'oops'
      except:
```

```
            pass

        #create an array of all Yellow status colors the length of our channel list
        temp=[]
        for i in range(24):
          temp.append('Y')
        dlist.append(temp)
        temp=[]
        for i in range(24):
          temp.append(i)
        dlist.append(temp)
        temp=[]
        for i in range(16):
          temp.append('Y')
        dlist.append(temp)
        temp=[]
        for i in range(16):
          temp.append(i)
        dlist.append(temp)

        #if the status is Simmering, No fault, or Done charging, indicators are green
        #else the indicators are red, if the response is timed out, it will be yellow

        for i in range(len(clist2)):
          if slist2[i]=='S':
            colors2.append('G')
          elif slist2[i]=='N':
            colors2.append('R')
          elif slist2[i]=='D':
            colors2.append('G')
          elif slist2[i]=='C':
            colors2.append('R')
          else:
            print 'oops'

        for i in range(len(clist)):
          if slist[i]=='S':
            colors.append('G')
          elif slist[i]=='N':
            if clist[i]%3==2:
              colors.append('G')
            elif clist[i]%3==1:
              colors.append('R')
          elif slist[i]=='D':
            colors.append('G')
          elif slist[i]=='C':
            colors.append('R')
          elif slist[i]=='F':
            colors.append('R')
          else:
            print 'oops'

        self.status(dlist[1],dlist[0],dlist[3],dlist[2])
        self.status(clist,colors,clist2,colors2)

        time.sleep(4)
```

#Manual toggles

```
#manual toggle of fake PLC trigger
class fakerun():
  def __init__(self):
    self.indie=0
```

```
  def go(self):
    if self.indie==0:
      self.indie=1
      rt.put(1)
    else:
      rt.put(0)
      self.indie=0

#toggle of CCS part 1
class ccstog():
  def __init__(self):
    self.ccs=['','','off','off','off','off','off','off','off','off']
  def go(self,ccsnum):
    if self.ccs[ccsnum]=='off':
      self.ccs[ccsnum]='on'
      ccst.put(ccsnum)
    elif self.ccs[ccsnum]=='on':
      self.ccs[ccsnum]='off'
      ccst.put(ccsnum*(-1))
    else:
      print 'holy crap CCS exception'

#toggle of CCS part 2
class _ccsrun(threading.Thread):
  def run(self):
    while 1:
      ccsnum=ccst.get()
      _mainbaton.acquire()

      if ccsnum>0:
        a=comsyn ('EnableCCS','0',ccsnum)
        _csq.put(a)
      else:
        a=comsyn ('DisableCCS','0',abs(ccsnum))
        _csq.put(a)
      time.sleep(.3)
      _mainbaton.release()

#Toggle of simmer part 1
class simmertog():
  def __init__(self):
    self.sim=['','','off','off','off','off','off','off','off','off']
  def go(self,simmernum):
    if self.sim[simmernum]=='off':
      self.sim[simmernum]='on'
      st.put(simmernum)
    elif self.sim[simmernum]=='on':
      self.sim[simmernum]='off'
      st.put(simmernum*(-1))
    else:
      print 'holy crap simmer exception'

#Toggle of simmer part 2
class _simmerrun(threading.Thread):
  def run(self):
    while 1:
      simnum=st.get()
      _mainbaton.acquire()

      if simnum>0:
        a=comsyn ('EnableSS','0',simnum)
        _csq.put(a)
        print 'enabling',simnum
      else:
```

```
        a=comsyn ('DisableSS','0',abs(simnum))
        _csq.put(a)
        print 'disabling',simnum
      time.sleep(.3)
      _mainbaton.release()

#Background threads

class _sender (threading.Thread):
  def run(self):
    #should send serial of qt.get, for now print
    while 1:
      cs=_csq.get()
      #_mainbaton.acquire()
      _sendbaton.acquire()
      while 1:
        #modi
        pser.write(cs)
        print cs
        """
        if cs.find('SetPulseDelay 05')!=-1:
          _sendbaton.release()
          _mainbaton.release()
          break
        elif cs.find('SetPulseDelay 09')!=-1:
          _sendbaton.release()
          _mainbaton.release()
          break
        elif cs.find('GetGDBFault 09')!=-1:
          _sendbaton.release()
          _mainbaton.release()
          break
        elif cs.find('SetPulseWidth 19')!=-1:
          _sendbaton.release()
          _mainbaton.release()
          break
        else:
          cs=_csq.get()
        """
        try:
          cs=_csq.get(0)
        except:
          _sendbaton.release()
          #_mainbaton.release()
          break
        time.sleep(.3)

