

Magnetic Mode Analysis in the ZaP Flow-Stabilized Z-pinch Experiment

by

Justin E. Bright

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

Master of Science in Aeronautics and Astronautics

University of Washington

2002

Program Authorized to Offer Degree: Aeronautics and Astronautics

University of Washington

Graduate School

This is to certify that I have examined this copy of a master's thesis by

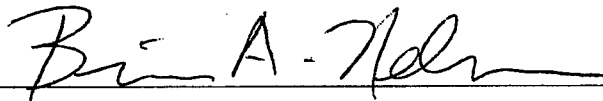
Justin E. Bright

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:



Uri Shumlak, Chair



Brian A. Nelson

Date: 18 Mar 02

Master's Thesis

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purposes or by any means shall not be allowed without my written permission.

Signature Justin Brant

Date 18 MAR 02

**THE VIEWS EXPRESSED IN THIS ARTICLE
ARE THOSE OF THE AUTHOR AND DO NOT
REFLECT THE OFFICIAL POLICY OR
POSITION OF THE UNITED STATES,
DEPARTMENT OF DEFENSE, OR THE U.S.
GOVERNMENT**

University of Washington

Abstract

Magnetic Mode Analysis in the ZaP Flow-Stabilized Z-pinch Experiment

by Justin E. Bright

Chairperson of the Supervisory Committee:
Assistant Professor Uri Shumlak
Department of Aeronautics and Astronautics

The ZaP Flow-Stabilized Z-pinch experiment at the University of Washington studies the stabilizing effects of sheared flow on a Z-pinch. An internal eight-probe azimuthal array is used to measure the magnetic field. Fourier analysis of the magnetic field yields information on the structure of the plasma. The $m=1$ mode is related to the radial offset of the plasma, while the $m=2$ mode is related to the elongation of the plasma. Using this information, a non-linear fitting routine was developed to determine the radial position and structure of the plasma. The plasma is simulated as two independent current-carrying filaments. To improve speed and accuracy, a neural net was developed. The neural net attempts to simulate the non-linear fit as closely as possible while performing at speeds up to twenty times faster. Unlike a non-linear fitting routine, the neural net requires no initial guesses, and thus runs independently of the user. Testing of both the neural net and the non-linear fitting routine show that both techniques compute results which fit the experimental $m=1$ data well. Both techniques display similar error, although the neural net is less accurate during periods in which the plasma is unstable. Further work is being done to improve the calculation of the separation between the two filaments.

TABLE OF CONTENTS

List of figures	iv
Introduction	1
Chapter 1: The Z-pinch and the ZaP Experiment	3
Force Balance	3
Classical Instabilities	4
Sheared Flow Stabilization	6
The ZaP Experiment.....	8
Magnetic Probes in the ZaP Experiment.....	9
Chapter 2: Calibration of Magnetic Probes.....	12
Motivation	12
Internal Magnetic Chord Probes on the ZaP Experiment	12
Calibration.....	14
Results of Calibration.....	18
Chapter 3: Fourier Representation of the Magnetic Field.....	27
Representation of Functions with Fourier Series	27
Representing the Magnetic Field.....	27
Magnetic Field Modes.....	29
Normalization.....	32
Chapter 4: Determining Plasma Properties from Magnetic Mode Data	34
Computer Simulation	34

Flux Conservation	37
The $m=1$ Mode.....	38
The $m=2$ Mode.....	39
Chapter 5: The Non-Linear Fitting Routine.....	41
Layout of the Non-Linear Fitting Computer Program	41
Geometries	42
Testing Different Geometries.....	46
The CURVEFIT Routine	49
Initial Results.....	52
Discussion of Results	55
Chapter 6: The Neural Network.....	59
Background of the Neural Network	59
The Neuron.....	60
Neural Network.....	63
Feed-Forward Backpropagation Multi-Layered Perceptrons.....	65
Application to the Magnetic Probe Problem.....	67
Training and Implementing the Neural Network Using MATLAB.....	67
Training Techniques.....	70
Final Structure of the Magnetic Probe Neural Net.....	72
Chapter 7: Results of the Neural Net	75
Comparison of the $m=1$ Mode	75
Error Comparisons	88

Conclusions	95
Chapter 8: Summary and Conclusions	97
Summary	97
Conclusions	98
Future Work	99
References	104
Appendix A: Chord Probe Calibration Code	105
Appendix B: Mode Data Simulation Code	108
Appendix C: Non-Linear Fit Routine Code	113
Appendix D: Neural Net Code	119

LIST OF FIGURES

<i>Number</i>	<i>Page</i>
Figure 1.1: Z-pinch equilibrium	4
Figure 1.2: $m=0$ sausage mode	5
Figure 1.3: $m=1$ kink mode.....	6
Figure 1.4: Normalized velocity shear	7
Figure 1.5: ZaP experimental layout.....	9
Figure 1.6: Design and dimensions of the magnetic probe cups.....	11
Figure 1.7: Locations of the axial array and two azimuthal arrays	11
Figure 2.1: Top view of the internal chord probes.....	13
Figure 2.2: LRC circuit for calibration of the chord probes.....	15
Figure 2.3: Top and side views of the calibration setup.....	16
Figure 2.4: Magnetic field in the Helmholtz coil	19
Figure 2.5: The signal from a magnetic probe (surface normal parallel).....	20
Figure 2.6: The signal from a magnetic probe (surface normal perpendicular).....	21
Figure 2.7: Calibration fitting for the first orientation	23
Figure 2.8: Calibration for the second orientation	24
Figure 2.9: Calibration for the third orientation	25
Figure 2.10: Final results from the calibration of a chord probe.....	26
Figure 3.1: The azimuthal probe array layout	29
Figure 3.2: The $m=0$ representation.....	31
Figure 3.3: The $m=1$ mode representation	31

Figure 3.4: The $m=2$ mode representation	32
Figure 3.5: Typical magnetic mode data for a Zap shot.....	33
Figure 4.1: The computer model of the plasma	35
Figure 4.2: Normalized $m=1$ mode versus radial offset.....	38
Figure 4.3: The $m=2$ mode versus offset of centroid.....	40
Figure 5.1: Geometry 1, based on mode structure	43
Figure 5.2: Geometry 2, based on polar coordinates.....	44
Figure 5.3: Predictor, with one current filament	45
Figure 5.4: Error comparison for geometry 1 and geometry 2 (synthetic data).....	47
Figure 5.5: Error comparison for geometry 1 and geometry 2 (experimental data)	48
Figure 5.6: Comparison of fitting routine to $m=1$ mode (Pulse 726025)	53
Figure 5.7: Comparison of fitting routine to $m=1$ mode (Pulse 726027)	53
Figure 5.8: Time evolution of the plasma centroid (Pulse 726025).....	54
Figure 5.9: Evolution of the separation of the plasma filaments	55
Figure 5.10: A negative field and a positive field at the same time.....	57
Figure 6.1: Mathematical model of a single neuron.....	60
Figure 6.2: Threshold function at $x=3$	61
Figure 6.3: The 'tansig' function.....	63
Figure 6.4: The weights for each input are unique for every neuron	64
Figure 6.5: The layout of a FFBP network.....	66
Figure 6.6: Typical training progress while training the neural net	72
Figure 6.7: The final net structure.....	73

Figure 7.1: Neural net results, Pulse 726025 (8.0 kV).....	76
Figure 7.2: Neural net results, Pulse 726027 (8.0 kV).....	77
Figure 7.3: Neural net Results, Pulse 10201015 (8.0 kV).....	78
Figure 7.4: Neural net results, Pulse 10306007 (6.0 kV).....	79
Figure 7.5: Neural net results, Pulse 10417007 (4.0 kV).....	80
Figure 7.6: Neural net results for quiescent period, Pulse 10201015.	82
Figure 7.7: Comparison of neural net to non-linear fitting routine, Pulse 726025	83
Figure 7.8: Comparison of neural net to non-linear fitting routine, Pulse 726027	84
Figure 7.9: Comparison of neural net to non-linear fitting routine, Pulse 10201015.	85
Figure 7.10: Comparison of neural net to non-linear fitting routine, Pulse 10306007	86
Figure 7.11: Comparison of neural net to non-linear fitting routine, Pulse 10417007	87
Figure 7.12: Error plot for Pulse 726025	89
Figure 7.13: Error plot for Pulse 726027.	90
Figure 7.14: Error plot for Pulse 10201015	91
Figure 7.15: Error plot for Pulse 10306007	92
Figure 7.16: Error plot for Pulse 10417007	93
Figure 7.17: Separation of the plasma filaments, calculated by the neural net.....	95
Figure 8.1: Change of magnetic field with separation of filaments	100
Figure 8.2: Calculated separation on synthetic data.....	101

ACKNOWLEDGEMENTS

I would like to thank all the people who helped make my work a success. First, I wish to thank Prof. Uri Shumlak and Prof. Brian Nelson for guiding my research and ensuring that I understood clearly the problems and principles at hand. I would especially like to thank Raymond Golingo for working with me on my programming skills, and helping me with my research on a day to day basis. Raymond was never too busy to stop and answer a question. I would also like to thank Stuart Jackson for his work done on the ZaP experiment, David Okada for his construction of the internal chord probes, and William Litsch for his help with neural networks.

INTRODUCTION

In the search for controlled thermonuclear fusion, scientists have studied many different plasma configurations. To reach a break-even point in energy generation, the plasma configuration must satisfy the well-known Lawson criterion: $n\tau \geq 10^{20} \text{ m}^{-3} \text{ - sec}$ (for a D-T reaction)¹. One of these configurations is the Z-pinch. The Z-pinch offers very high densities, a key requirement for sustained fusion. One of the primary drawbacks of the Z-pinch, however, is a short stability time.

Linear stability analysis has shown it is possible to stabilize a Z-pinch over longer time periods with a sheared axial flow². A sheared axial flow will prevent instabilities from forming, which typically destroy the Z-pinch equilibrium after very short time periods. The ZaP experiment studies the effects of sheared axial flow on a Z-pinch plasma. A Z-pinch of 50 cm has been created, with a radius of roughly 1.0 cm. By forming the pinch with a sheared axial flow, the ZaP Z-pinch has displayed stability times over 700 times greater than the theoretical instability growth time³. Stable time periods such as those demonstrated by ZaP suggest the Z-pinch holds promise for future work in the fusion energy field.

While fusion energy applications may rest in the more distant future, fusion space propulsion applications are possible in the near-term. The Z-pinch concept can be directly applied to space propulsion. A stable Z-pinch would create a high-energy axial flow, which would serve as a rocket propulsion device. The ZaP experiment could eventually be adapted to a prototype fusion rocket device. Initial calculations indicate specific impulses in the range of 10^6 s . Current space propulsion rockets are limited to

either high thrust and low specific impulse, or high specific impulse and low thrust. A Z-pinch fusion thruster would offer the best of both worlds.

The ZaP experiment promises to have applications in many fields. At this time, however, it remains primarily a scientific experiment aimed at better understanding plasmas, Z-pinches and the associated stability issues. Thus, a major portion of the time and energy spent on ZaP is dedicated to diagnostics. The ZaP experiment contains a large number of magnetic probes, including axial, azimuthal and radial. These probes help us better understand the evolution of the magnetic fields throughout the lifetime of the plasma. Along with the other diagnostics, the magnetic probes increase our understanding of the plasma and how we may one day be able to control it.

CHAPTER 1: The Z-pinch and the ZaP Experiment

Force Balance

In a classic Z-pinch, the plasma is configured so that there is only an axial current, J_z .

Thus, the magnetic field in a Z-pinch is purely azimuthal (B_θ). The azimuthal magnetic field self-contains the current, thus creating an equilibrium. The pressure balance for a Z-pinch is as follows⁴:

$$\frac{\partial}{\partial r} \left(p + \frac{B_\theta^2}{2\mu_0} \right) = -\frac{B_\theta^2}{\mu_0 r}. \quad (1)$$

We assume there is a uniform current profile and that the current disappears at the edges of the Z-pinch.

$$\begin{aligned} J_z(r) &= J_{z0} & r < a \\ J_z(r) &= 0 & r > a \end{aligned}$$

Since the current is uniform, we can write

$$B_\theta = \frac{B_{\theta a} r}{a}. \quad (2)$$

Integrating Equation 1, we see that

$$p(r) = p_0 - \frac{B_{\theta a}^2 r^2}{\mu_0 a^2}. \quad (3)$$

Since $p(a) = 0$, we can determine

$$p_0 = \frac{B_{\theta a}^2}{\mu_0} = \frac{\mu_0 I^2}{4\pi^2 a^2}. \quad (4)$$

The above equations outline the equilibrium pressure balance that describes a Z-pinch.

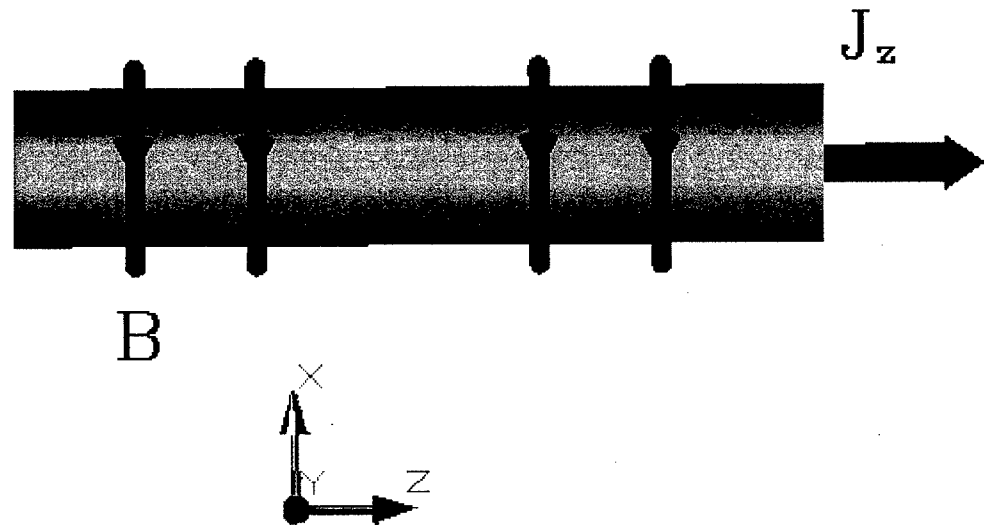


Figure 1.1: The Z-pinch equilibrium, with the magnetic field purely in the theta direction and the current in the axial direction.

Classical Instabilities

Although the theoretical equations describe an equilibrium configuration, real Z-pinchs are susceptible to several instabilities. Imperfections in the current profile or magnetic field can introduce these instabilities, which often lead to the destruction of the equilibrium and the Z-pinch plasma.

The $m=0$ instability, often called the “sausage mode,” occurs when the radius of the pinch (denoted as “ a ” in the above equations) decreases at one point. The magnetic field at this point will increase, since the magnetic field B_θ is inversely proportional to radius (Equation 2). As the magnetic field increases, the magnetic pressure also increases, and the radius of the current at that point further decreases. This leads to an eventual collapse of the current at that point.

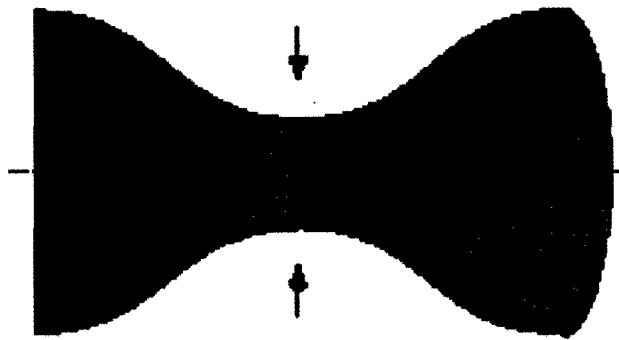


Figure 1.2: The $m=0$ sausage mode instability starts with a decrease in the pinch radius.

Another instability that occurs in Z-pinches is the $m=1$ mode, called the “kink mode.” This mode occurs when there is a bend in the current column. The magnetic pressure will be stronger at the inside of the bend than at the outside, thus aggravating the kink even further. Like the sausage mode, this instability will eventually destroy a Z-pinch, breaking the current flow and disrupting the self-containing magnetic field.

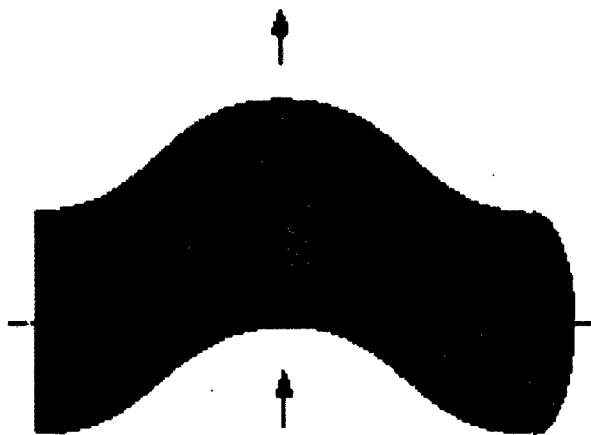


Figure 1.3: The $m=1$ kink mode instability begins when the pinch curves, or “kinks”.

Sheared Flow Stabilization

The instabilities described above appear to limit the usefulness of Z-pinchs, since such instabilities often arise very rapidly—on the order of the Alfvén Velocity. Alfvén Velocity is the velocity at which perturbations in the magnetic field propagate down the field lines, and is defined as

$$V_A = \frac{B}{\sqrt{\mu_0 \rho}}, \quad (5)$$

where B is the magnetic field and ρ is the mass density⁵. Linear stability analysis has predicted, however, that a sheared axial flow will stabilize these instabilities indefinitely⁶. In general, the sheared flow along the edges of the Z-pinch column will force instabilities (such as kinks) to move at different velocities at different radii. If the shear ($\frac{dv_z}{dr}$) is

great enough, the mode phases mix and the instabilities cannot move as a coherent whole.

This prevents the instabilities from destroying the Z-pinch. The following figure shows the stable/unstable region for a Z-pinch plasma based on the sheared axial flow. In the

figure, $v' = \frac{dv_z}{dr}$, v_A = Alfvén Velocity, r_w = wall radius, a = pinch radius and $k = \frac{2\pi}{\lambda}$

(where λ is wavelength). Notice that as the pinch radius approaches the wall radius, the wall prevents instabilities from forming. We typically want pinches that do not make contact with the wall, however, so we are interested in the right-hand portion of the graph.

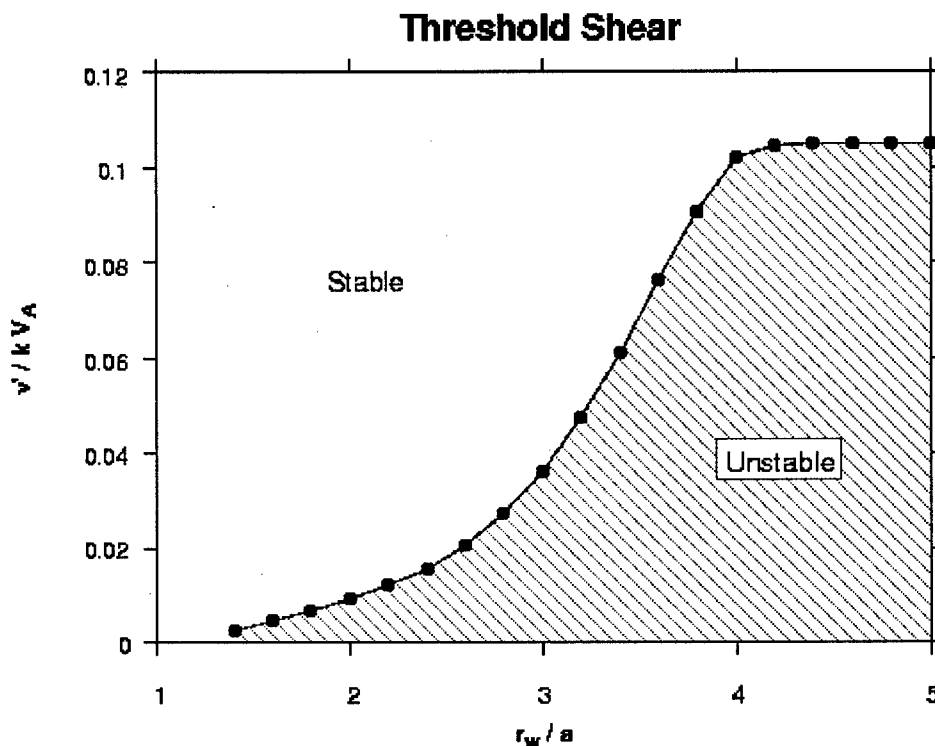


Figure 1.4: The normalized velocity shear must be above a particular threshold to stabilize the instabilities.

The ZaP Experiment

The ZaP experiment (see Figure 1.5) is studying the effects of sheared axial flow on the stabilization of Z-pinches. The experiment creates a 50 cm plasma Z-pinch approximately 1 cm in radius. The ZaP experiment typically uses hydrogen gas to run, although a 50% methane dopant fill has been used to study carbon lines and other plasma features.

The experiment uses a high voltage differential to create the ionized plasma. Shots run from 1 kV up to 9 kV, where 9 kV is the normal operating value. To create the plasma, neutral gas is injected into a vacuum chamber, and the potential (supplied by a capacitor bank) is applied to the inner electrode. This ionizes the gas, and causes a current to run between the inner and outer electrode. The $J \times B$ force then pushes the current down the inner electrode. Eventually, the current sheet collapses on the z-axis, creating a pinch with a sheared axial flow. Current still moving along the electrodes due to the $J \times B$ force supplements the sheared axial flow.

The ZaP experiment is primarily a science experiment, and therefore much effort is dedicated to diagnostics. The ZaP experiment hosts a variety of diagnostics, including a two chord interferometer to measure electron density, a holography system to measure density profiles, a 20-channel ICCD spectrometer to measure the velocity of impurities, and an Imacon fast framing camera to take visual pictures. The experiment also utilizes several magnetic field probes, including an axial array to measure the current sheet location, azimuthal arrays to measure mode structures and a recently installed chord

array. This thesis is primarily concerned with the azimuthal arrays and the new chord arrays.

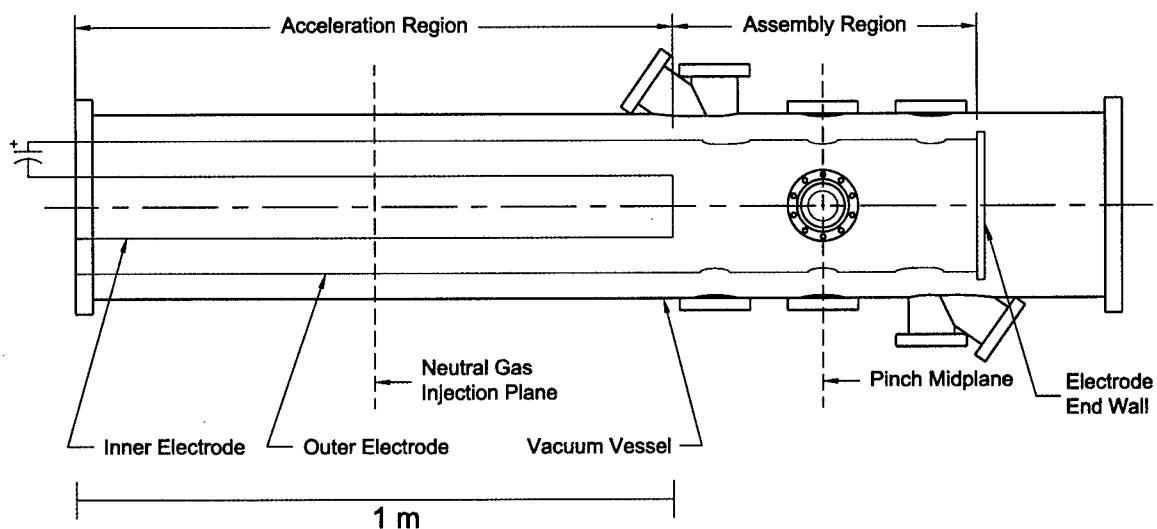


Figure 1.5: The ZaP experimental layout, with the acceleration region and the assembly region. The pinch midplane is also referred to as 'P0'.

Magnetic Probes in the ZaP Experiment

There are several different arrays of magnetic probes in the experiment. A surface mounted axial array measures the magnetic field in the theta direction. Two azimuthal arrays measure the magnetic field in the theta and z directions. Finally, an internal linear array, lying along a chord, measures the magnetic field in the theta, radial and z directions. The magnetic probes are B-dot probes that measure magnetic flux in the theta direction. The magnetic probes utilize the relation (defined by Faraday's Law) between electric fields and changing magnetic fields:

$$\oint_C \mathbf{E} \cdot d\mathbf{l} = - \int_S \dot{\mathbf{B}} \cdot d\mathbf{s} . \quad (6)$$

Specifically, the voltage induced in the coils is

$$V = -N A \dot{B} , \quad (7)$$

where N is the number of windings and A is the area. After integration (using an active integrator with an integration RC), we have

$$V_0 = \frac{N A B}{RC} . \quad (8)$$

The output voltage is then passed through a digitizer and multiplied by the RC value. If we know N and A , we can determine the magnetic field perpendicular to the flux surface⁷.

The surface magnetic probes in the ZaP experiment are contained within stainless steel cups. These cups are inserted into a relief within the surface of the outer electrode. The portion of the cup exposed to the plasma is protected with a tantalum covering. The tantalum covering itself is held in place with a copper ring. The copper wire windings (used to measure the magnetic field) are wrapped around a KEL-F block within the cup. A diagram of the probe and the probe layout follows.

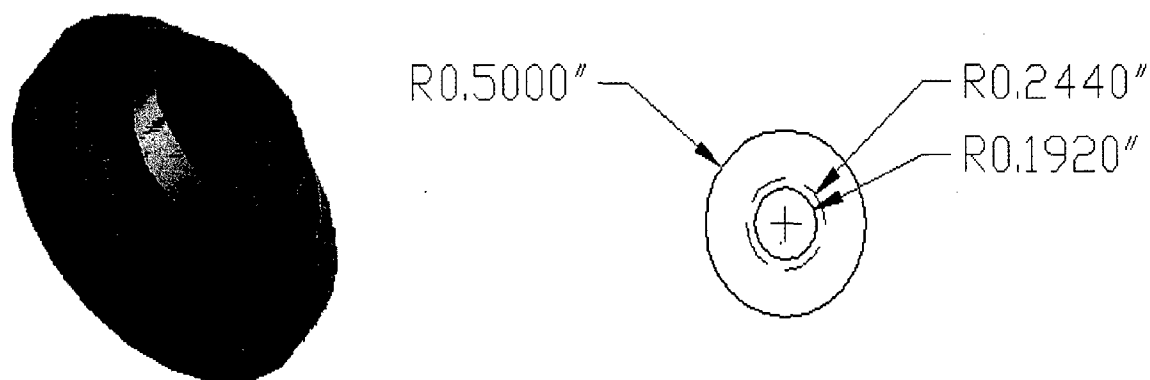


Figure 1.6: Design and dimensions of the magnetic probe cups. The bottom layer is the stainless steel. The second layer is the copper ring, which holds the tantalum covering (top layer). The magnetic probe is located beneath the tantalum covering, inside the cup.

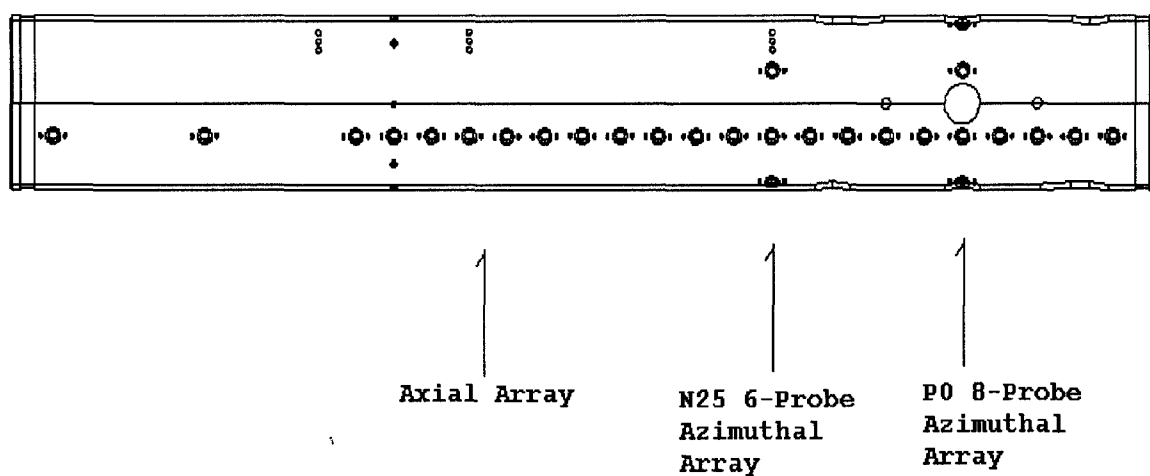


Figure 1.7: Location of the axial array and the two azimuthal arrays (at N25 and P0).

Chapter 2: Calibration of Magnetic Probes

Motivation

As outlined in the previous chapter, the magnetic field generated by the experiment induces a voltage in the magnetic probes. This voltage is integrated, digitized and then stored in the ZaP database. Theoretically, we could use the equations from Chapter 1 to calculate a conversion from the voltage recorded in the database to the actual magnetic field from the experiment. In reality, however, this proves impractical, as there are other factors that affect the signal.

A more accurate approach is to create a known magnetic field around the magnetic probes, measure the voltage from the probes and then calculate a conversion factor. By using the same data acquisition equipment for this calibration as will be used in actual experiment runs, the calibration will take into account any deviations due to the effects of the copper wall, digitizer or integrator. This calibration will also help compensate for any effects due to the wire leads, feedthrough connections, and so forth. Calibrating the probes in this manner is typically more accurate than straight calculations.

Internal Magnetic Chord Probes on the ZaP Experiment

An excellent example of the process required to calibrate magnetic probes comes from the calibration of the internal magnetic chord probes. These probes differ from the surface probes described in Chapter 1 in that they are inserted within a Boron Nitride tube into the chamber. They are inserted such that the tube describes a chord of the

circular cross section of the chamber; hence, the name of the probes. Although they are at a different location than the surface probes, they are still B-dot probes and so the physics behind their calibration offers a good example of the process of calibrating magnetic probes.

There are five chord probes, all located on the same KEL-F form, which is shown below in Figure 2.1. There are five slots on the KEL-F form, each of which is used for one 3-dimensional magnetic probe. The magnetic probes are simply copper wire wound around the slots in three orthogonal directions, with holes drilled through the KEL-F to allow the copper wire to pass through.

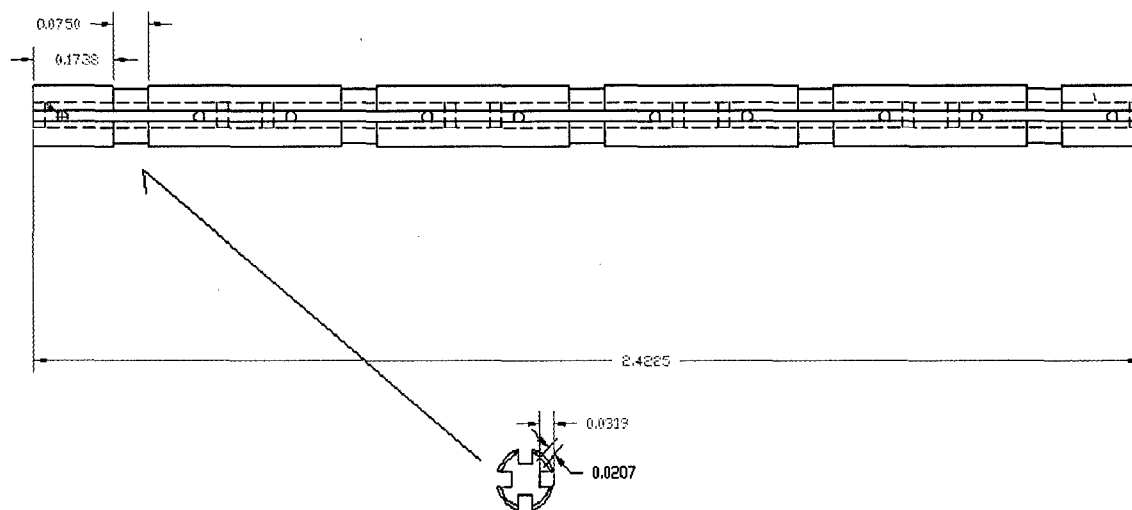


Figure 2.1: Top view of the internal chord probes. The copper coils are wrapped around each indentation in three orthogonal directions.

Calibration

Good calibration depends on a known magnetic field. To create this, we use a Helmholtz coil. The Helmholtz coil generates a known, uniform magnetic field over a wide space (relative to the magnetic probes). The uniformity of the magnetic field eliminates possible user error (through positioning of the probes) and is thus ideal for magnetic probe calibration. When calibrating the probes, we use the Helmholtz coil as part of an LRC circuit, so that a changing magnetic field is present (necessary for B-dot probes).

The entire circuit is diagramed in Figure 2.2 below. The Insulated Gate Bipolar Transistor (IGBT) serves as a gate that is triggered by a pulse generator (connected through a fiber optic cable). The pulse lasts roughly 50 ms. When the gate opens, it completes an LRC circuit, with the Helmholtz coil providing the induction. The chord probes are placed directly in the center of the Helmholtz coil (Figure 2.3). The LRC circuit creates a characteristic decaying sine wave current, which in turn stimulates a decaying sine wave magnetic field in the Helmholtz coil.

$$B = ae^{-\omega_1 t} \sin(\omega_2 t + \phi) . \quad (9)$$

The magnetic field in the coil is a function of current (measured with a Pearson probe) and the number of turns in the Helmholtz coil:

$$B = \frac{\mu_0 NI}{R} \left(\frac{4}{5} \right)^{\frac{3}{2}} . \quad (10)$$

In the above equation, N is the number of turns, R is the radius of the coil and I is the current. For our assembly, $R = 4.175$ inches (0.106045 m), and $N = 12$. Thus,

$$B = (1.017503 \times 10^{-4})I.$$

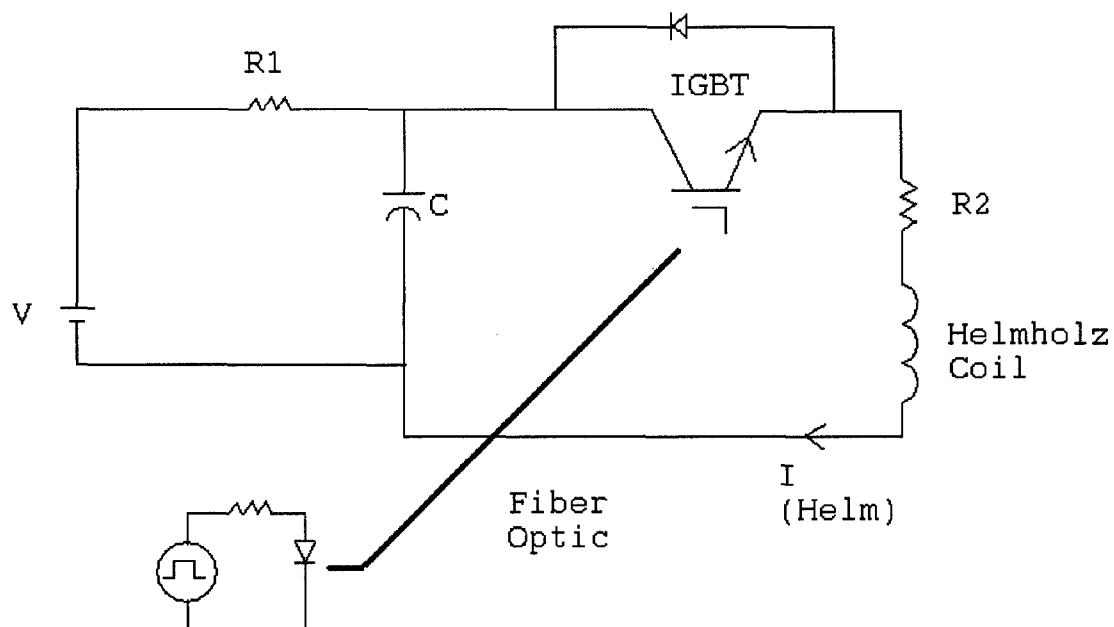


Figure 2.2: LRC circuit for calibration of the chord probes. The IGBT is connected to a pulse generator (controlled by a trigger).

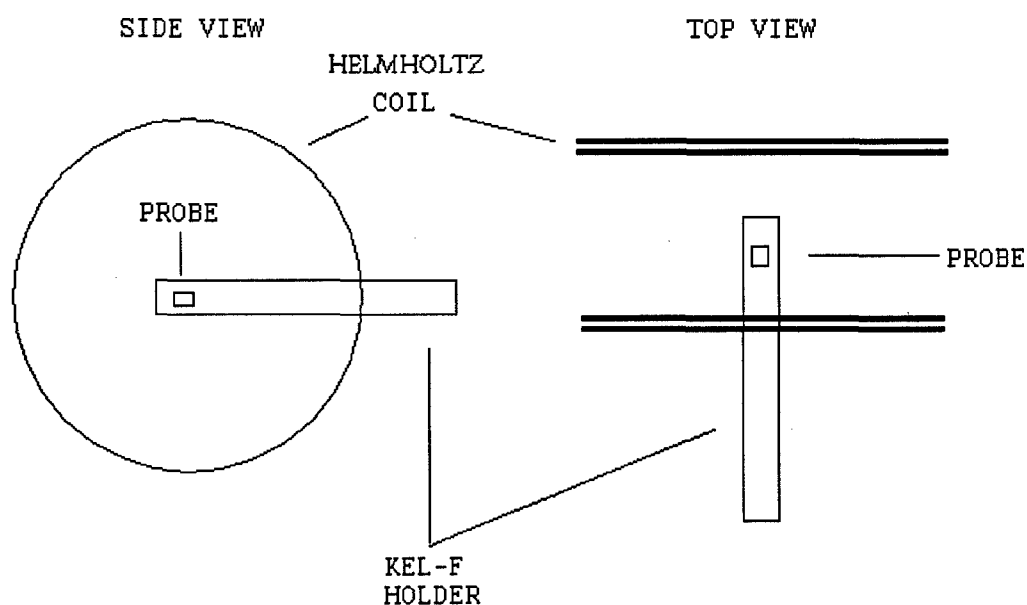


Figure 2.3: Top and side views of the calibration setup. The holder shown in the diagram is the same as in Figure 2.1.

Each of the five probes has three windings that measure the magnetic field in three perpendicular directions. To calibrate the probes, the holder is inserted such that one probe is dead center in the coil. Several shots are taken, and then the holder is moved forward such that the next probe is dead center. This process is repeated for all five probes. Next, the holder is rotated 90° (counter-clockwise, in our case) and the whole process starts once again.

Since the magnetic field of a Helmholtz coil, near the center, is purely in one direction, we would expect only one orientation of probe windings to register a signal. In practice, however, this is not the case. We must assume that the copper windings in each

probe are not perfectly aligned—the angle between windings is not exactly 90° . Thus, multiple windings will register a signal. In fact, the actual magnetic field is a combination of all three signals. For each three-axis probe, the measured signal is determined by:

$$\begin{aligned} a_{11}B_x + a_{12}B_y + a_{13}B_z &= B'_x \\ a_{21}B_x + a_{22}B_y + a_{23}B_z &= B'_y \\ a_{31}B_x + a_{32}B_y + a_{33}B_z &= B'_z \end{aligned} \quad (11)$$

which can be represented by

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} = \begin{bmatrix} B'_x \\ B'_y \\ B'_z \end{bmatrix}, \quad (12)$$

where B is the actual magnetic field and B' is the measured magnetic field. The a_{ij} constants must be determined through the calibration procedure.

When the probes are inserted in the coil, the magnetic field is only in one direction. For example, when the z-axis probes are oriented with a surface normal parallel to the coil's magnetic field, Equation 11 reduces to

$$\begin{aligned} a_{13}B_z &= B'_x \\ a_{23}B_z &= B'_y, \\ a_{33}B_z &= B'_z \end{aligned} \quad (13)$$

since B_x and B_y are 0. This allows us to solve for the constants in Equation 13. By rotating the holder 90° we solve for three more constants. One more rotation allows us to

solve for the last three constants and completely fill out the $\mathbf{a}$ matrix. Finally, we note that

$$\begin{bmatrix} & & \\ & \mathbf{a}^{-1} & \\ & & \end{bmatrix} \begin{bmatrix} B'_x \\ B'_y \\ B'_z \end{bmatrix} = \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix}. \quad (14)$$

Once we fill out the $\mathbf{a}$ matrix, we can take the inverse, $\mathbf{a}^{-1}$ and multiply by the measured signals to return the actual magnetic field, which is our desired goal.

Results of Calibration

To determine the a_{ij} constants, we must fit the actual magnetic field signal to the measured signal. In our calibration, we used the Interactive Data Language (IDL) CURVEFIT routine (a non-linear least-squares fit), which fits a single constant multiplier for a_{ij} by comparing the two signals. CURVEFIT is described in more detail in Chapter 5. Please see Appendix A for the IDL code used to calibrate the chord probes.

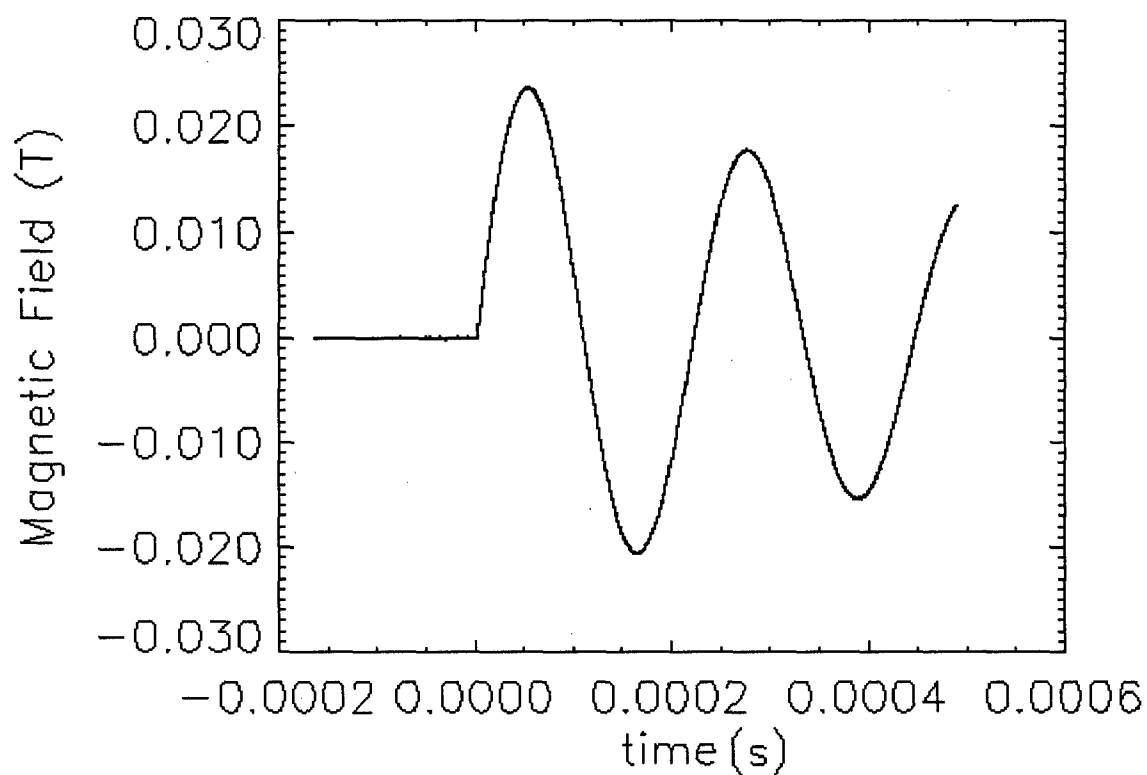


Figure 2.4: Magnetic field in the Helmholtz coil—calculated from Pearson probe readings, Pulse 11119024.

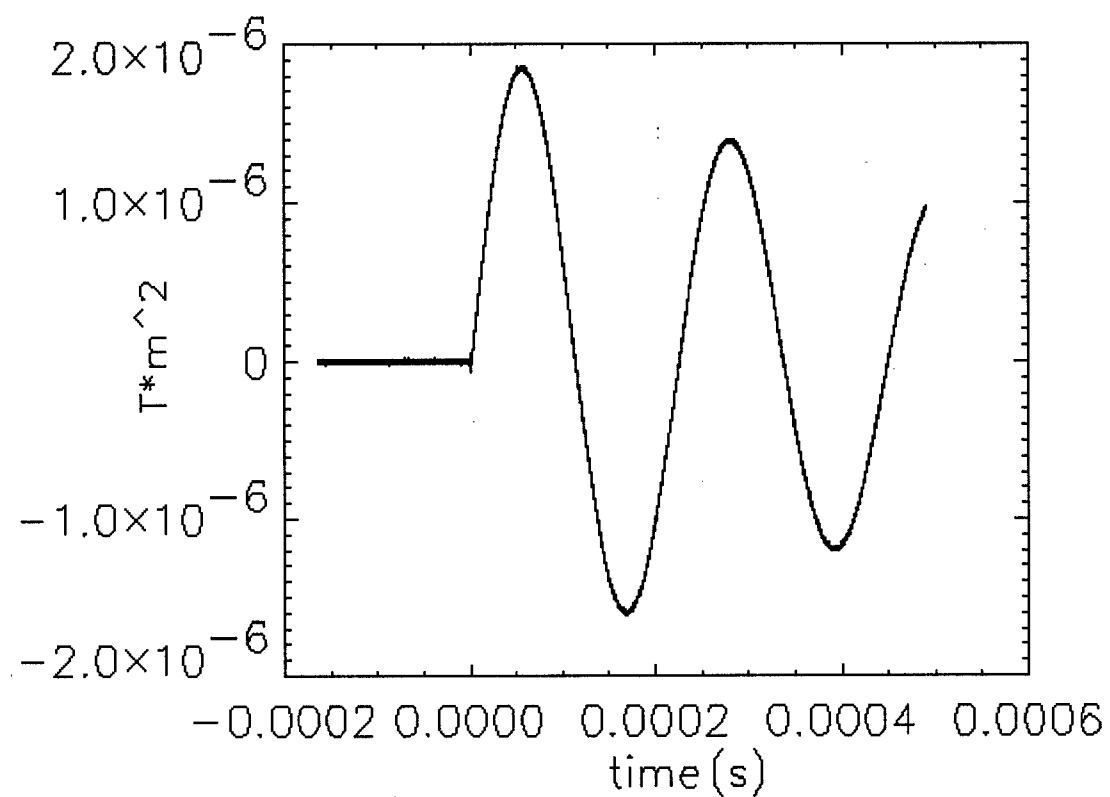


Figure 2.5: The measured signal from a magnetic probe, Pulse 11119024. The probe winding surface normal is parallel to the magnetic field.