#Thread which will continuously check for a PLC trigger, and upon receiving one,
#wait for other processes to finish before locking them out
class _plcb (threading.Thread):
  def __init__(self,status):
    threading.Thread.__init__(self)
    self.flag=0
    self.status = status
  #checks <interface> for PLc trigger
  def run(self):
    while 1:
      time.sleep(.20)
      try:
        a=rt.get(0)
        if a==1:
          #print 'Received runmode request'
          if self.flag==0:
```

```
            self.flag=1
            _mainbaton.acquire()
            _sendbaton.acquire()
            print 'run mode'
            #modi
            dghser.write('$1SB09\r\n')
          elif flag==1:
            pass
          #this portion acquires the _csq lock so that status and command button
          #cannot access it
          #the brq join will make sure the bulk response queue is empty,
          #for now we assume this starts filling up while _csq is being filled.
          #DO: send serial command for run mode
        #elif trigger off
        elif a==0:
          if self.flag==1:
            print 'off runmode'
            self.flag=0
            #modi
            dghser.write('$1CB09\r\n')
            time.sleep(.2)
            _mainbaton.release()
            _sendbaton.release()
          elif flag==0:
            pass
      except:
        pass

class _reader(threading.Thread):
  """Class which will read all serial communications into the _brq, and then
  sort into various solicited and unsolicited lists"""
  def init(self):
    threading.Thread.__init__(self)
    self.started=0
  def run(self):
    while 1:
      #modi
      #pass
      a=pser.readline()
      print a
      """
      if a.find('Power On Init Complete')!=-1:
        if self.started==0:
          self.started=1
          _status().start()
      else:"""
      #modi
      _brq.put(a)

class _sorter(threading.Thread):
  def run(self):
    while 1:
      b=_brq.get()
      if b.find('Set')!=-1:
        _crq.put(b)
      elif b.find('PulseWidth')!=-1:
        _crq.put(b)
      elif b.find('MonitorVoltage')!=-1:
        _srq.put(b)
      elif b.find('Status')!=-1:
        _srq.put(b)
      else:
        _erq.put(b)
```