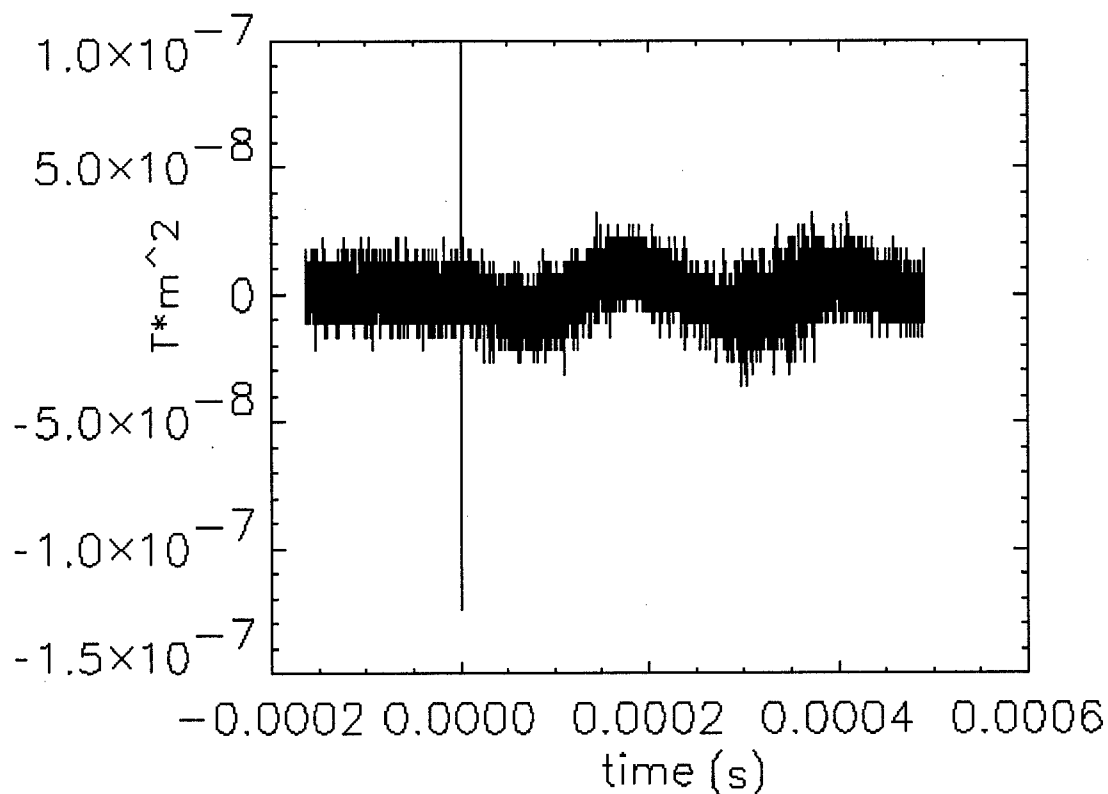


Figure 2.6: The measured signal from a magnetic probe, Pulse 11119024. The probe winding surface normal is perpendicular to the magnetic field.

The above figures show examples of the signals used in calibration. Figure 2.4 is the actual magnetic field, calculated by taking the current (measured with a Pearson probe) and using Equation 10. Figure 2.5 is the signal directly measured by the probe, after being integrated and digitized. Notice that the signals are in phase, so the CURVEFIT procedure is rather straightforward. Also, notice the scale difference between the two signals. Figure 2.6 shows a typical signal from a probe with a surface

normal that is perpendicular to the magnetic field. Finally, notice that the signal is not perfectly zero; this implies that the windings are not perfectly aligned.

The following figures show the results of the CURVEFIT routine for one probe. The black line represents the signal from the probe and the red line represents the actual magnetic field multiplied by the a_{ij} constant. Each set of three graphs represents a particular orientation. Each plot represents the signal from one winding. Notice that with nine graphs, all nine constants have been calculated, and the a matrix (as well as its inverse) can be calculated for this particular probe.

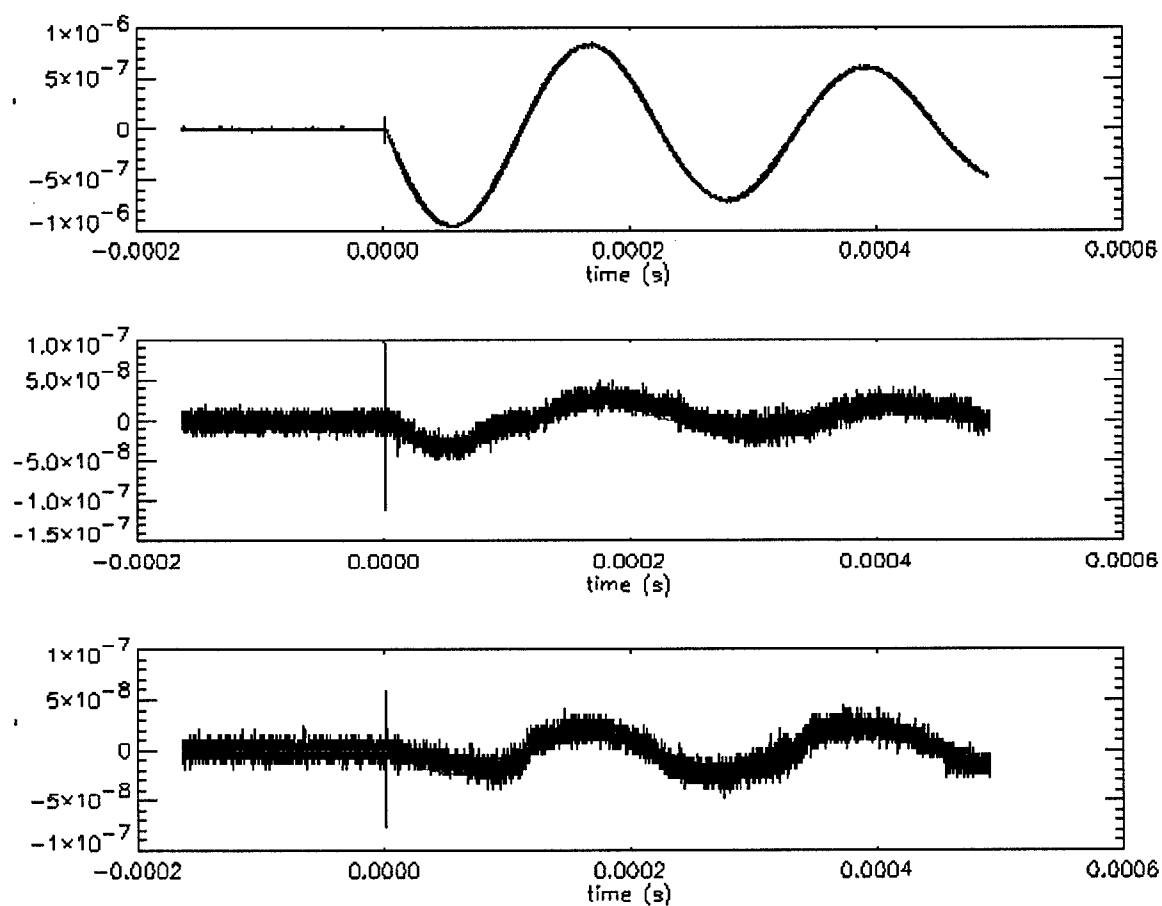


Figure 2.7: Calibration factor fitting for the first orientation. The top graph is winding 1, corresponding to the x direction (relative to the probe). The second graph corresponds to the y direction and the final graph corresponds to the z direction.

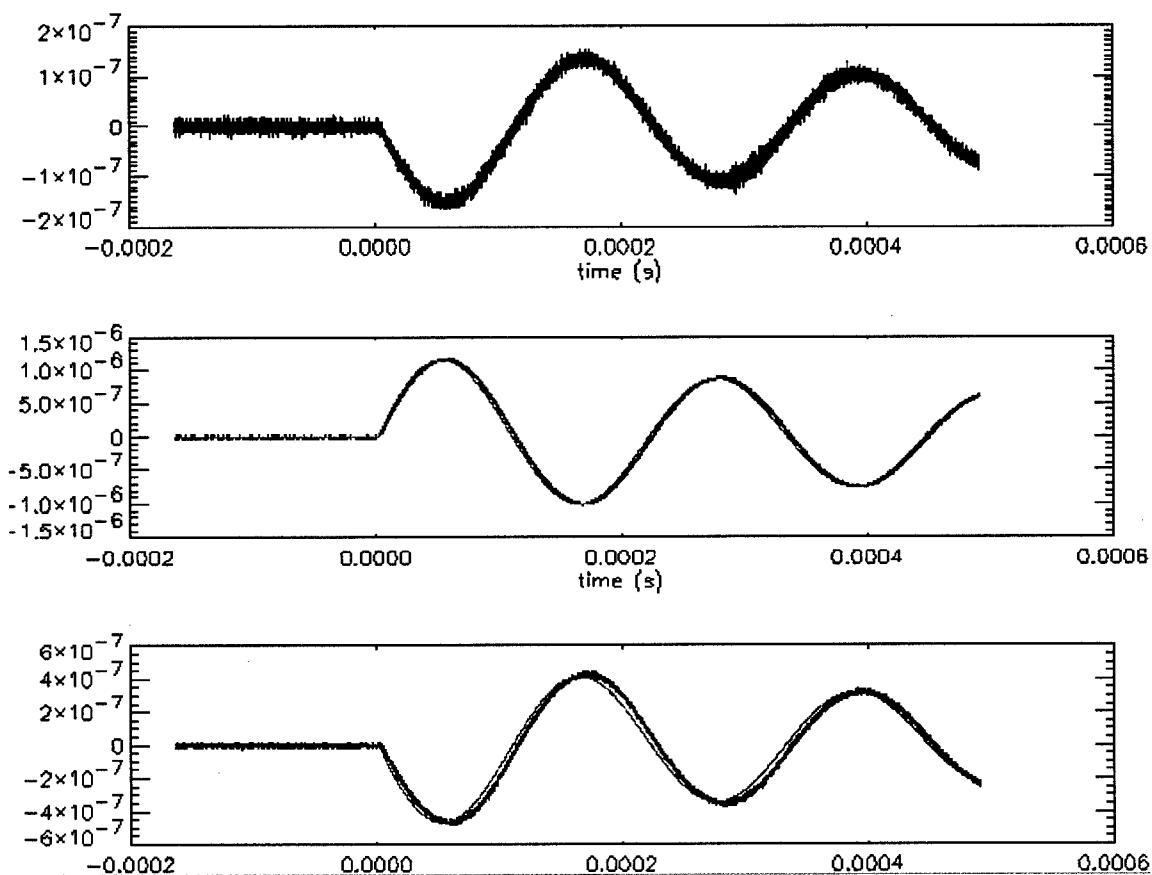


Figure 2.8: Calibration factor fitting for the second orientation. Notice that winding 2 (y direction) is showing the strongest signal.

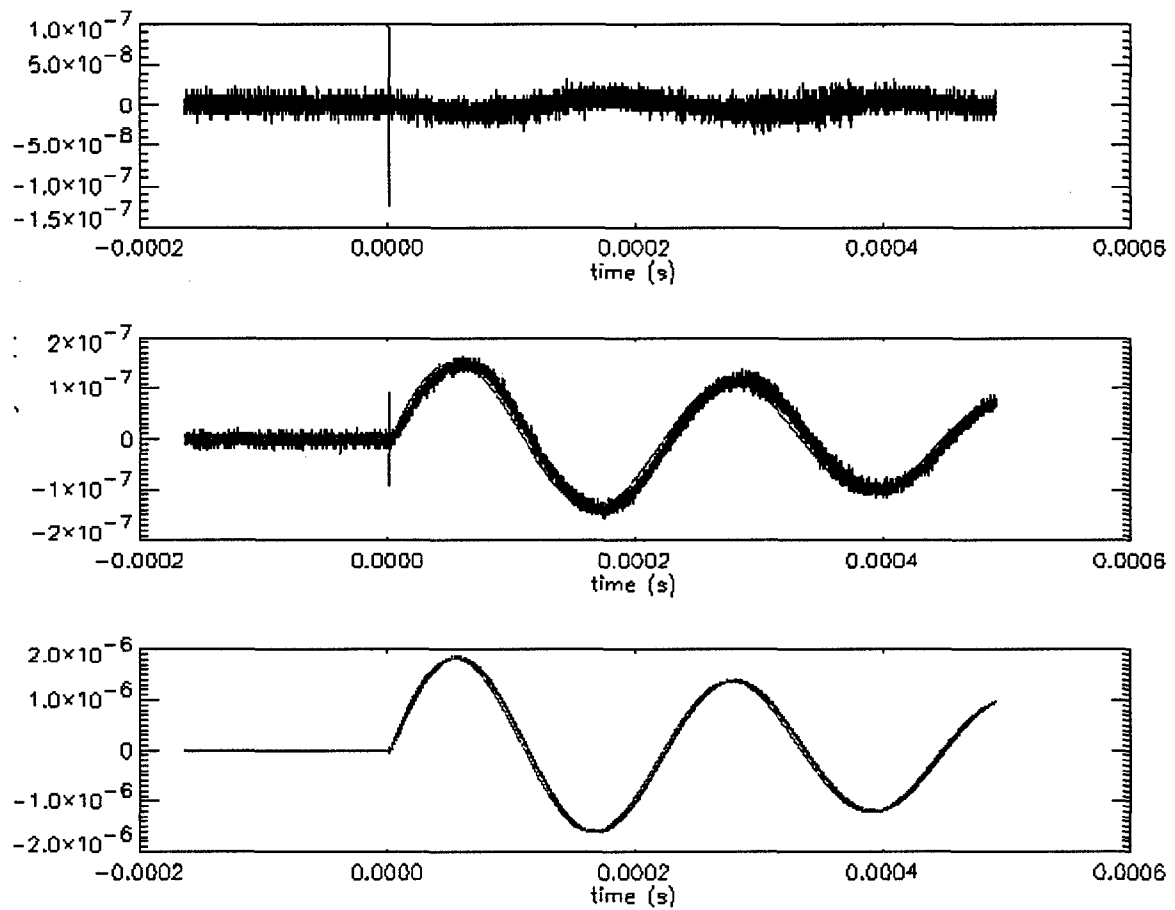


Figure 2.9: Calibration factor fitting for the third orientation. The z direction is showing the strongest signal.

We can now multiply the original measured signals by the inverse a matrix and compare the results to our desired results. We should see the magnetic field in one direction only, with the other two directions reading zero. Below is a plot that shows the results of multiplying the measured signals by the a^{-1} matrix. The black line is the actual magnetic field of the Helmholtz coil; the red, blue and green lines are the signals from the magnetic probe multiplied by the inverse matrix. The red line is from the winding with a

surface normal parallel to the magnetic field, the other two correspond to the two windings with a surface normal perpendicular to the magnetic field.

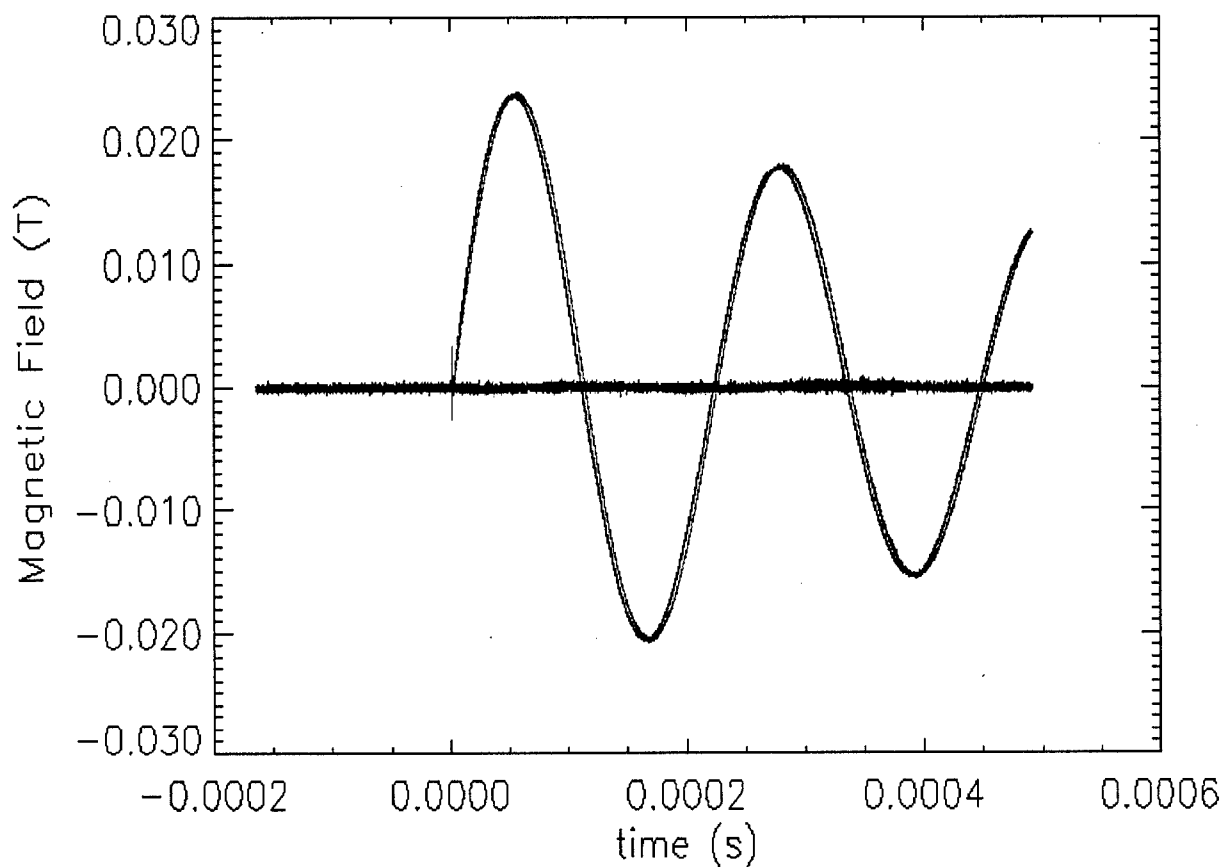


Figure 2.10: Final results from the calibration of a chord probe, with the green and blue signals representing the two windings with a surface normal perpendicular to the magnetic field.

Chapter 3: Fourier Representation of the Magnetic Field

Representation of Functions With Fourier Series

The magnetic field in the ZaP experiment is complicated and cannot be described by the simple theoretical equations such as those outlined in the first chapter. Although the plasma is a Z-pinch, the structure is changing with time. Furthermore, the structure is not a perfect Z-pinch at all times, which leads to a complex current profile. Due to these complications, it is generally not possible to represent the magnetic field with a simple closed form theoretical equation. Instead, we represent the magnetic field with a Fourier series function.

Not all mathematical functions can be represented by a Fourier series, but most can. Specifically, a Fourier series can represent most functions that describe physical problems (such as waves or electric fields). The Convergence Theorem for Fourier series states that:

“If $f(x)$ is piecewise smooth on the interval $-L \leq x \leq L$, then the Fourier series of $f(x)$ converges to the periodic extension of $f(x)$, where the periodic extension is continuous”⁸.

Representing the Magnetic Field

In the case of the magnetic field, the function is always continuous, even if the signal is zero. Thus, we are able to represent the magnetic field successfully with a Fourier series:

$$B(\theta) = \sum_{m=0}^{\infty} c_m \cos m\theta + \sum_{m=0}^{\infty} s_m \sin m\theta . \quad (15)$$

The constants, c_m and s_m , can be calculated by:

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} B(\theta) d\theta \quad (16)$$

$$c_m = \frac{1}{\pi} \int_{-\pi}^{\pi} B(\theta) \cos m\theta d\theta \quad (17)$$

$$s_m = \frac{1}{\pi} \int_{-\pi}^{\pi} B(\theta) \sin m\theta d\theta . \quad (18)$$

To create this function, we use measurements obtained from the azimuthal magnetic probe arrays. We do not, however, have an infinite number of measurements from $-\pi \leq \theta \leq \pi$. Instead, we have eight magnetic probes. Each probe is located 45° apart. Thus, we must approximate the integral used in calculating the constants in the Fourier series. We use the trapezoid method to calculate the integral, which offers a good approximation ($B_n(\theta)$ represents the measured magnetic field at probe n):

$$c_0 = \frac{1}{8} \sum_{n=1}^8 B_n(\theta) \quad (19)$$

$$c_m = \frac{1}{4} \sum_{n=1}^8 B_n(\theta) \cos m\theta \quad (20)$$

$$s_m = \frac{1}{4} \sum_{n=1}^8 B_n(\theta) \sin m\theta . \quad (21)$$

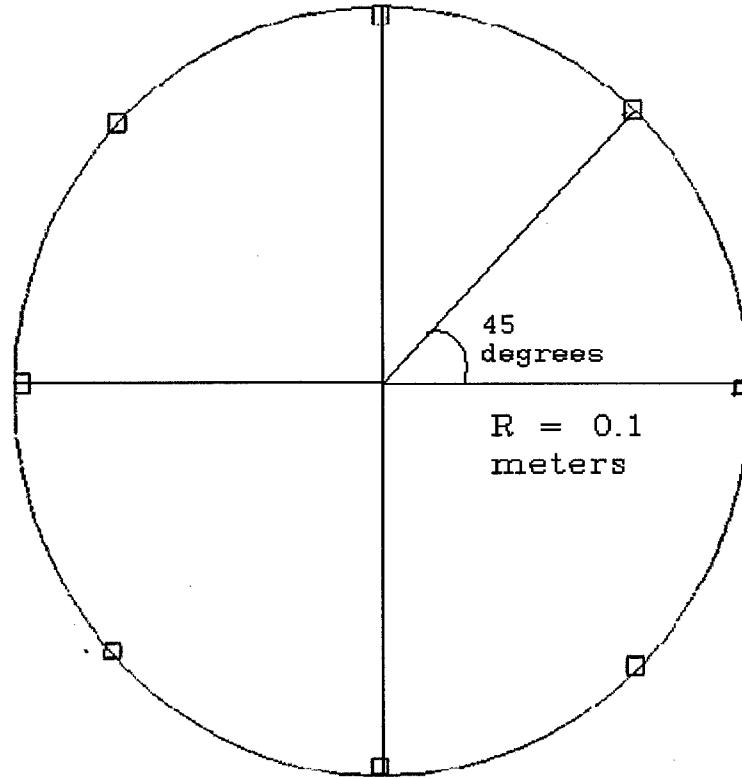


Figure 3.1: The azimuthal probe array layout, with each probe 45° apart.