```
  #function that puts command inputs into proper syntax for FLPSCS communication
def comsyn (cmd,id='0',nn='',vv='',ww=''):
  #takes in command, device ID, device number/status byte etc,
  #and voltage for PulseWidthSet only, ID defaults to zero, nn and vv are optional
  #note that device ID comes after cmd, since default values are annoying in python
  if vv!='':
    vv ="%.4d" % vv
  if nn!='':
    nn ="%.2d" % nn
  if ww!='':
    ww ="%.6d" % ww
  a = '* '+str(id)+' '+str(cmd)+' '+str(nn)+' '+str(vv)+str(ww)
  #fill out whitespace
  a = a+(29-len(a))*' '+'$'+'\r'+'\n'
  return a
```

Appendix C

Mode Analysis

We can express the magnetic field measured by the toroidal array coils as:

$$B(\phi) = \sum_{n=0}^{\infty} a_n sin(n\phi) + b_n cos(n\phi)$$

where:

$$a_n = \frac{1}{\pi} \int_{\phi=0}^{2\pi} B(\phi) sin(n\phi) d\phi$$

$$b_n = \frac{1}{\pi} \int_{\phi=0}^{2\pi} B(\phi) cos(n\phi) d\phi$$

or alternatively, we can let:

$$B(\phi) = \sum_{n=0}^{\infty} c_n cos(n\phi - \delta_{cn}) \tag{C.1a}$$

$$= \sum_{n=0}^{\infty} c_n[cos(n\phi)cos(\delta_{cn}) + sin(n\phi)sin(\delta_{cn})]$$

$$= \sum_{n=0}^{\infty}[c_n cos(\delta_{cn})]cos(n\phi) + [c_n sin(\delta_{cn})]sin(n\phi)$$

$$= \sum_{n=0}^{\infty} b_n cos(n\phi) + a_n sin(n\phi)$$

(C.1b)

Giving:

$$\begin{aligned} c_n cos(\delta_{cn}) &= b_n \\ c_n sin(\delta_{cn}) &= a_n \\ tan(\delta_{cn}) &= \frac{a_n}{b_n} \\ and c_n^2(cos^2(\delta_{cn}) + sin^2(\delta_{cn})) & \\ c_n^2 &= a_n^2 + b_n^2 \\ c_n &= \sqrt{a_n^2 + b_n^2} \qquad \text{(C.2)} \\ and & \\ \delta_n &= arctan(\frac{a_n}{b_n}) \qquad \text{(C.3)} \end{aligned}$$

C.0.3 Error Fields

We can add Fourier decomposed stationary error fields x_n, y_n to our coil measurements as follows:

$$B(\phi,t) = \sum a_n(t)sin(n\phi) + b_n(t)cos(n\phi) + \sum x_n sin(n\phi) + y_n cos(n\phi) \tag{C.4}$$

which can be expressed as (supressing the (t) terms):

$$B(\phi) = \sum (a_n + x_n)sin(n\phi) + (b_n + y_n)cos(n\phi) \tag{C.5}$$

or

$$B(\phi) = \sum C_n cos(n\phi - \delta_c) + Z_n cos(n\phi - \delta_z) \tag{C.6}$$

Here, we have added a time-dependence to the $a_n s$ and $b_n s$ to distinguish them from the stationary error fields. We may repeat the same procedure outlined above to convert this signal into a total mode amplitude, cosine, and phase $B_n cos(n\phi - \delta_B)$. If using equation A.5:

$$\begin{aligned} B(\phi) &= B_n cos(n\phi - \delta_B) \\ &= B_n[cos(n\phi)cos(delta_B) + sin(n\phi)sin(\delta_B)] \\ &= [B_n cos(\delta_B)]cos(n\phi) + [B_n sin(\delta_B)]sin(n\phi) \\ B(\phi) &= \sum (a_n + x_n)sin(n\phi) + (b_n + y_n)cos(n\phi) \end{aligned}$$

As with before, this results in (inserting time dependence back in):

$$\begin{aligned} B_n &= \sqrt{(a_n(t)+x_n)^2+(b_n(t)+y_n)^2} \\ B_n &= \sqrt{(a_n^2(t)+b_n^2(t))+(x_n^2+y_n)^2+2(a_n(t)x_n+b_n(t)y_n)} \end{aligned} \tag{C.7}$$

and

$$\delta_B = arctan\frac{(a_n(t)+x_n)}{(b_n(t)+y_n)} \tag{C.8}$$

We may derive this using equation A.6 as well:

$$\begin{aligned} B(\phi) &= \sum C_n cos(n\phi-\delta_c)+Z_n cos(n\phi-\delta_z) \\ &= C_n[cos(n\phi)cos(\delta_c)+sin(n\phi)sin(\delta_c)] \\ &\quad +Z_n[cos(n\phi)cos(\delta_z)+sin(n\phi)sin(\delta_z)] \\ &= [C_n cos(\delta_C)+Z_n cos(\delta_z)]cos(n\phi) \\ &\quad +[C_n sin(\delta_C)+Z_n sin(\delta_Z)]sin(n\phi) \end{aligned}$$

Similar to before, from $B_n cos(n\phi - \delta_B)$ this results in:

$$\begin{aligned}
B_n cos(\delta_B) &= C_n cos(\delta_C) + Z_n cos(\delta_z) \\
B_n sin(\delta_B) &= C_n sin(\delta_C) + Z_n sin(\delta_z) \\
B_n^2 &= C_n^2 cos^2(\delta_C) + Z_n^2 cos^2(\delta_z) \\
&\quad + C_n^2 sin^2(\delta_C) + Z_n^2 sin^2(\delta_z) \\
&\quad + 2C_n Z_n [sin(\delta_C) sin(\delta_Z) + cos(\delta_C) cos(\delta_Z)]
\end{aligned}$$

giving

$$B_n^2 = C_n^2 + Z_n^2 + 2C_n Z_n cos(\delta_Z - \delta_C) \quad \text{(C.9)}$$