Magnetic Field Modes

To represent the magnetic field perfectly, one would have to calculate an infinite number of constants. This is impossible, of course, so we must approximate the field by looking at a finite number of modes. Each mode represents a different structure of the magnetic field. By looking at these structures (individually and composite) we can learn a great deal about the plasma and the current that is creating the magnetic field. In order to analyze the modes, we look at the magnitude and phase.

$$\text{magnitude of mode } m = \sqrt{c_m^2 + s_m^2} \quad (22)$$

$$\text{phase of mode } m = \arctan\left(\frac{s_m}{c_m}\right). \quad (23)$$

Typically, the first several modes ($m=0, 1, 2$) dominate the physical structure. Higher order modes tend to add small detail to the overall structure. Thus, a good approximation of the structure can be determined from the initial modes.

The $m=0$ “mode” is actually the average magnetic field of the plasma at this plane. There is no phase for $m=0$, as there is no s_0 constant. In the ZaP experiment, this plane is either at P0 or at N25, the location of the two azimuthal probe arrays. The average magnetic field does not give us any information on the location or orientation of the plasma.

The $m=1$ mode represents the displacement of the plasma from the center axis. The higher the magnitude of the $m=1$ mode, the further the plasma is located from the center axis. The phase for this mode represents the orientation. Although this mode gives us location information, there is no information regarding the shape of the plasma.

The $m=2$ mode represents elongation of the plasma. The $m=1$ mode essentially assumes a line source for the current. The $m=2$ mode gives us information on the ellipticity of the actual plasma. The phase of this mode can give us the orientation of this elongation. Thus, the $m=2$ mode offers shape information, presenting more of a two dimensional image of the plasma.

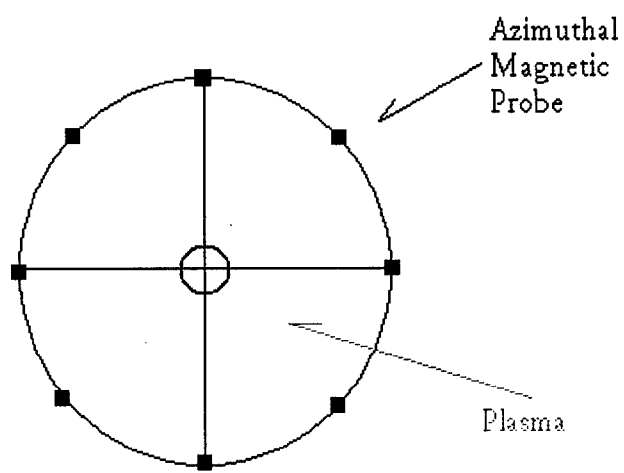


Figure 3.2: The $m=0$ representation.

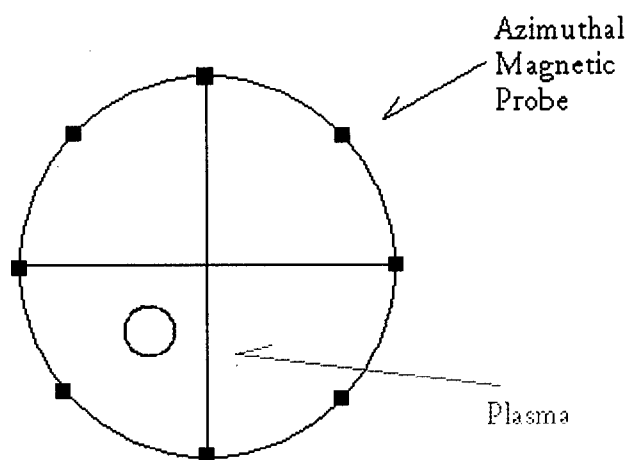


Figure 3.3: The $m=1$ mode represents the offset of the plasma current.

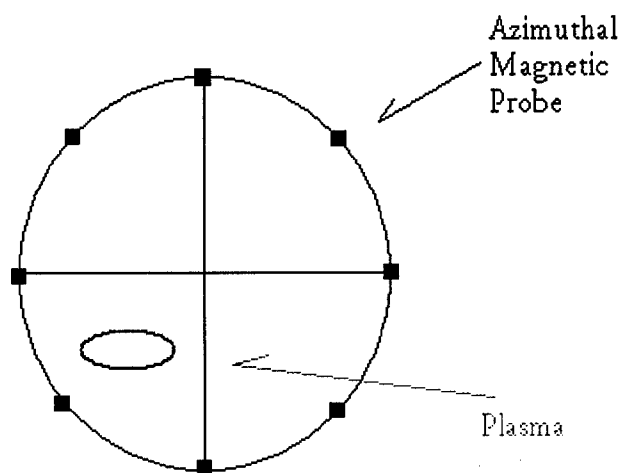


Figure 3.4: The $m=2$ mode represents elongation of the plasma current.

Normalization

We would like to normalize the mode calculations. Typically, we normalize the magnitudes by the average magnetic field at that plane. In practice, this translates to eliminating reliance on current, since the overall magnetic field is a direct function of current. By eliminating reliance on current, we can compare the displacement or elongation of different plasmas with different currents.

Below is a graph that depicts the magnetic mode values over time for a typical shot in the ZaP experiment. The current is included as a reference. The magnitude of the higher order modes is much lower than the average magnetic field, so they are multiplied by two in order to make them more visible.

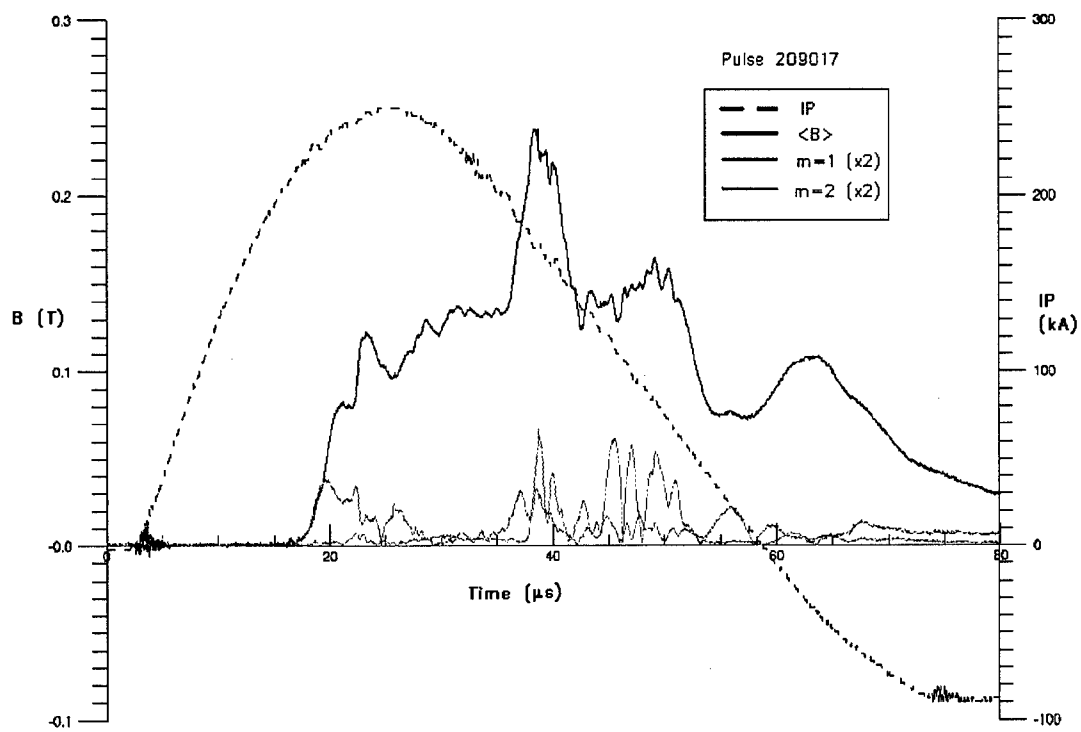


Figure 3.5: Typical magnetic mode data for a ZaP shot, Pulse 209017.

CHAPTER 4: Determining Plasma Properties from Magnetic Mode Data

Computer Simulation

The information obtained from the mode data returns information about the plasma that we are studying. Ideally, we would like to determine plasma properties from this information. As mentioned above, each mode equates to a distinct feature of the plasma. Is it possible, however, to determine physical measurements of plasma properties purely from magnetic mode data alone?

To test this, an IDL computer program was developed (see Appendix B). The purpose of this computer program is to simulate the ZaP experiment and predict magnetic probe measurements based on different positions of plasma current. After the probe measurements are predicted, the computer program calculates mode information (based on these measurements). Then, the mode data is compared to the physical location and shape of the simulated plasma. This gives us the relation between mode data and physical properties.

The program simulates the plasma as an infinite line of current. The magnetic field from this plasma is:

$$B_{\theta}(r) = \frac{\mu_0 I}{2\pi r}, \quad (24)$$

where I is the current and r is the radial distance from the current line. This equation is used to calculate the magnetic field at the eight distinct probes (see the “Flux Conservation” section later for a discussion of some assumptions made). Initially, the

program only modeled the plasma as one line of current. To model the $m=2$ effects, however, another line of current was added. Thus, the magnetic field at each probe was a sum of the contribution of each line of current. The program sets an arbitrary value for the overall current (100000 amps), but this value is not important as the results are eventually normalized to eliminate any reliance on current. Each line of current is considered to be carrying exactly 50% of the overall current.

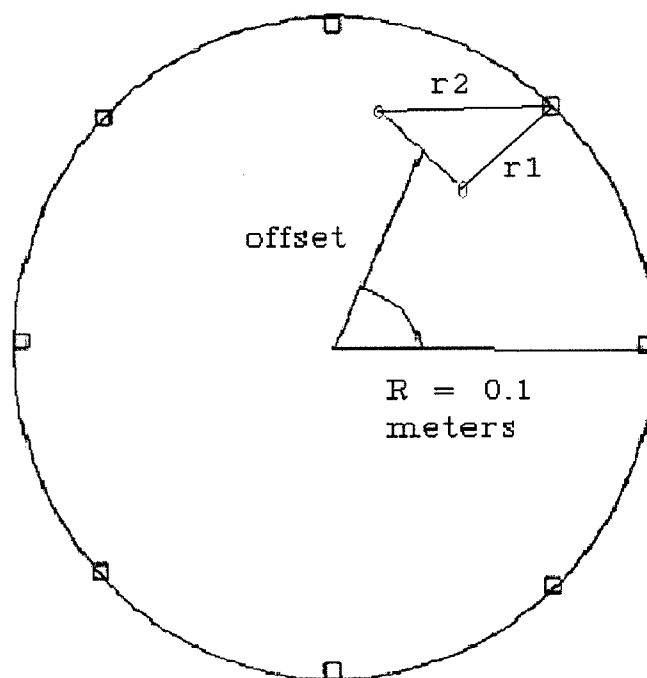


Figure 4.1: The computer model of the plasma represents the elongation of the plasma current with two distinct current filaments, shown in orange.

Four parameters are allowed to change in the computer program: the offset, the phase of the offset, the separation, and the phase of the separation. Notice that this corresponds to the magnitude and phase of the $m=1$ and $m=2$ modes. The program systematically increments the phase, then the offset, then separation of the filaments, and finally the phase of the filaments. The program then resets all the parameters and changes the offset phase again. In this manner, the program covers the entire input space. The input space is a 0.1 meter radius circle, which corresponds to the approximate size of the outer electrode in the ZaP experiment. The program is designed so that the user can change how many increments are used when changing different parameters.

When the program moves the plasma filaments about, it calculates the radius from each plasma filament to each magnetic probe. It then implements the following:

$$B_{\theta}(r) = \frac{\mu_0 I}{2\pi r_1} + \frac{\mu_0 I}{2\pi r_2}. \quad (25)$$

In the above equation, I is set to 50000 amps so that the total current is 100000 amps.

There are two approaches to calculating the values of r_1 and r_2 . The original method used in the computer program relied heavily on the law of cosines and the law of sines, drawing triangles and calculating angles until the length of r_1 and r_2 could be calculated as a function of the four independent parameters.

Eventually, however, a more straightforward method was developed. This method again uses trigonometry, but relies on calculations of coordinates in the X-Y plane and utilizes the classic distance formula. Thus, r_1 and r_2 are determined as follows:

$$\begin{aligned}
x_1 &= \cos(\alpha) * \zeta + \cos(\gamma) * \Delta r \\
y_1 &= \sin(\alpha) * \zeta + \sin(\gamma) * \Delta r \\
x_2 &= \cos(\alpha) * \zeta - \cos(\gamma) * \Delta r \\
y_2 &= \sin(\alpha) * \zeta - \sin(\gamma) * \Delta r \\
r_1 &= \sqrt{x_1^2 + y_1^2} \\
r_2 &= \sqrt{x_2^2 + y_2^2}
\end{aligned} \tag{26}$$

In the above equations, α is the phase of the $m=1$ offset, ζ represents the offset of the plasma filaments (as detailed in Figure 4.1), γ is the phase of the $m=2$ mode, and Δr is half the separation of the two plasma filaments. Note that the $m=2$ phase is only rotated 180° . Since both filaments carry the same current, rotating them more than 180° would repeat previous calculations.

Flux Conservation

One important issue that has to be considered is flux conservation. The question is: if the plasma is moved toward the wall, will the magnetic field at the wall be purely in the B_θ direction? This would be true if the magnetic flux from the plasma current was contained entirely within the 0.1 meter radius circle. However, if it is assumed that the magnetic fields can penetrate the copper lining of the outer electrode, then the field lines will not necessarily be perpendicular to the wall at the edge of the outer electrode. We decided to assume that flux was conserved; in other words, the field lines would be contained within the chamber, and would be parallel to the wall at the edge. This made calculations easier, and turned out to be a reasonably accurate assumption, especially when dealing with short time periods.

The $m=1$ Mode

The computer program reveals several important results. First, the program shows the exact correlation between the $m=1$ mode and the actual position of the plasma within the chamber. If the $m=1$ mode is normalized by the average magnetic field, and then multiplied by the radius of the outer electrode (0.1 meters), it is equal to the radial position of the plasma filament (or the centroid, if we are dealing with two plasma filaments). This statement is mostly true, although after about 0.08 meters the linear relationship no longer holds.

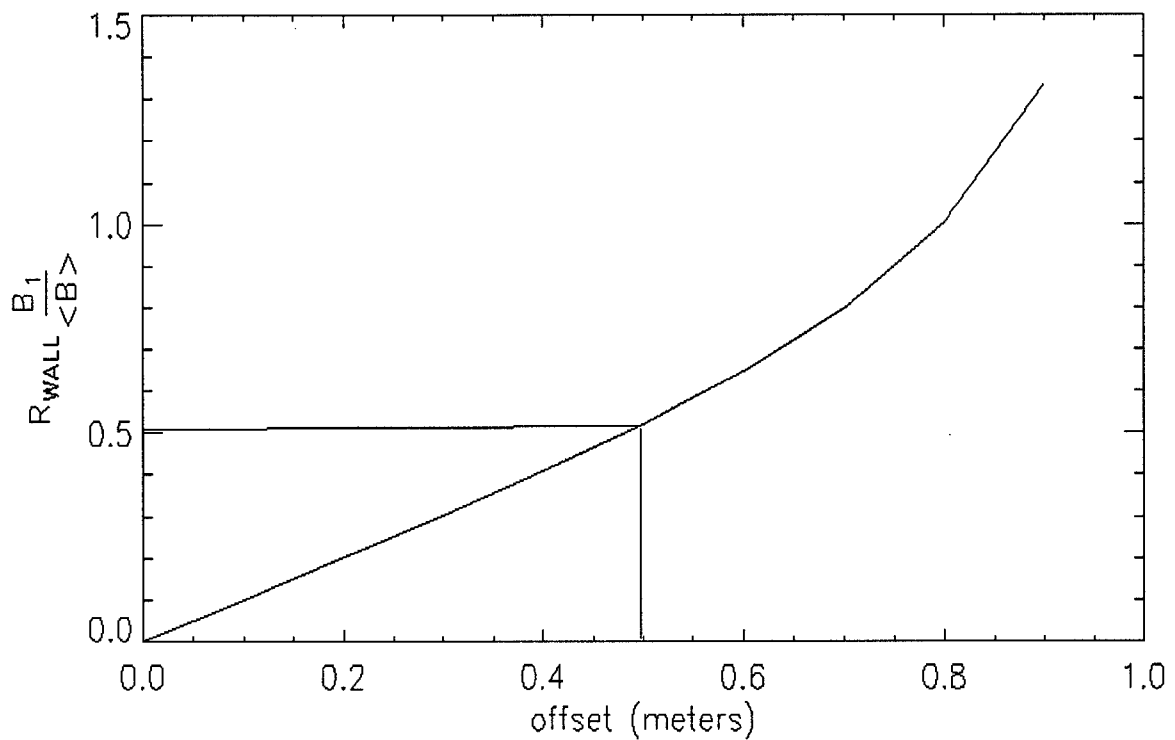


Figure 4.2: Normalized $m=1$ mode versus radial offset shows a mostly linear relationship.

This relationship between the radial position and the $m=1$ mode allows us to determine the plasma position within the chamber from magnetic probe data. Although the linear relationship breaks down slightly after 0.08 meters, this is not an enormous problem, as we rarely see normalized $m=1$ values above 0.08 (after they are multiplied by the radius of the outer electrode, 0.1 meters).

The $m=2$ Mode

Unfortunately, the $m=2$ mode does not supply complete information about the separation of the plasma current (or elongation). The computer program shows that the normalized $m=2$ mode is coupled to the normalized $m=1$ mode. In other words, if we increase the offset, without changing the elongation, we will still see a change in the $m=2$ normalized value. This makes it difficult to use only $m=2$ information when attempting to determine the elongation of the plasma.

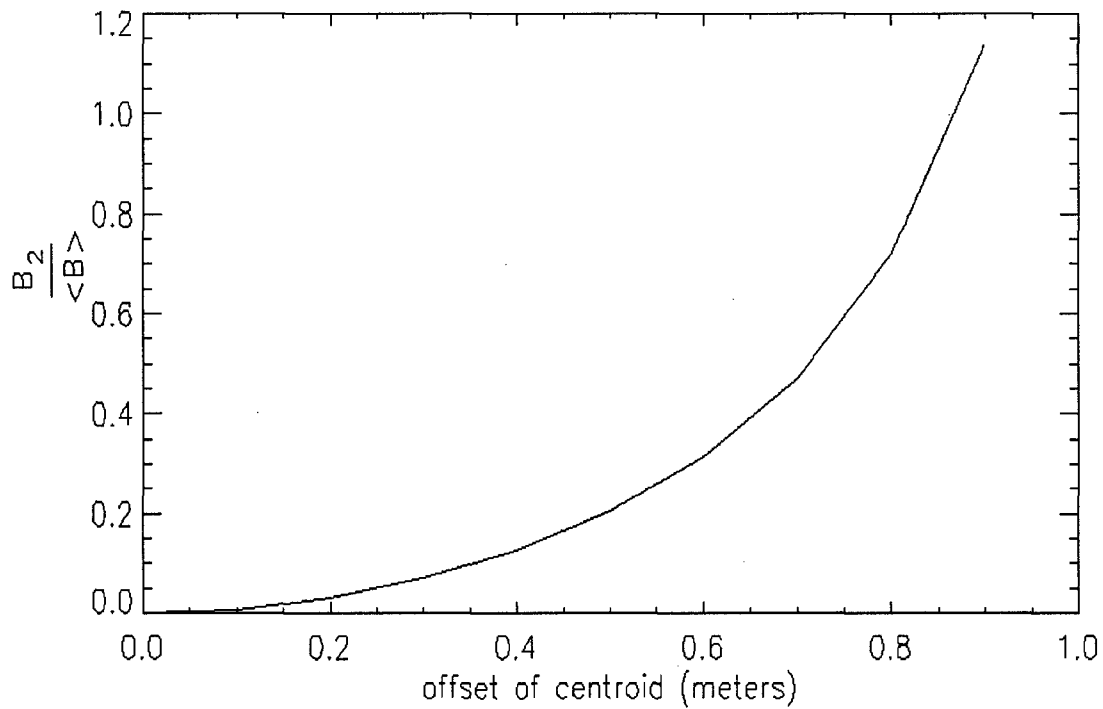


Figure 4.3: The $m=2$ mode versus offset of centroid shows a coupling between the $m=1$ mode and the $m=2$ mode.

Although the mode information provides us with some information regarding the plasma, obviously it alone cannot provide everything we would like to know. The $m=1$ mode is not entirely linear and the $m=2$ mode is coupled to the $m=1$ mode. This prevents us from using only mode information to determine the physical size and location of the plasma in our experiment. To obtain time dependent position and size information, we decided to develop a computer program to determine these properties, without using any mode analysis at all. This would allow us to not only create an independent method of calculating position and size, but it would also serve to validate some of the magnetic mode analysis that has been done at a theoretical level so far.

Chapter 5: The Non-Linear Fitting Routine

Layout of the Non-Linear Fitting Computer Program

The goal of the non-linear computer fitting routine is to create a program that calculates the position of the plasma current, based on the magnetic probe readings. This calculation is independent of any mode computations, so the mode data can be used to check and verify the results of the non-linear fit. The program is designed to be used while the experiment is running, returning a time-based position calculation for the plasma after every pulse.

The non-linear fitting routine is designed to fit four parameters. These parameters completely describe the position of two plasma currents within the 0.1 meter radius outer electrode. The plasma is considered to be two currents (like before) to represent the $m=2$ mode. Both plasma filaments carry an equal amount of current in the calculations, so that each filament is given equal weight when calculating the magnetic field. (This can be changed at a later time, if any evidence is found which suggests that the current distribution in the plasma is not symmetrical). The program was created to calculate the plasma position at 'P0', which is the pinch midplane (see Figure 1.5). The program can easily be adapted, however, to calculate the position at 'N25', which is located at the end of the acceleration region.

The fitting routine attempts to fit values to four geometric parameters (describing the position of two current filaments) so that the magnetic field generated by these currents matches—as closely as possible—the magnetic probe readings at each probe

location. Essentially, the two current filaments are allowed to move throughout the 0.1 meter radius circle until the magnetic field generated by these currents is as close to the measured value as possible. As before, the magnetic field is calculated using the assumption that the current filaments are infinite lines of current, and the magnetic fields are then calculated using Equation 25.

The program was written so that it accepts as inputs the time dependent values of the eight magnetic probes (at P0) for the entire experimental run. It then returns the position of the two current filaments for that same time period. Typically, a shot consists of 16,383 data points (depending on the digitizer), over a time period of roughly 650 μ s. This means that the program accepts 16,383 eight-element vectors as input.

Geometries

Any four parameters that describe the position of the two current filaments within the 0.1 m circle can theoretically be used with the fitting routine to determine the position. In practice, however, different geometries work with varying degrees of success. Two different geometries were designed to calculate the position of the plasma filaments. The first geometry is based on the mode structure, as described in Chapter 4. Thus, the four parameters are:

- 1) *R_offset, the offset of the centroid of the plasma currents.*
- 2) *α , the angle between R-offset and the 0° plane.*
- 3) *Delta_r, the separation between plasma currents.*
- 4) *γ , the angle between Delta-r and the 0° plane.*

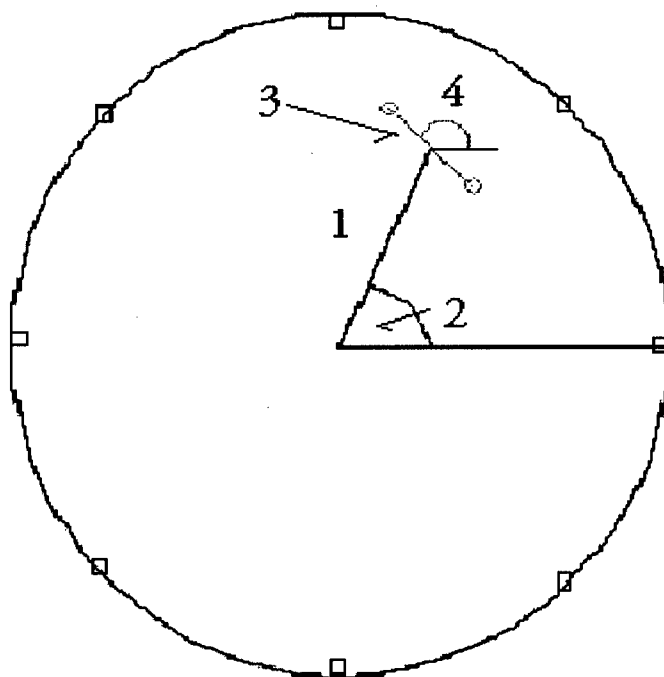


Figure 5.1: Geometry 1 is based on the mode structure.

The second geometry is based on a more traditional radius and angle combination.

Once again, there are four parameters. In the second geometry, however, there is no connection between the first current filament and the second. The four parameters are:

- 1) *Radius1, the radius to the first filament.*
- 2) *Angle1, the angle between Radius1 and the 0° plane.*
- 3) *Radius2, the radius to the second filament.*
- 4) *Angle2, the angle between Radius1 and the 0° plane.*

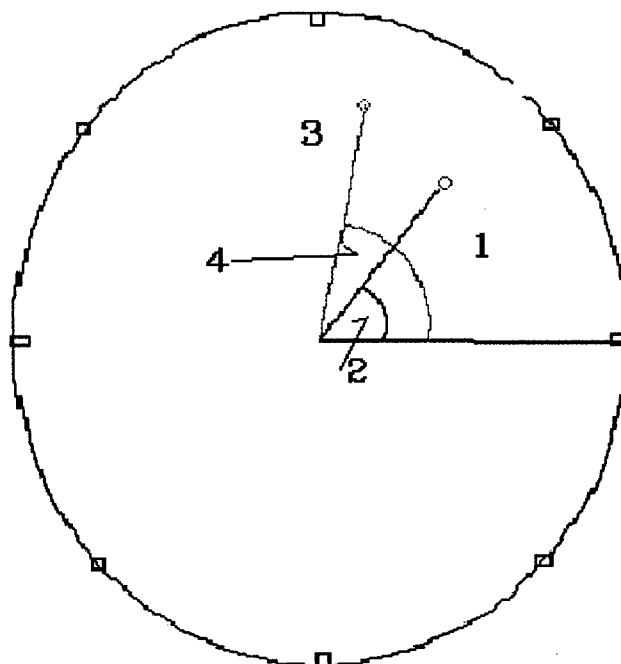


Figure 5.2: Geometry 2 is based on polar coordinates.

The computer program runs a non-linear routine to fit the parameters described above. Like most non-linear routines, this routine is heavily dependent on an initial guess. Without an accurate initial guess, the non-linear fit will often fail to converge, or return an incorrect (non-physical) value for the parameters. To supply an initial guess, the computer program is set to run a non-linear fit on a simpler geometry, with only two parameters.

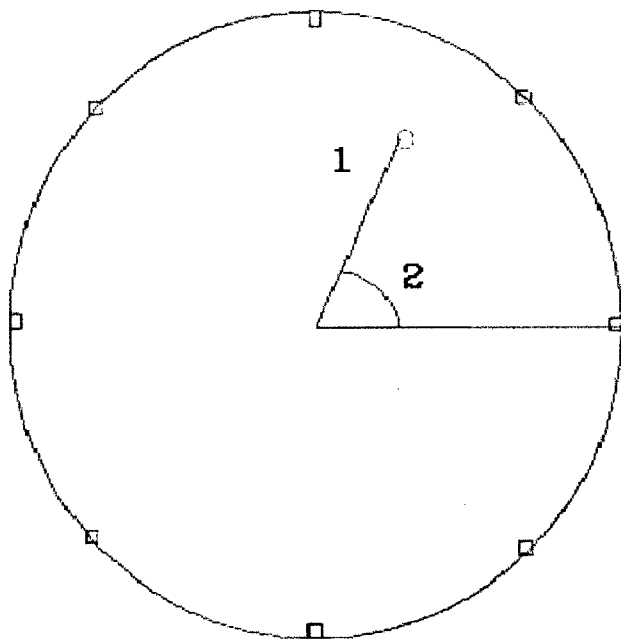


Figure 5.3: The Predictor represents the plasma as only one current filament.

The initial guess for these parameters is not as important, since the results do not have to be extremely accurate. The computer program sets an initial guess for parameter 1 to be 0.001 meters, and parameter 2 is set to 0.05 radians. The program then runs the non-linear fit on the first data point from the magnetic field data, using the predictor geometry. After the two parameters are determined, it passes these values as initial guesses for some of the parameters for the more complicated fitting routine. After the first data point is complete, the program uses the final results from the four variable non-linear fit as initial guesses for the next data point. At every data point, the predictor routine is used to supply initial guesses for the more complicated routine. This method of

using a predictor is helpful in ensuring convergence for the more complex fitting routine (using four parameters).

Testing Different Geometries

Although both geometries should theoretically be usable, several tests were created to test the performance of both methods. Ideally, we would like a stable routine that converges consistently and accurately. The first test utilizes synthetic data points. Using the computer program described in Chapter 4, the magnetic field is calculated for different plasma orientations. Then, both geometries are used to attempt to fit the parameters and return the correct magnetic field. Both geometries use the predictor to supply initial guesses.

The first test shows that geometry 2 performs better than geometry 1. In general, geometry 1 is inconsistent, failing to converge on many data points. Convergence failures are dangerous, since the non-linear fit relies on previous time values to supply an accurate initial guess for the current fit. Furthermore, geometry 2 appears to be more accurate than geometry 1. It is not immediately obvious why this should be. One possibility is that the value of the $m=2$ phase (parameter 4) can become meaningless as the magnitude of the $m=2$ mode approaches zero. The standard deviation for parameter 4 is often very high, meaning the computer program is having difficulty fitting this number. At times this would result in a failure to converge.

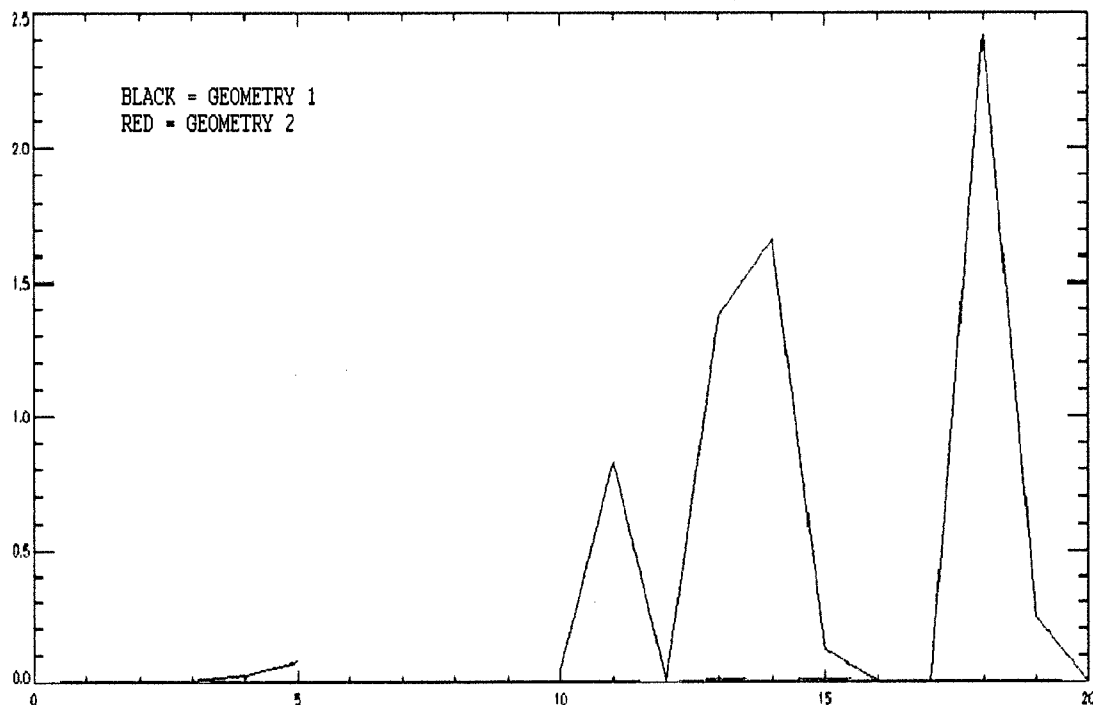


Figure 5.4: Error comparison for geometry 1 and geometry 2, synthetic data.

Figure 5.4 shows an error comparison for the two geometries, using synthetic data. Notice that the error for geometry 2 is at all times far below 0.1 (and is thus barely visible), while the error for geometry 1 grows. Also, notice that geometry 1 fails to converge for several data points, where the error line is non-continuous.

The second test used to compare the two methods utilizes data from actual experiment shots. The test runs a non-linear fit on the real data, using both geometries, and then compares the results. As before, both methods use the predictor to establish an initial guess for the more complicated four-parameter geometry. To measure accuracy, χ^2 is plotted for both methods. χ^2 is calculated as the difference between the actual magnetic field, and the magnetic field that would be generated given the position of the

plasma currents after the fit is completed. This difference is squared in order to eliminate negative values.

The second test again reveals that geometry 2 works better than geometry 1. We see that geometry 1 will at times fail to converge. Also, the error for geometry 2 appears to be lower than that of geometry 1. The results for one such test is displayed below. Note that the significant results are in the range of 20-50 μs , when the plasma is more stable. It is in this time period that the assumption of an infinite line of current is most accurate, and the fitting routine works best.

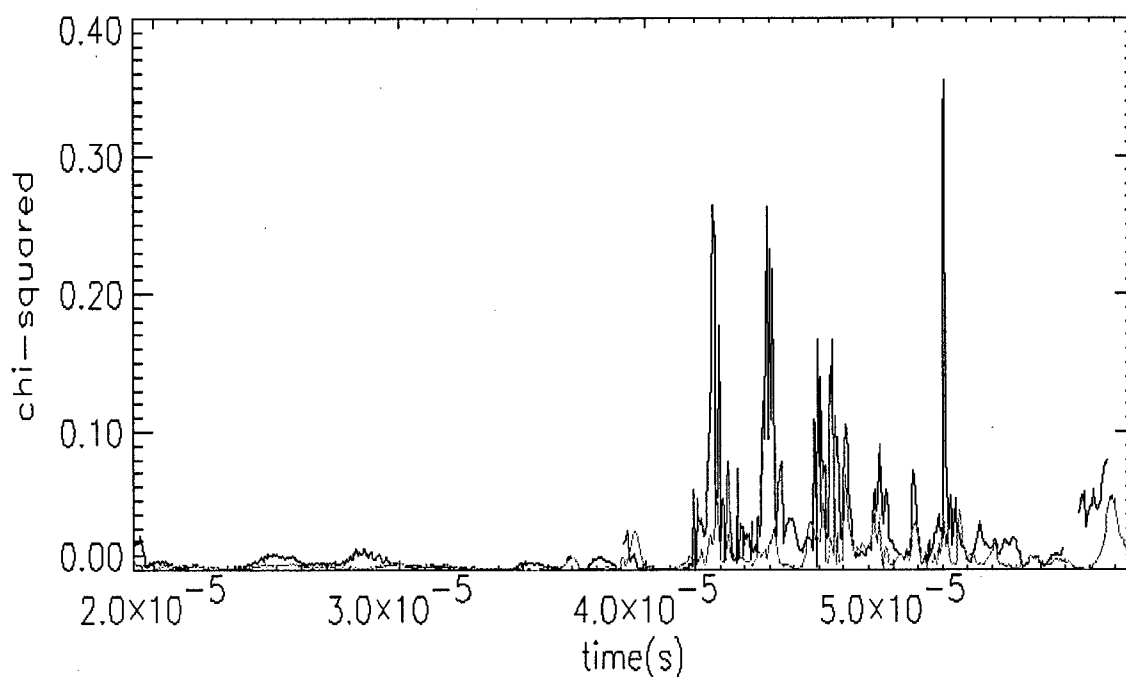


Figure 5.5: Error plot for geometry 1 (black) and geometry 2 (red), Pulse 726025.

Overall, geometry 2 works more accurately and with better consistency. This geometry was selected for use with the non-linear fitting routine.

The CURVEFIT Routine

The non-linear fit program uses Interactive Data Language (IDL) for the base computer language. The program calls the IDL routine "CURVEFIT" to run through the actual regression. CURVEFIT is a non-linear, least-squares gradient-expansion algorithm.

Using geometry 2, CURVEFIT attempts to determine the four parameters so that the difference between the actual probe readings and the calculated probe readings is minimized. CURVEFIT accepts the four parameters within the context of a non-linear function. The four parameters are all contained within one vector, **A**.

$$A_0 = \text{radius1}$$

$$A_1 = \text{radius2}$$

$$A_2 = \text{angle1}$$

$$A_3 = \text{angle2}$$

The angular positions of the magnetic probes are stored in an eight element vector, **X**.

Thus, $X_1 = 0^\circ$, $X_2 = 45^\circ$, etc. The fitting function is derived using the law of cosines to determine a length from each plasma filament to the actual probe (δ_1 and δ_2). For each probe, we define the following:

$$\beta_i = X_i - A_2, \quad (27)$$

$$\gamma_i = A_3 - X_i, \quad (28)$$

$$\delta_{1,i} = \sqrt{R^2 + A_0^2 - 2 \cdot R \cdot A_0 \cdot \cos(\beta_i)} , \quad (29)$$

$$\delta_{2,i} = \sqrt{R^2 + A_1^2 - 2 \cdot R \cdot A_1 \cdot \cos(\gamma_i)} , \quad (30)$$

where R is the radius of the outer electrode, 0.1 m, and the subscript i refers to each of the eight magnetic probes. The $\delta_{x,i}$ terms are the distance from plasma filament x to magnetic probe i . We could now use Equation 25 to determine the magnetic field at each probe, with r_1 and r_2 being equal to δ_1 and δ_2 , respectively. The current, however, is not constant from shot to shot. To eliminate the reliance on current, we normalize the magnetic field values:

$$\overline{B}_i = \frac{B_i}{B_{AVG}} , \quad (31)$$

where B_i is the measured magnetic field at each probe and B_{AVG} is the average field of all eight probes. To remain consistent, we must divide the right side of Equation 25 by the average magnetic field. Note that the average magnetic field is:

$$B_{AVG} = \frac{\sum_{i=1}^N B_{\alpha}}{N} = \frac{\mu_0 I}{2\pi N} \sum_{i=1}^N r_i^{-1} , \quad (32)$$

where N is the number of magnetic probes (eight). Since there are two plasma filaments, we have

$$B_{AVG} = \frac{\sum_{i=1}^N B_{\alpha}}{N} = \frac{\mu_0 I}{2\pi N} \sum_{i=1}^N (r_{1i}^{-1} + r_{2i}^{-1}) . \quad (33)$$

With two plasma filaments, we will assume that the current through each is equal:

$$I_1 = I_2 = \frac{1}{2} I. \quad (34)$$

Thus we divide Equation 25 by the average magnetic field, and we have:

$$\bar{B}_i = \frac{\frac{\mu_0 \frac{1}{2} I}{2\pi\delta_{1i}}}{\frac{\mu_0 \frac{1}{2} I}{2\pi N} \sum_{i=1}^N (\delta_{1i}^{-1} + \delta_{2i}^{-1})} + \frac{\frac{\mu_0 \frac{1}{2} I}{2\pi\delta_{2i}}}{\frac{\mu_0 \frac{1}{2} I}{2\pi N} \sum_{i=1}^N (\delta_{1i}^{-1} + \delta_{2i}^{-1})}, \quad (35)$$

where r_i has been replaced by δ_i , as defined in Equation 29 and Equation 30. After canceling like terms, we have the final function that the program fits to:

$$\bar{B}_i = \frac{\delta_{1i}^{-1}}{\frac{1}{N} \sum_{i=1}^N (\delta_{1i}^{-1} + \delta_{2i}^{-1})} + \frac{\delta_{2i}^{-1}}{\frac{1}{N} \sum_{i=1}^N (\delta_{1i}^{-1} + \delta_{2i}^{-1})}. \quad (36)$$

CURVEFIT requires partial derivatives of the fitting function in order to proceed.

The routine can also calculate partial derivatives on its own using a finite difference approximation. The partial derivatives of Equation 36, with respect to the parameters A_0 , A_1 , A_2 and A_3 would be cumbersome and difficult to program accurately, so the fitting routine uses the finite difference approximation instead. This is calculated automatically within the CURVEFIT routine.

Initial Results

The final test for the non-linear fit routine involved checking the performance on a variety of actual shots. Although error analysis allows us to compare different methods for the fit, it does not give us a good idea of the overall performance of either method. To check the overall performance of the non-linear method, we need to use mode analysis.

As described in Chapter 4, there is a linear relationship between the offset of the center of the plasma and the $m=1$ mode. The $m=1$ mode is calculated for all time points for each shot, so we have some base information to work with. By taking the average position (center) of the plasma filaments calculated by the fitting routine, we have useful information for comparison. We can compare the computed center of the plasma with the $m=1$ mode to see if we are getting an accurate fit.

The results of this comparison are shown below. As can be seen, the fit is very accurate during the initial stable period. The fit becomes less accurate as the rate of change of the position increases. This corresponds to the unstable period. Even during this period, however, the fit remains roughly accurate over time. Results from two different shots are shown (Figures 5.6 and 5.7) as an example of performance. The $m=1$ mode was normalized by the average magnetic field and then multiplied by 0.1 as described in Chapter 4.

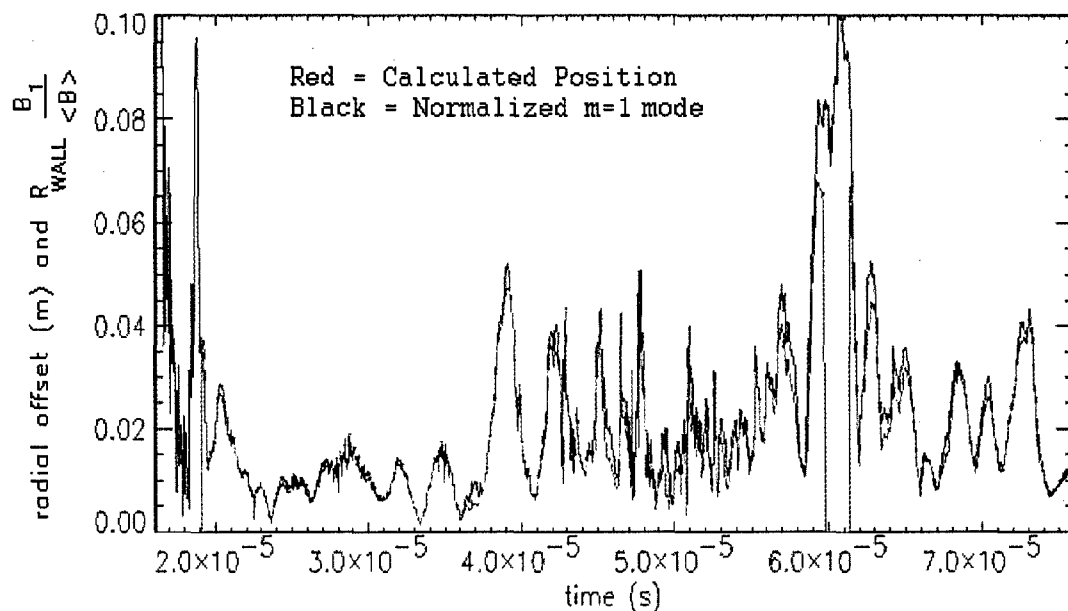


Figure 5.6: Comparison of fitting routine results to $m=1$ mode, Pulse 726025.

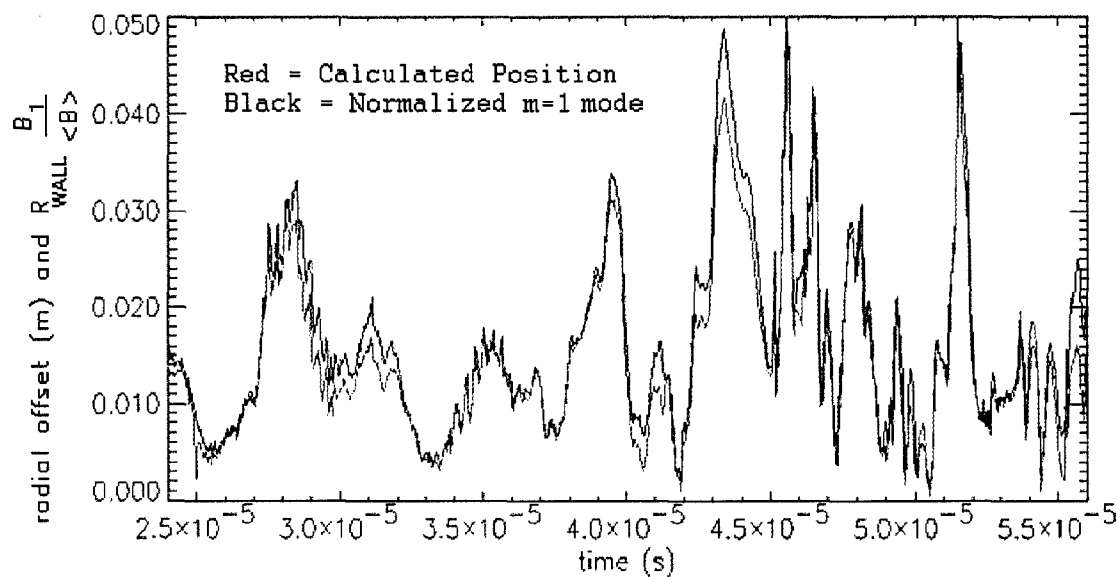


Figure 5.7: Comparison of fitting routine results to $m=1$ mode, Pulse 726027.

Figure 5.8 shows the actual polar position of the plasma over three distinct time periods, as determined by the fitting routine. The first time period, represented by green, is during the formation period of the Z-pinch. The next time period is during the quiescent (stable) period, and is represented by black. Notice how the plasma forms and then stays on axis. Finally, the last time period, plotted in red, represents the beginning of the unstable period. The black circle shows the position of the outer electrode.