$$\delta_B = arctan(\frac{C_n sin(\delta_C) + Z_n sin(\delta_z)}{C_n cos(\delta_C) + Z_n cos(\delta_z)}) \quad \text{(C.10)}$$

The above result is identical to the formulation Darren Craig used in the mode analysis write-up. Regardless of the flavor of expressing the stationary error fields, it is clear to see the impact of the error fields on not only the mode amplitude, but also the phase of the mode.

Appendix D

MSTFit Misc.

Note: MSTFit solves the Grad-Shafranov equation by solving for the parallel current density:

$$J_\phi = \frac{2\pi F F'}{\mu_0 R} + 2\pi R P'$$

It does so by iteratively creating a set of flux surfaces, solving for F, and taking T_e and n_e data to create a P' profile.

D.1 Transport Calculations

$$\begin{aligned} P_\Omega &= \int \mathbf{E}\cdot\mathbf{J}\, dV && \text{(D.1)} \\ &= \int (\eta\mathbf{J}\cdot\mathbf{J})\, dV && \text{(D.2)} \\ &= \int \eta J^2\, dV && \text{(D.3)} \\ & && \text{(D.4)} \end{aligned}$$

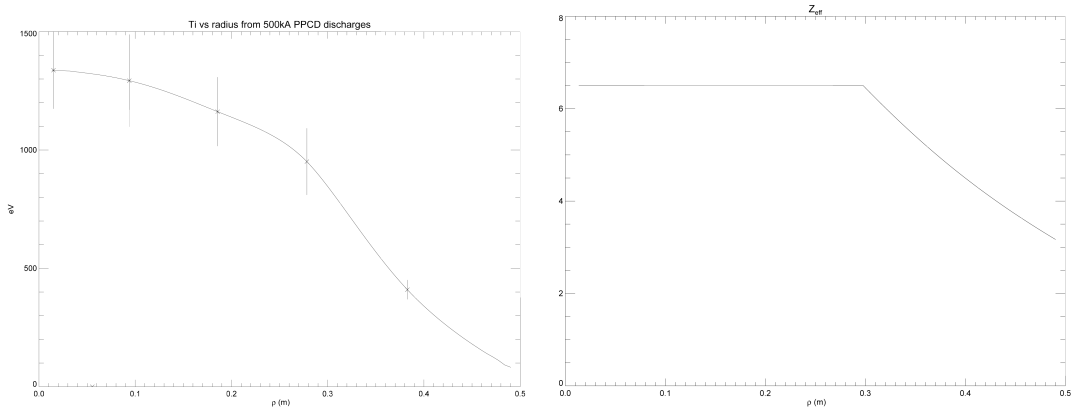

Figure D.1: T_i profile used for fitting and transport calculations.

where η is the neoclassical Spitzer restivity. As η depends on the Z-effective profile, an assumed profile for Z-effective that is flat in the core and decreasing at the edge has been applied to all the equilibrium reconstructions. Z_{eff} and T_i profile[28]: (D.1)

Loss terms include electron-to-ion collision losses:

$$P_{etoi} = 3\frac{m_e}{m_i}\frac{n_e(T_e - T_i)}{\tau_{coll}}$$

where τ_{coll} is the classical electron-ion coulomb collision time, as well as radiative loss which is empirically scaled to P_Ω

$$P_{rad}(r) = P_\Omega C_{rad}\left(\frac{r}{a}\right)^8$$

where is a coefficient C_{rad} is based off data from the bolometer.

By combining all of these terms, we can arrive at the local electron heat flux Q_e:

$$Q_e = \frac{P_\Omega - \dot{W}_e - P_{etoi} - P_{rad}}{4\pi^2 r\rho}$$

Dividing this value by the density and temperature gradient allows us to arrive at a thermal transport coefficient χ_e:

$$\chi_e = \frac{-Q_e}{n_e \nabla T_e}$$

MSTFit also generates a q-profile from the reconstructed equilibrium, using the definition $q = \frac{rB_z}{RB_\phi}$.

Appendix E

Calculating χ_{st}

$$\chi_e = v_{Te} D_m \tag{E.1}$$

$$D_m = L_{ac} \frac{< \tilde{B}_r^2 >}{B_0^2} \tag{E.2}$$

$$L_{ac} \approx \frac{\pi}{\Delta k_{\|}} \tag{E.3}$$

$$\begin{aligned} k_{\|} &= \frac{\boldsymbol{k} \cdot \boldsymbol{B}}{|B|} \\ &= \frac{1}{|B|} \left(\frac{m}{r} B_\theta + \frac{n}{R} B_\phi \right) \\ &= \frac{m B_\theta}{r|B|} + \frac{n B_\phi}{R|B|} \\ &= \frac{n}{R} \left(\frac{m}{n} \frac{R B_\theta}{r|B|} + \frac{B_\phi}{|B|} \right) \end{aligned} \tag{E.4}$$

where

$$|B| = \sqrt{B_\theta^2 + B_\phi^2}$$

and

$$\Delta k_{||} = \Delta r \left| \frac{\partial k_{||}}{\partial r} \right|_{rs}$$

which gives

$$\Delta k_{||} = \frac{n}{R} \Delta r \left| \frac{\partial}{\partial r} \left[\frac{m}{n} \frac{RB_\theta}{r|B|} + \frac{B_\phi}{|B|} \right] \right|_{rs}$$

Appendix F

Island width calculation

The island width was calculated using the equation:

$$w_{mn} = 4\sqrt{\frac{r_{mn}}{n}\frac{\tilde{b}_{mn}}{B_\theta}\frac{1}{|q\prime_{mn}|}} \tag{F.1}$$

The modes were assumed to be dominated by m=1 fluctuations. The values for r_n, B_θ, and $|q\prime_n|$ were taken from the MSTFit generated q-profile, its derivative, and equilibrium field profile, and the local $\tilde{b}_{r,mn}$ was obtained from reprocessed edge coil data, scaled using the same eigenfunctions as with the $\chi_{e,stoch}$ calculations. The

Chirikov stochasticity parameter was calculated using the equation:

$$s = \frac{1}{2}\frac{w_{n+1} + w_n}{r_{n+1} - r_n} \tag{F.2}$$

For the representative data point, the stochasticity parameter between islands was:

n and $n+1$	s
8 and 9	1.22
9 and 10	1.19
10 and 11	1.22
11 and 12	1.05
12 and 13	0.99
13 and 14	1.09
14 and 15	1.06
15 and 16	2.27