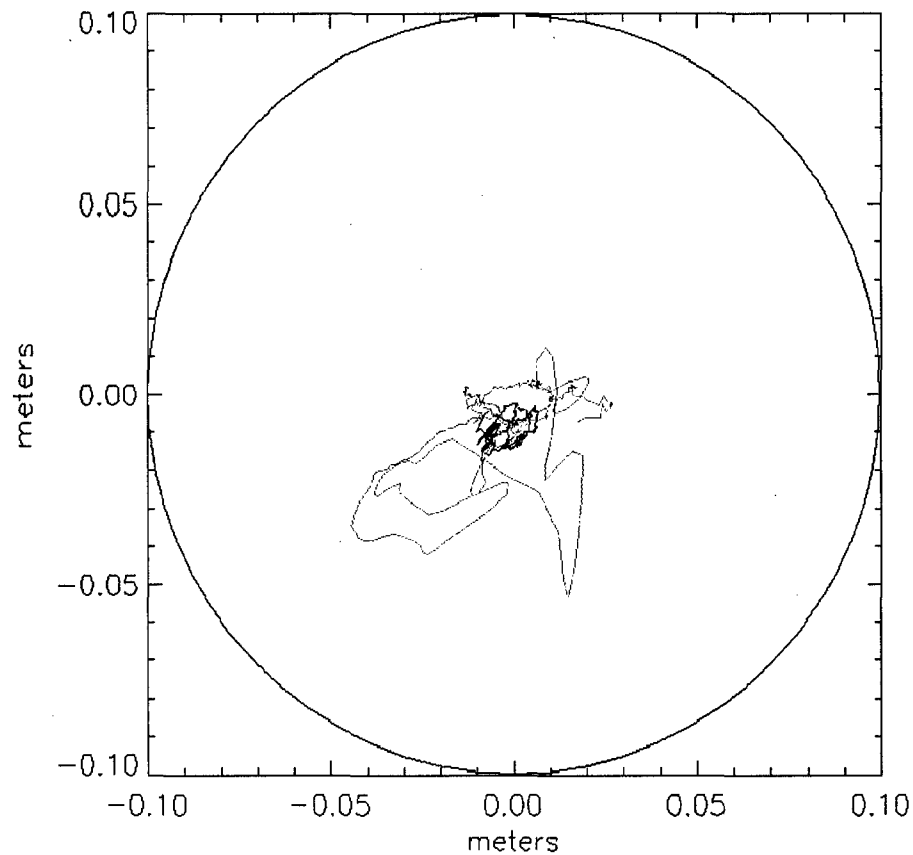


Figure 5.8: Time evolution of the plasma centroid, Pulse 726025. The green plot is from $20\ \mu\text{s}$ - $28\ \mu\text{s}$, the black plot is from $28\ \mu\text{s}$ - $36\ \mu\text{s}$, and the red plot is from $36\ \mu\text{s}$ - $44\ \mu\text{s}$.

Figure 5.9 shows a plot detailing the structure of the plasma. In this case, the plot shows the separation of the plasma filaments versus time. Notice that the frequency of oscillation of separation distance increases as the stable period ends.

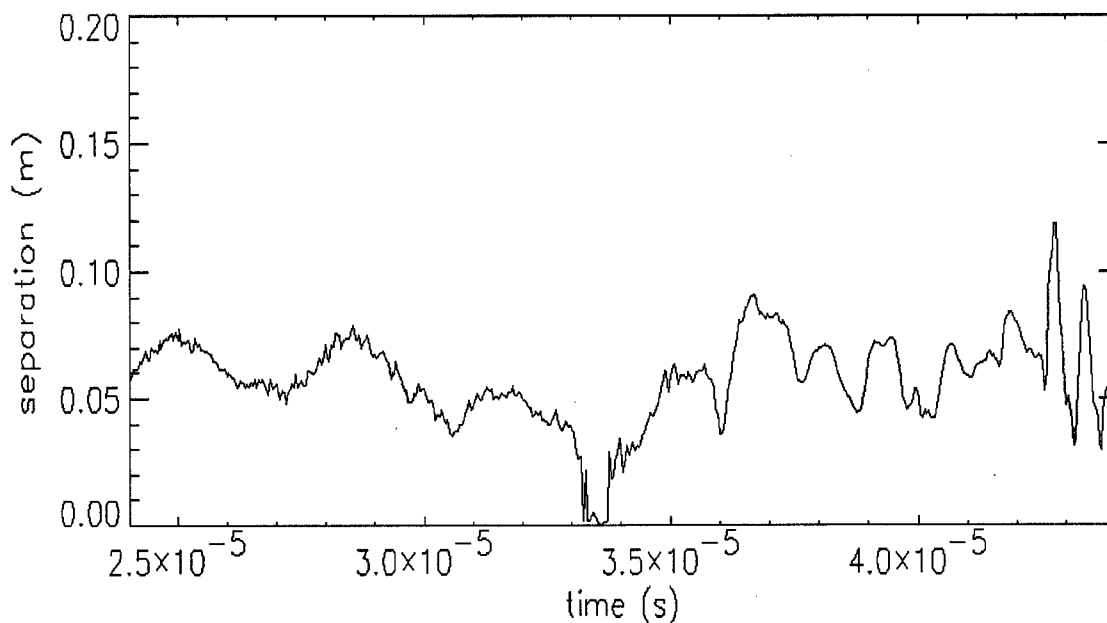


Figure 5.9: Evolution of the separation of the plasma filaments as calculated by the computer program.

Discussion of Results

As can be seen above, the results from the fitting routine are accurate. There are occasional lapses when the routine fails to converge—in Figure 5.6, for example, when the fit returns a value of zero at 60 μ s. This typically occurs when there are extremely steep gradients in the $m=1$ mode (such as at 18 μ s or 60 μ s). These are isolated incidents,

however, and do not affect the overall performance. In general, the fitting routine works quite well on a consistent basis.

Unfortunately, there are some major drawbacks to this routine. Primary among these is the fact that the routine often takes as long as five minutes to run. The routine must do multiple iterations at every time point—of which there are thousands—so the routine takes a considerable amount of time to run. As the program was designed to be used with every shot of the experiment, this computation time is unacceptable.

Another issue with the program arises from the data that it can use. The mathematics of the fitting routine are based off of an infinite line of current. This assumption does not always hold true within the actual experiment; after the stable period, the plasma breaks down and no longer represents an infinite line of current. This presents problems for the fitting routine. Often, during this time period, a magnetic probe will record a negative value, while the other probes still record a positive value. This indicates that the magnetic field in the theta direction is not consistently in one direction. Such readings throw off the non-linear fit, resulting in non-convergence or unphysical answers.

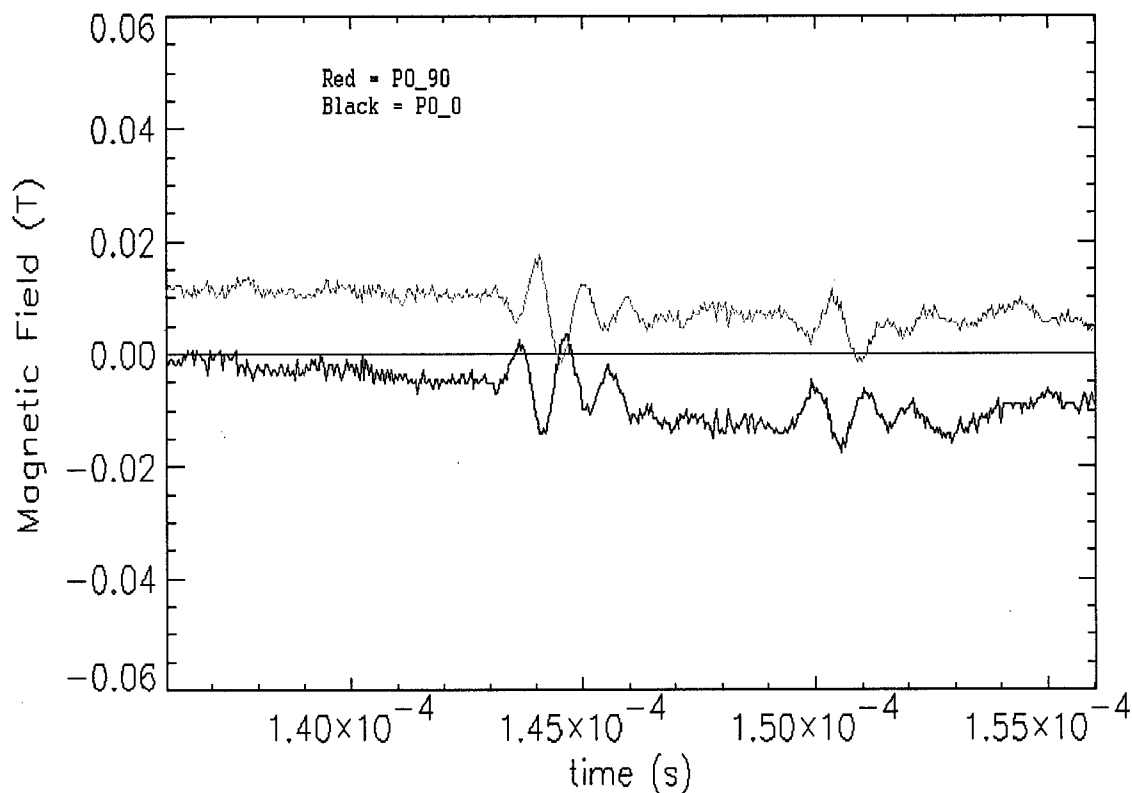


Figure 5.10: A negative magnetic field at 0° and a positive field at 90° during the same time period, Pulse 726025

A similar problem is apparent in the time period before the current sheet passes the magnetic probes. During this time, the data acquisition system only records noise. If the non-linear routine attempted to fit during this time, the results would be meaningless and would negatively affect calculations at future time points.

To overcome these problems, the computer program is designed to seek out only data points with eight positive magnetic probe readings (see Appendix C for the IDL code). Furthermore, it will not begin fitting until all eight probes have reached a value

greater than 0.03 T. This allows the program to skip over noise in the beginning.

Although these rules allow the program to run without problems, they add an undesirable programmer bias to the routine. Ideally, the program would analyze all the data points without any subjective searching.

Overall, the program runs well, but could use improvements. The largest stumbling block is the time the program requires to run. The prohibitive time requirements led us to look for an alternative method for calculating positions, which would be faster and more reliable. Eventually, we decided to replace the program with a neural network.

Chapter 6: The Neural Network

Background of the Neural Network

Artificial neural networks were initially designed in order to mimic the problem-solving style of the human brain. The human brain, unlike traditional computers, is massively parallel, relying on countless inputs to form an answer. Computers, on the other hand, are very linear. While computers can perform specific tasks (mathematics is a typical example) incredibly faster than the human brain, the human brain is much more adapt at tasks such as pattern recognition. One of the primary reasons that the human brain is better at pattern recognition is that it is capable of accepting a large number of inputs, weighing each differently based on the situation. These inputs are often highly interconnected, combining for a total picture.⁹

The mathematical neural network attempts to copy this style of problem-solving. Neural networks are parallel in design, with many interconnected inputs carrying different weights. These weighted inputs can be processed in a non-linear manner, imitating the biological neural network found in the human brain. Using this parallel, interconnected method, mathematical neural networks become a fast, flexible method for pattern recognition.

One important branch of pattern recognition is non-linear function representation. The neural network sees a set of inputs and—using pattern recognition—determines a corresponding set of outputs. The neural network has no knowledge of the actual physical functions involved, nor does it require any iterative process. Thus, neural

networks are ideally suited to computationally intense fitting routines, such as that created for the magnetic probes.

The Neuron

At the heart of any neural network is the neuron. The mathematical neuron attempts to model the biological model closely (as well as we understand it). The main features of a biological, and thus mathematical, neuron, are:

- 1) The output from the neuron can be either on or off.
- 2) The output depends only on the inputs.
- 3) The inputs are weighted depending on their importance.¹⁰

A model of the mathematical neuron follows.

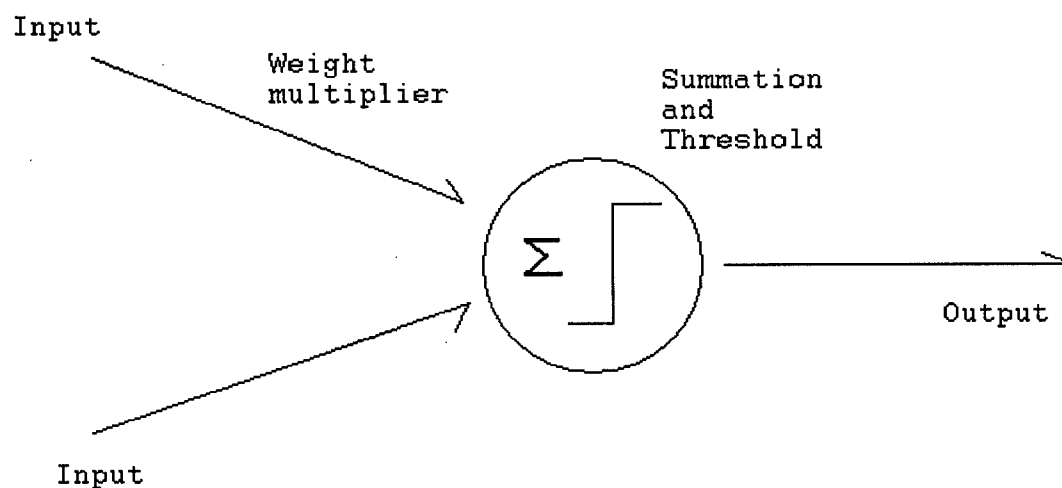


Figure 6.1: Mathematical model of a single neuron.¹¹

The neuron accepts the weighted inputs and then performs a summation. If the total is greater than a preset threshold value, the neuron turns on. This is known as a linear “feedforward” neuron. In this way, the neuron imitates a biological neuron, which accepts many different inputs but only fires if the inputs pass a certain threshold.

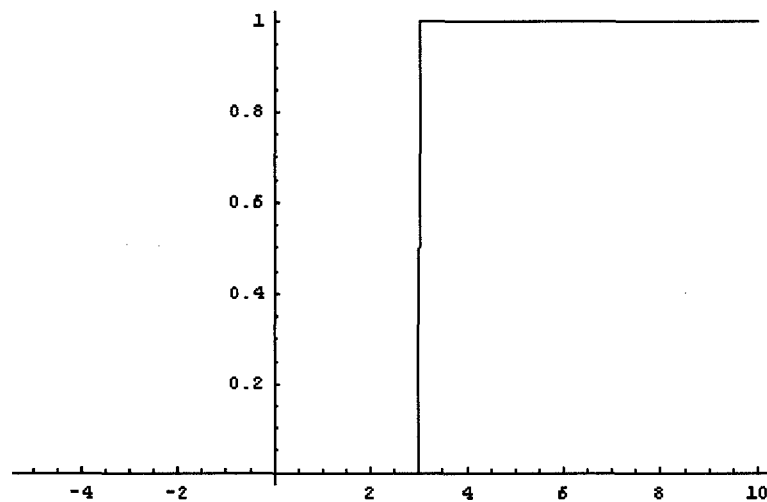


Figure 6.2: Threshold function at $x=3$.

Subtracting a bias from the weighted input also simulates the threshold. In the example above (Figure 6.2), we could subtract 3 from the total weighted input. The output for the mathematical model would then be:

$$y = f_h \left[\sum_{i=1}^n w_i x_i - b \right], \quad (37)$$

where f_h is the Heaviside function, w is the weight for each input x , and b is the bias.

Equation 37 is the original model of the mathematical neuron, developed by McCulloch and Pitts in 1943.¹²

This model is limited in one important aspect: the mathematics are entirely linear. This linear aspect greatly reduces the usefulness of the neuron and thus the entire neural net. In order to create a more useful neural net, it is necessary to replace the linear Heaviside function with a non-linear function. Typically, a sigmoid function is used. The sigmoid function closely resembles the Heaviside function in shape, but is non-linear in nature. This non-linearity allows a greater degree of flexibility. The output function for a neuron remains the same as in Equation 37, but the function f_h is now a sigmoid function. In Figure 6.3 below, the sigmoid function is the “tansig” function, which is defined as

$$\text{tansig}(x) = \frac{2}{1 + e^{-2x}} - 1. \quad (38)$$

The tansig function is mathematically equivalent to the hyperbolic tangent function, but it is quicker (computationally) to explicitly use Equation 38.

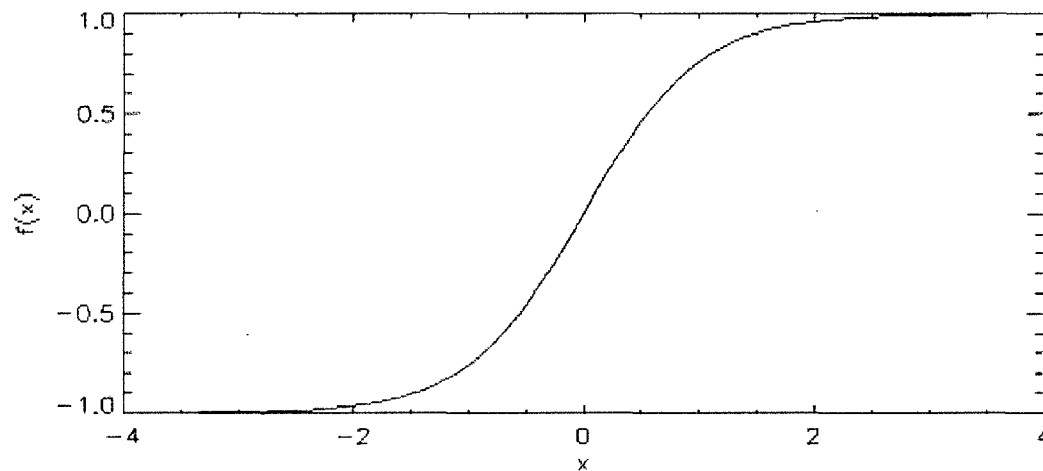


Figure 6.3: The 'tansig' function, which is the same as the hyperbolic tangent function.

Neural Network

One of the most important aspects of pattern recognition is the use of multiple, inter-connected inputs. To represent this feature in a mathematical neural network, we use multiple neurons. This model is called the "multi-layered perceptron". The neurons all accept the same inputs, but the weights and biases are different for each neuron. The number of neurons determines the number of outputs. Thus, neural networks can receive any number of inputs and reduce them to a desired number of outputs. This feature of neural networks makes them especially useful for applications such as probes and sensors.

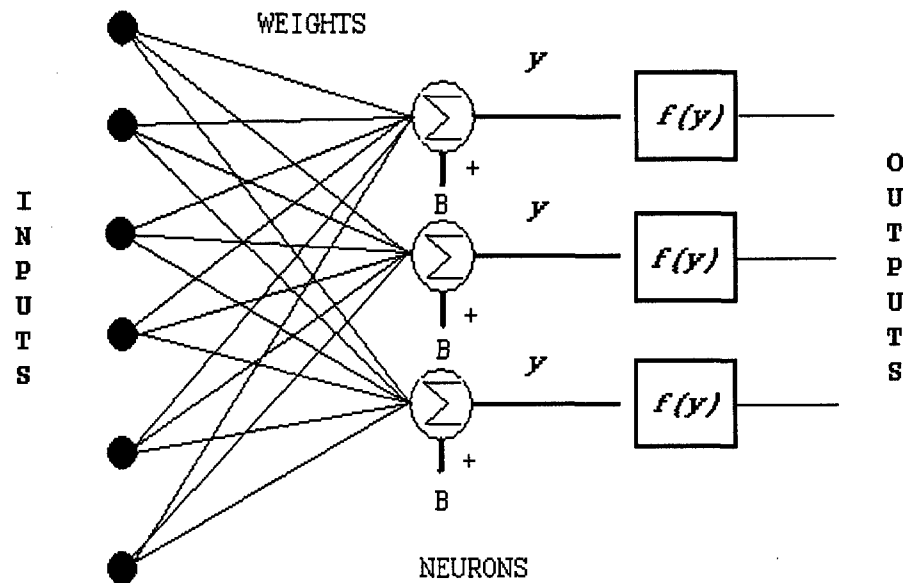


Figure 6.4: The weights for each input are unique for every neuron, as shown by the lines from each input to each neuron.

Neural networks are not limited to a single layer of neurons. Often, neural networks are designed such that the outputs from one layer of neurons become inputs for the next layer. Each layer can have different transfer functions, $f(y)$, and will have a different set of biases and weights. Each layer can have an independent number of neurons. Although any single layer of neurons can have an infinite number of neurons, it is sometimes helpful to have multiple layers of neurons with the same transfer function, as opposed to one massive layer. The multiple layers can serve to pick out different details of the pattern, with the first layer differentiating global patterns and the second layer determining local patterns.

Feed-Forward Backpropagation Multi-Layered Perceptrons

As mentioned above, neural networks are useful in non-linear regression problems. The Feed-Forward Backpropagation (FFBP) network is especially adept at this problem. The FFBP is a two-layer network that can theoretically represent any function to an arbitrary degree of accuracy, given an arbitrarily large number of neurons. The first layer of the FFBP has non-linear functions as their transfer functions. Usually the tansig function is used. The number of neurons in this layer depends on the complexity of the function and the desired accuracy. The second layer has the same number of neurons as the desired number of outputs. This layer is usually a pure linear function. This linear layer allows the outputs of the tansig function (which are limited to a range of $-1 \leq y \leq 1$) to be mapped onto final outputs with an infinite range.

Figure 6.5 shows the basic layout of a FFBP. $\mathbf{R}$ is a vector of R inputs, S is the number of neurons in the hidden layer, and $\mathbf{n}$ is a vector of n outputs. $\mathbf{IW}\{1,1\}$ is the matrix containing the initial set of weights, $\mathbf{LW}\{2,1\}$ is the second weight matrix, and $\mathbf{b}\{1\}$ and $\mathbf{b}\{2\}$ are the bias matrices.

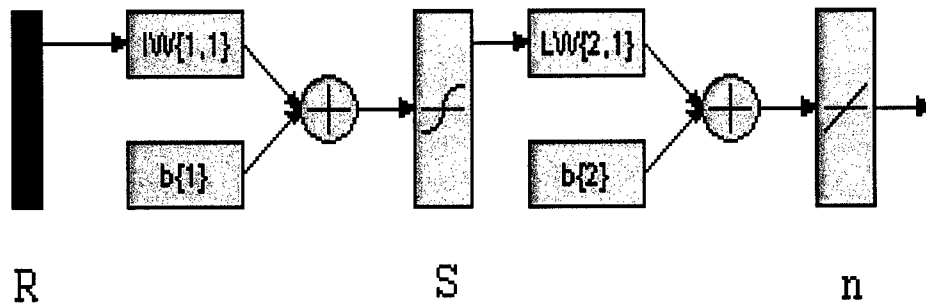


Figure 6.5: The layout of a FFBP network shows two distinct layers with different transfer functions.

There is a unique value for each weight between every input and every neuron, as represented in Figure 6.4. Mathematically, this corresponds to the weight matrices marked as **IW** and **LW** in Figure 6.5. The size of these matrices depends on the number of inputs and the number of neurons. In Figure 6.5, **IW** is a matrix of size $S \times R$ and **LW** is $n \times S$. The vector **b{1}** is $R \times 1$, while **b{2}** is $n \times 1$ ¹³. The output from layer 1 is:

$$y_1 = f_1(\text{IW}\{1,1\} \cdot R + b\{1\}) \quad (39)$$

and the output from layer 2 is:

$$y_2 = f_2(\text{LW}\{2,1\} \cdot f_1(\text{IW}\{1,1\} \cdot R + b\{1\}) + b\{2\}), \quad (40)$$

where f_1 is the tansig function and f_2 is the pure linear function.

Application to the Magnetic Probe Problem

As the FFBP network is well-suited for complex function representation, it is an ideal replacement for the non-linear iterative regression program that was described in Chapter 5. The neural network is inherently fast as there are no iterations to perform. The only mathematical operations that are performed are matrix multiplication/addition and the use of some simple functions. The data can be presented to the neural network as one giant matrix; in the case of the magnetic probes it is possible to combine all 16,383 vectors into one matrix and feed this into the neural network.

Another advantage that the neural network presents is that there is no need for initial guesses or data selection. The fit at any time is independent of the previous fits. The neural network can handle any input—if the inputs are unusual, such as in Figure 5.10, the neural network may return a non-useful answer. It will not, however, crash the program or cause further solutions to be skewed. Since the neural network can handle all data, it is not necessary to go through the data and select a starting and ending point for the data that is going to be processed.

Training and Implementing the Neural Network Using MATLAB

MATLAB was used originally to train and develop the neural network. MATLAB supports a toolbox called the Neural Network Toolbox (version 4.0), which allows users to specify attributes of the neural network through a Graphical User Interface. This interface allows the user to make quick changes and visualize these

changes immediately. It is thus not necessary to write new code for every desired change.

One of the most useful features of neural networks is their ability to learn patterns and adapt to new ones. In order to learn, however, the neural network must be “trained”. The most time consuming and difficult part of implementing a neural network comes in training the network. Training is simple to understand but complicated (mathematically) to implement. At the basic level, we want to adjust the weights and biases of the individual neurons such that when a set of inputs is received, the neural network returns the correct outputs. As the weights/biases are usually initialized to a random value, we want a learning method that adjusts the weights and biases so that the error decreases with each training iteration. This is the basic task that is performed when training a neural net.

Training progresses in the following manner:

- 1) A set of inputs is fed to the neural net.
- 2) The outputs from the neural net are compared to the desired outputs, or “targets.”
- 3) The weights and biases of the neurons are adjusted based on the Mean Squared Error (see Equation 41).
- 4) The inputs are again fed to the neural net, and the process repeats itself.

The set of inputs and desired outputs is called the “training set” or “target set”. Ideally, this set will cover the entire input space, giving the neural net examples of all possible inputs it will see. Since there may be an infinite number of inputs possible, we usually

present the neural net with a training set that covers the input space as evenly and densely as possible.

The training set for the magnetic probe problem was created using the computer program described in Chapter 4. As the computer program is designed to systematically move the plasma filaments throughout the entire space of the test section, this program provides an excellent training set for the neural net. The inputs to the neural net are the eight normalized magnetic probe readings. The outputs are two pairs of Cartesian coordinates (one pair for each plasma filament). Originally, the outputs were two radius-angle combinations, but the neural net performed better when the targets were Cartesian coordinates. This anomaly will be explained later.

Creating and training the neural network became an interesting balancing game between the number of neurons and the number of training set inputs. Increasing the neurons returns an increase in accuracy, as the neural net has more flexibility. However, with too many neurons, the neural net will "overfit", a process by which the neural net fits the target set inputs accurately, but returns a grossly inaccurate answer for any points not seen in the training set.

The number of inputs on the training set required a balance as well. If there were not enough targets, the neural net would overfit to the few examples it saw, but would not learn the overall pattern. If too many targets were given, the neural net might not have enough neurons to learn the pattern. In this case, the net would over-generalize, creating a rough solution that would minimize error, but never accurately learn the desired pattern.

Training Techniques

Training a neural net requires the user to adjust the weights and biases of the neurons in order to reduce the error between output and target. The neural net that we created uses the Mean Squared Error as a performance guide.

$$Error = \frac{1}{N} \sum_i^N (y_i - y'_i)^2. \quad (41)$$

The training methods used by MATLAB analyze the gradient of the error function, moving the weights and biases in the direction of decreasing error. MATLAB training methods typically invoke “momentum,” a process that allows the algorithm to pass over small bumps in the error surface and seek out the global minimum.

The magnetic probe neural net was trained with the Levenberg-Marquardt (L-M) algorithm. The L-M algorithm was the fastest and most successful algorithm out of the many that were tested. The algorithm is a faster version of the quasi-Newtonian algorithms. The basic Newtonian algorithm updates the weights as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{A}_k^{-1} \mathbf{g}_k, \quad (42)$$

where $\mathbf{x}$ is the vector of weights and biases, $\mathbf{A}$ is the Hessian matrix (second derivatives of the error function) and $\mathbf{g}$ is the gradient. The L-M method estimates the Hessian matrix $\mathbf{A}$ with an approximation, $\mathbf{H}$.

$$\mathbf{H} = \mathbf{J}^T \mathbf{J}, \quad (43)$$

where $\mathbf{J}$ is the Jacobian matrix of first derivatives of the error function. L-M estimates the gradient, $\mathbf{g}$, as

$$\mathbf{g} = \mathbf{J}^T \mathbf{e}, \quad (44)$$

where $\mathbf{e}$ is a vector of network errors. Finally, L-M adds a scalar term, μ , to adjust the weight of the algorithm from the Hessian matrix to the gradient. This scalar can change depending on the error surface. The final algorithm is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}]^{-1} \mathbf{J}^T \mathbf{e}. \quad (45)$$

The key to the speed of the L-M method lies in the estimation of the Hessian matrix with the Jacobian matrix, which is much faster to calculate. However, the use of a Hessian approximation allows the algorithm to approach second order accuracy.¹⁴

The MATLAB user interface can be configured to run the L-M algorithm automatically. Thus, the training of the neural net was performed automatically by the MATLAB interface, using the L-M method. Below is a typical graph depicting the training progress on the neural net. The target accuracy for the net was 10^{-6} , which corresponded to a positional accuracy of 0.001 meters for the plasma filaments. Notice the leveling of the error curve near the last epochs. This feature was seen in all the training attempts; the challenge lay in adjusting the neurons and targets such that the error curve leveled off as close to 10^{-6} as possible.

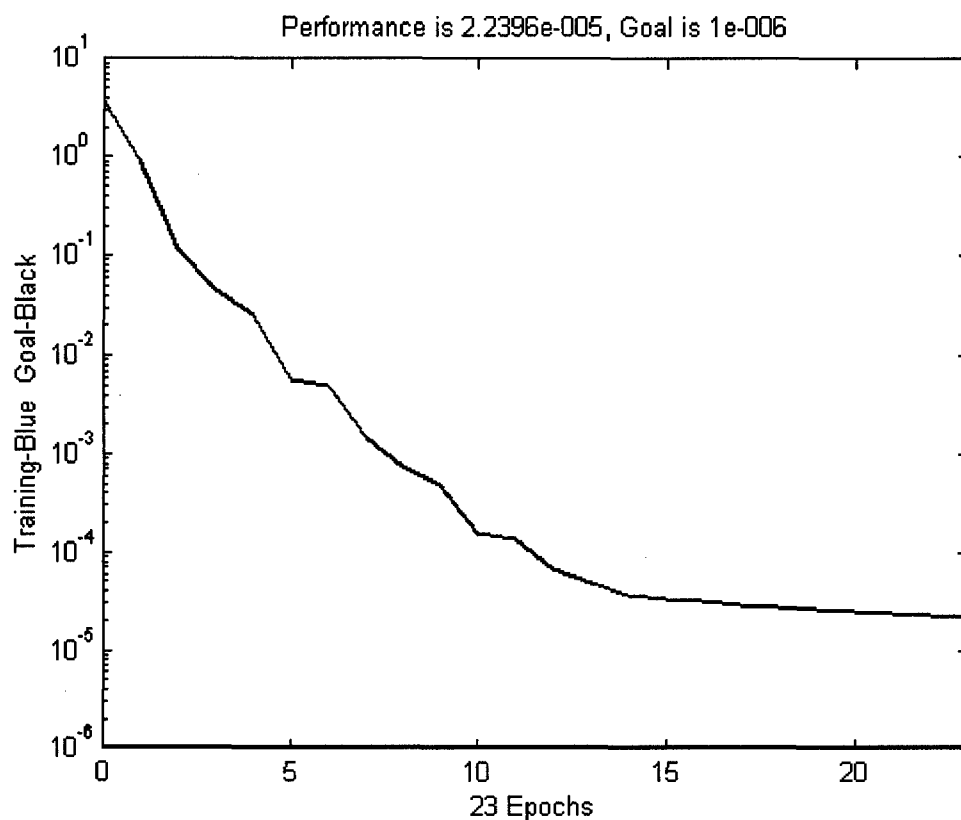


Figure 6.6: Typical progress while training the neural net. Some training sessions covered 500 epochs, while others stopped at 10, depending on the structure of the net.

Final Structure of the Magnetic Probe Neural Net

After a great many training sessions, it became obvious that the normal FFBP structure was not going to properly model the magnetic probe problem. The training sessions would not bring an accuracy of greater than 10^{-4} . Increasing the number of neurons might increase the accuracy slightly above 10^{-4} , but the result would be overfitting of the target set. A different approach was needed.

The breakthrough in the neural net training came in deciding to add another layer to the basic two-layer FFBP structure. An additional layer, with the same sigmoid transfer function as the first layer, would allow the neural net to break the problem into global and local patterns. Furthermore, due to the cross connections between the first two layers, the extra layer allowed us to create a structure with fewer neurons, yet still retain the accuracy benefits of having more neurons. In effect, having two layers of 25 neurons was equivalent to having one layer of 625 neurons. The final structure is shown below.

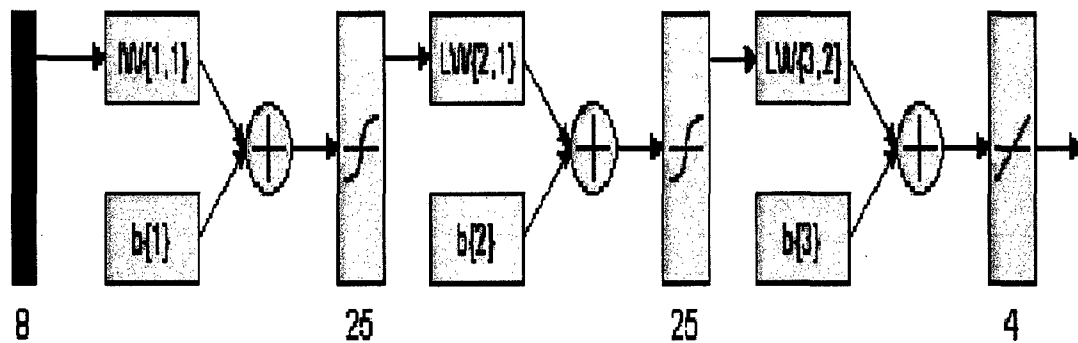


Figure 6.7: The final net structure has three distinct layers, instead the usual two.

Notice that there are eight inputs (the eight magnetic probe readings), with two sigmoid layers of 25 neurons each, and then the final output layer, with four outputs and a pure linear transfer function.

The outputs from the neural net are four Cartesian coordinates, not polar coordinates. It was discovered, during the training process, that the neural net does not perform very well when outputting polar coordinates. The problem lay with the fact that the values of the angles ($0 \leq \theta \leq 2\pi$) are much larger than the possible values of the radii ($0 \leq r \leq 0.1$). Since the error function is Mean Squared Error (see Equation 41), which is not normalized, the error function is heavily weighted toward the angle outputs. The final neural net outputs will roughly match the angle values, but the radial values will be off by more than 100%. The Neural Net Toolbox does include a function that allowed the user to normalize the inputs (and targets, during training) on a scale of $-1 \leq x \leq 1$. This is known as "preprocessing". This process adds several steps of complexity, however, since the outputs have to be "postprocessed" as well. A more efficient manner of bringing the inputs of the neural net into the same scale is to use Cartesian coordinates instead. Thus, we switched the outputs of the neural net to Cartesian coordinates ($-0.1 \leq x \leq 0.1$ and $-0.1 \leq y \leq 0.1$). Now, all four outputs are within the same scale, and all are equally considered during error calculations.

Chapter 7: Results of the Neural Net

Comparison with the $m=1$ Mode

After the neural net has been designed and trained successfully, it still needs to be tested to ensure proper performance. We would like to test the neural net on actual data, so that we can see how well the net performs when there is actual signal noise and so forth.

There are several tests we can run to check the performance of the net. The most basic test is to compare the output of the neural net to the $m=1$ mode of the actual data. From Chapter 4, we know there is a linear relationship between the $m=1$ mode and the offset of the current filaments. This fact is exploited to test the neural net outputs. We can determine the centroid of the current filaments (as calculated by the neural net) and compare this to the $m=1$ mode. If the net is running correctly, they should match.

The following graphs represent several different shots for which the neural net was tested. Each graph shows the output from the neural net versus the $m=1$ mode, calculated directly from the shot data. The mode data was normalized by the average magnetic field and multiplied by R_{WALL} , 0.1 meters.

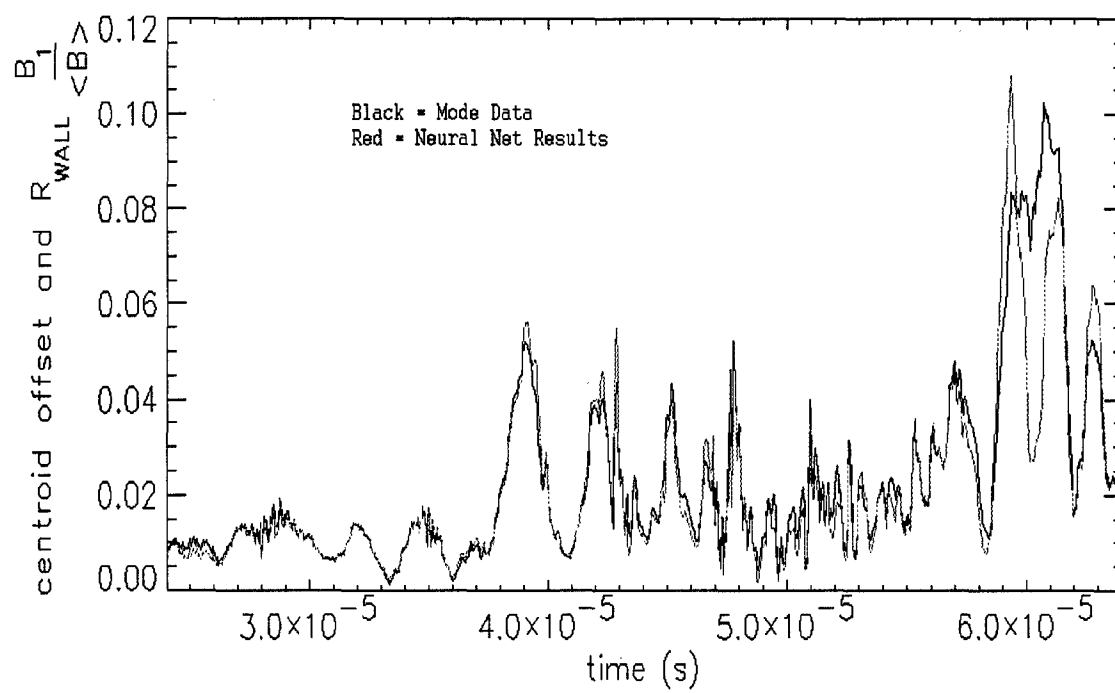


Figure 7.1: Neural net results, Pulse 726025 (8.0 kV).

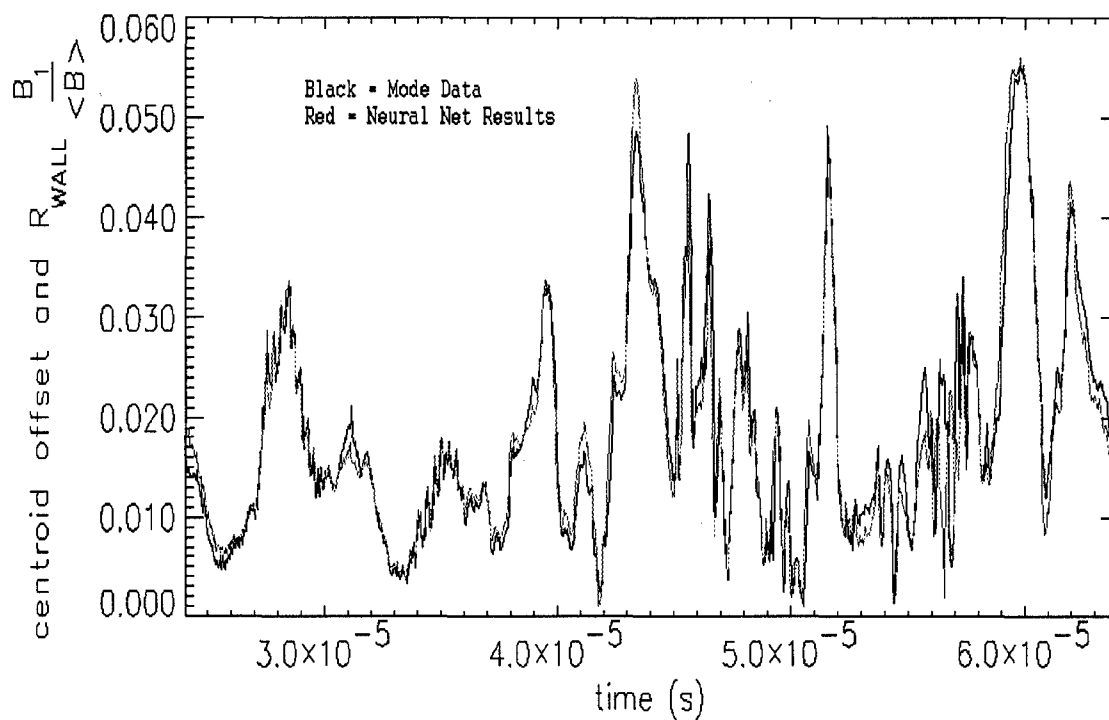


Figure 7.2: Neural net results, Pulse 726027 (8.0 kV).

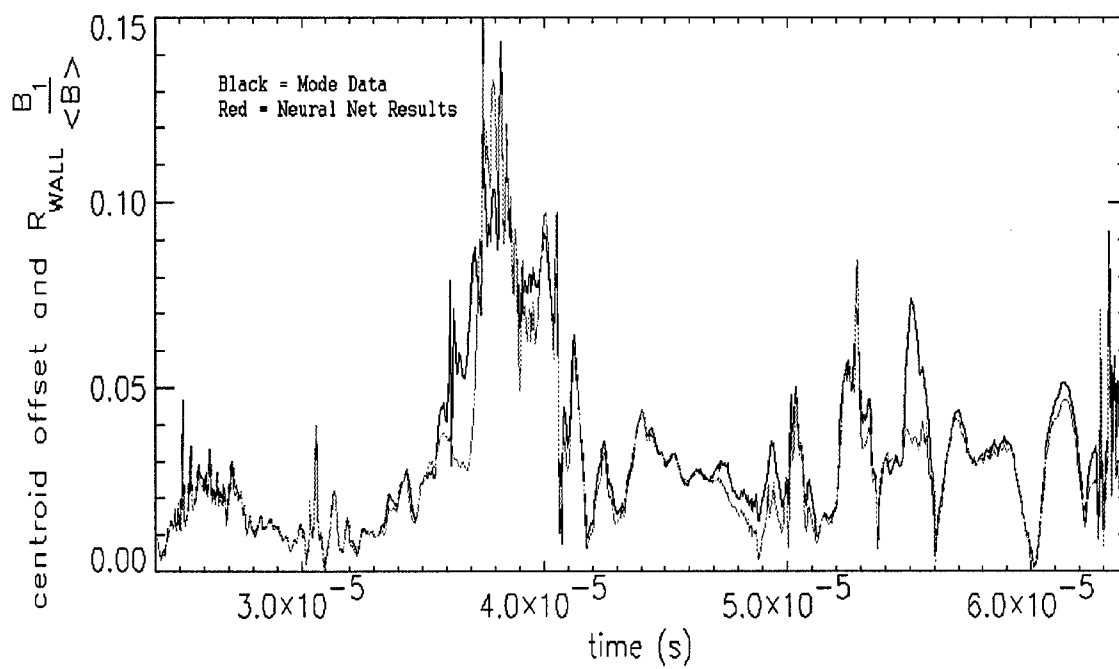


Figure 7.3: Neural net Results, Pulse 10201015 (8.0 kV).

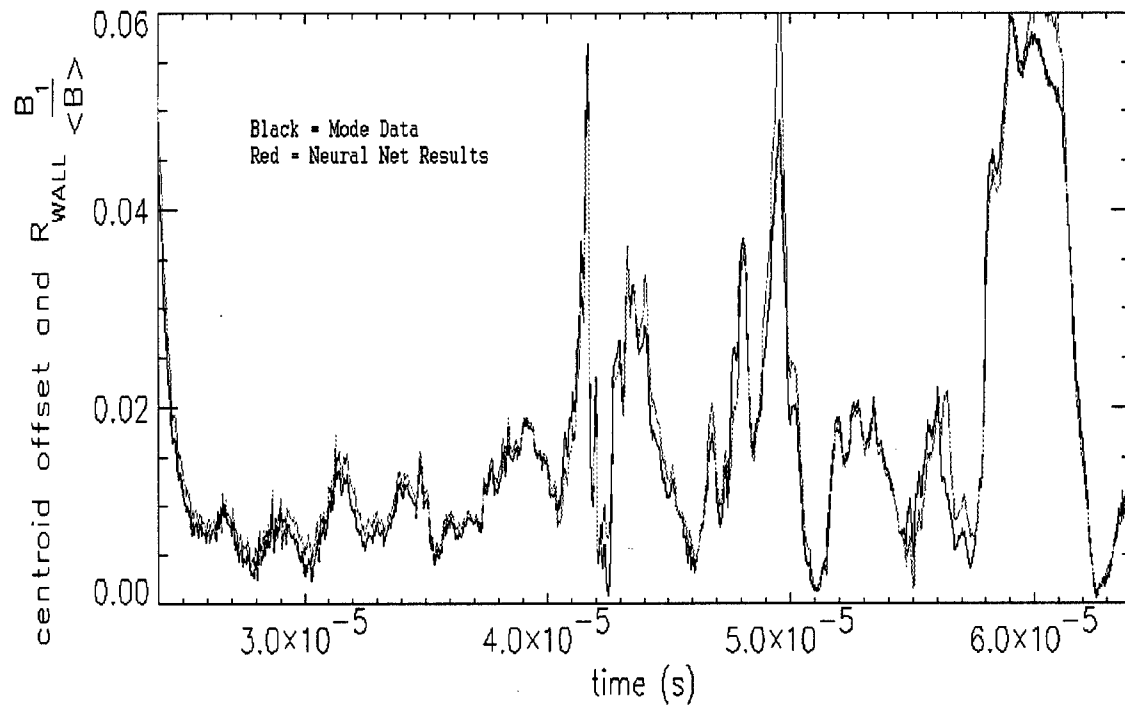


Figure 7.4: Neural net results, Pulse 10306007 (6.0 kV).

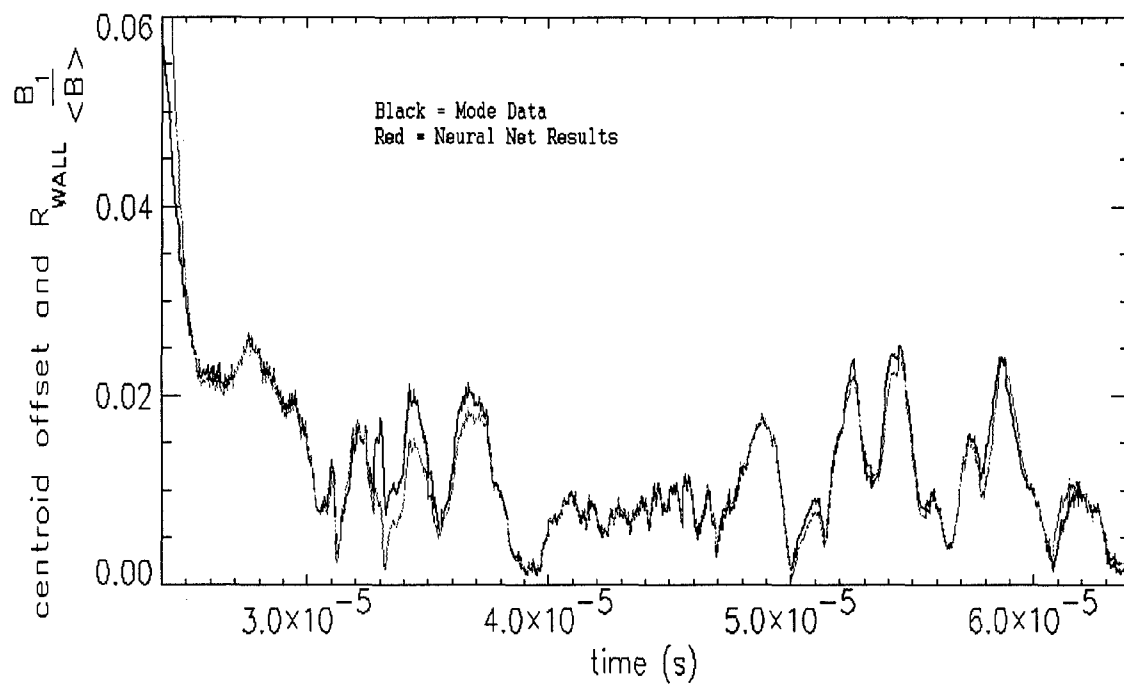


Figure 7.5: Neural net results, Pulse 10417007 (4.0 kV).

The above graphs show shots ranging from 8.0 kV down to 4.0 kV. In all cases, the neural net follows the mode data quite well. There are a few specific exceptions. In Figure 7.3, at roughly 38 μ s, there is a peak where the neural net results diverge from the mode data. At this point it is reasonable to assume the plasma has gone unstable, and so the results from the neural net do not apply. The neural net was trained on a stable plasma only. However, it is likely that the original non-linear fitting routine, described in Chapter 5, would not be able to resolve this data either.

The most important time period is the stable time period (or quiescent period). This period typically lasts roughly 15 μ s, and starts around 20 μ s. The fit during this time period is critical, and the neural net performs best during this period. Figure 7.6 shows a close up of the fit during the quiescent period for shot 10201015.

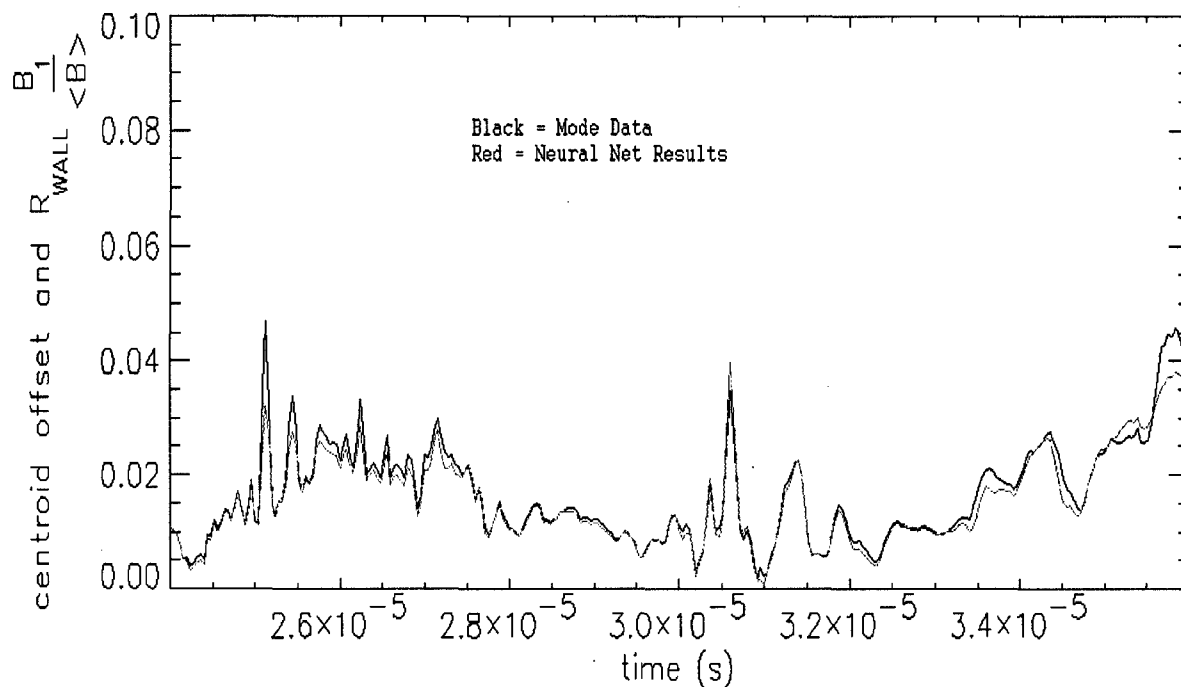


Figure 7.6: Neural net results for quiescent period, Pulse 10201015.

It is interesting to compare the results from the neural net to the results from the non-linear fitting routine. Notice, in the following figures, that the biggest difference between the two is when the non-linear fitting routine crashes (fails to converge). The program will default to a value of 0 for all parameters at this point, so it is easy to pick out on the plots where the program has crashed.

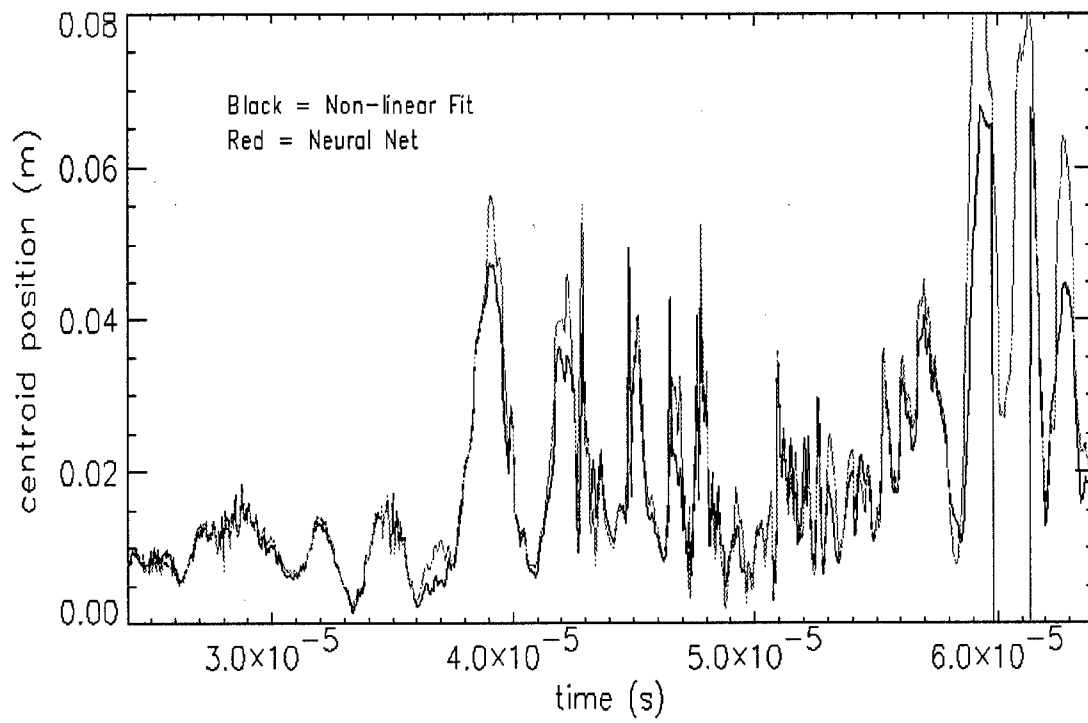


Figure 7.7: Comparison of neural net to non-linear fitting routine, Pulse 726025.

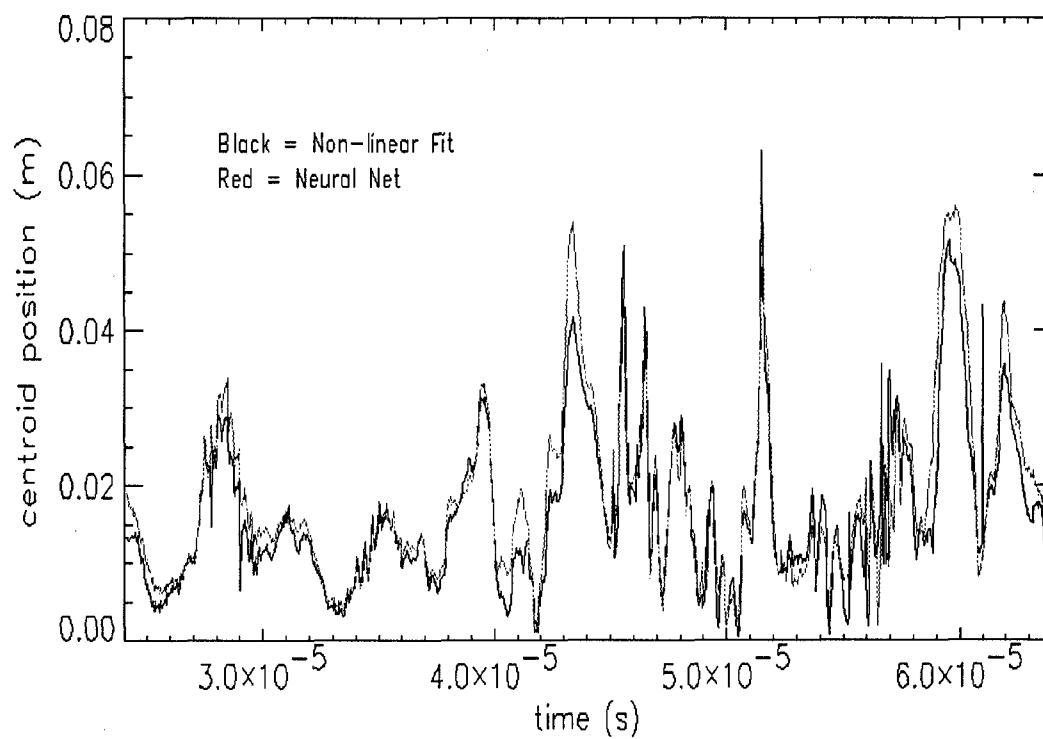


Figure 7.8: Comparison of neural net to non-linear fitting routine, Pulse 726027.

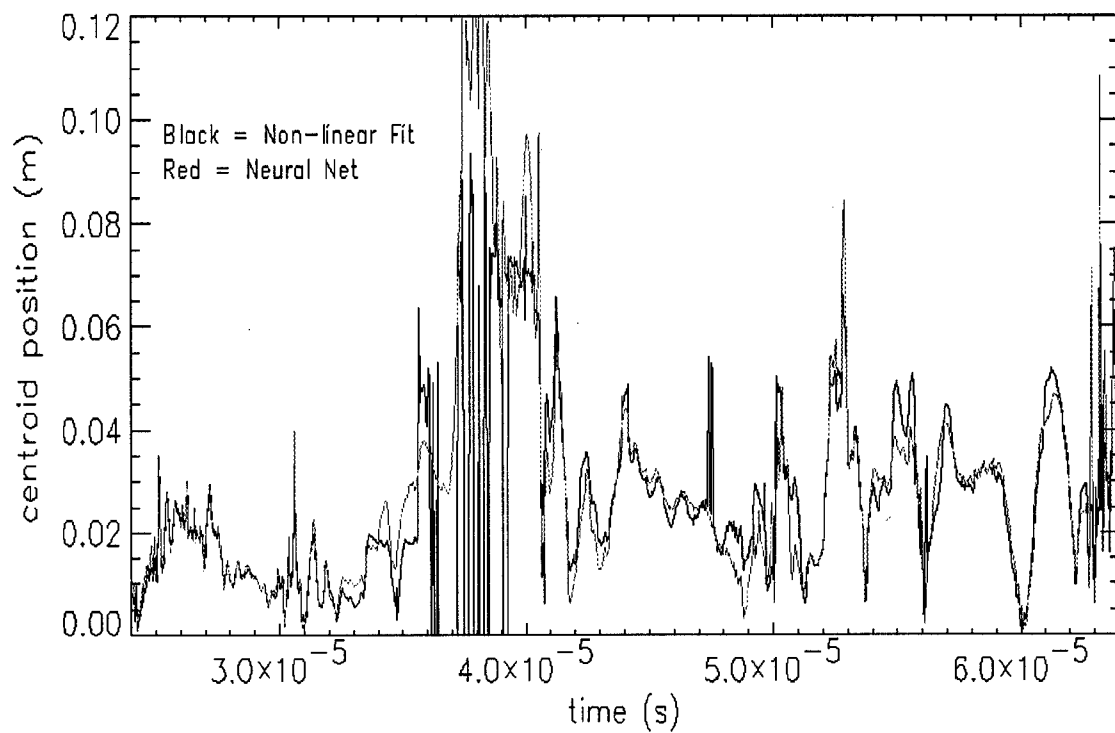


Figure 7.9: Comparison of neural net to non-linear fitting routine, Pulse 10201015.

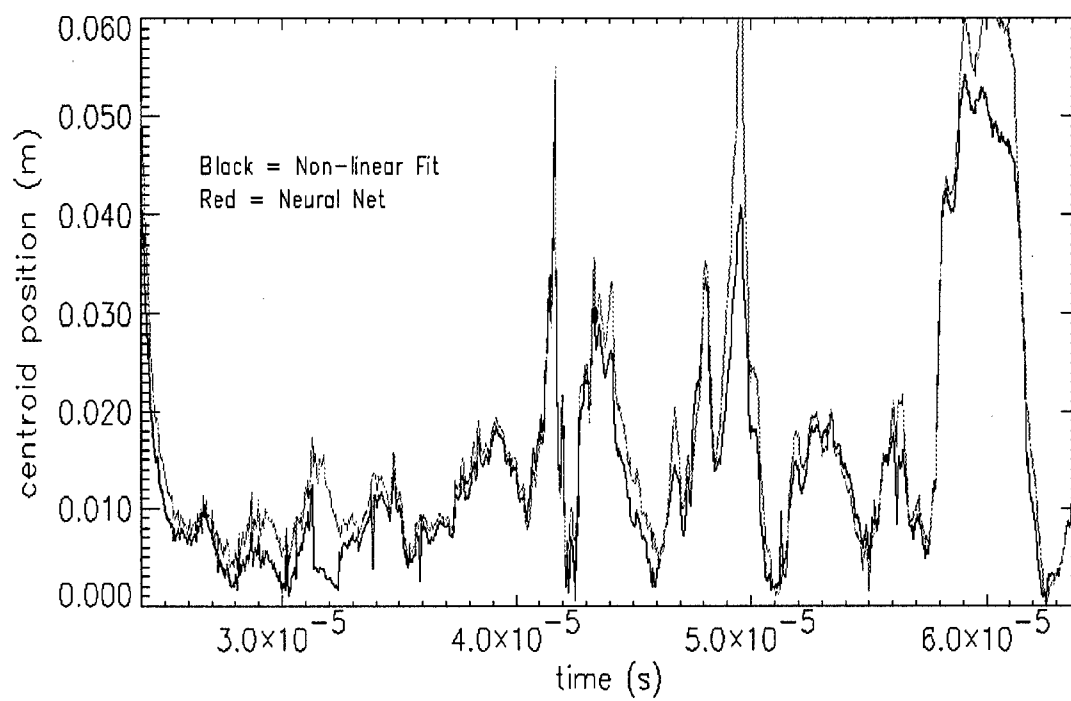


Figure 7.10: Comparison of neural net to non-linear fitting routine, Pulse 10306007.

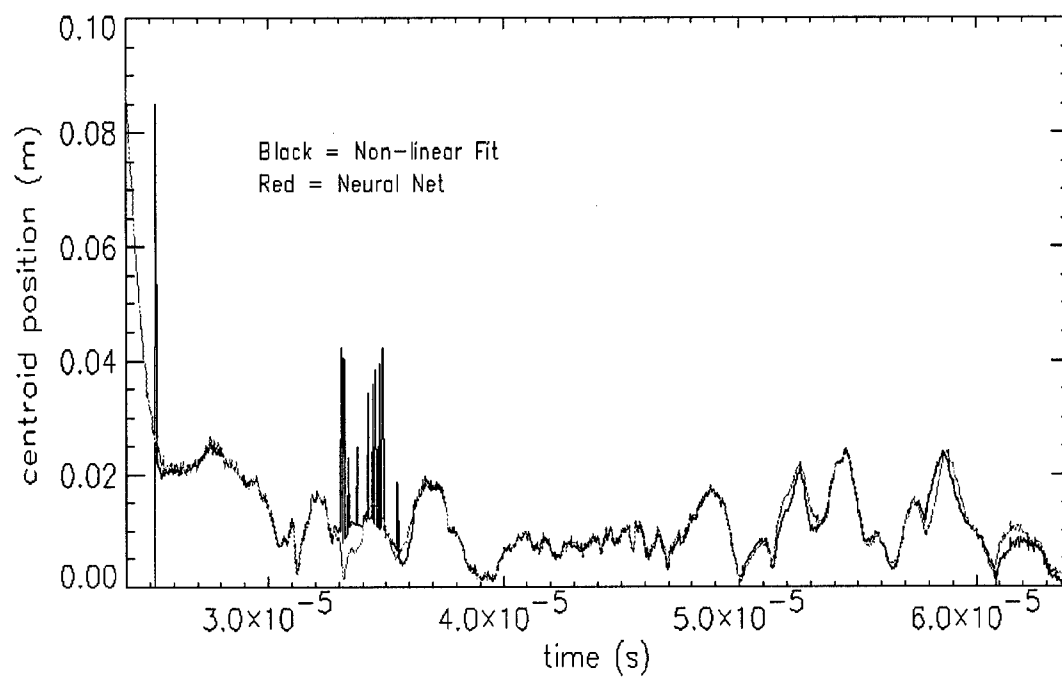


Figure 7.11: Comparison of neural net to non-linear fitting routine, Pulse 10417007.

Error Comparisons

The tests above only look at the $m=1$ component. Since there is coupling between the $m=2$ mode (filament separation) and the $m=1$ mode, the $m=2$ mode data cannot be used to test the neural net results. It is helpful to look at another test to gauge the performance of the neural net. One possibility is to compare the error of the neural net to the error of the non-linear fit. The error can be computed by taking the calculated position of the plasma filaments, and using Eq. 25, calculate the magnetic field at the probes. The following formula is then applied:

$$\left| Error = \frac{B_{ACTUAL} - B_{CALCULATED}}{B_{ACTUAL}} \right|, \quad (46)$$

where B_{ACTUAL} is the magnetic field recorded by the probes and $B_{CALCULATED}$ is the field calculated as described above. All eight errors (one for each probe) are then averaged into one error. The error plots for the same shots as in Figures 7.1-7.5 are shown below.

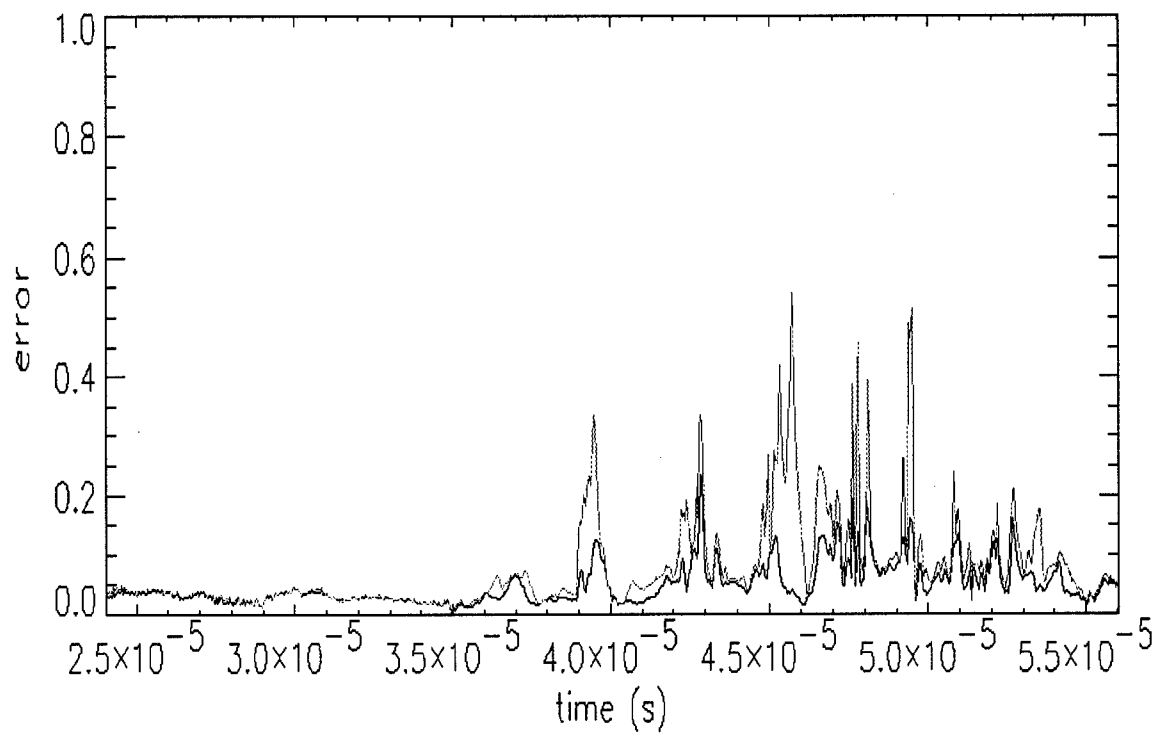


Figure 7.12: Error plot for Pulse 726025. The red line represents the neural net, and the black line represents the non-linear fitting routine.

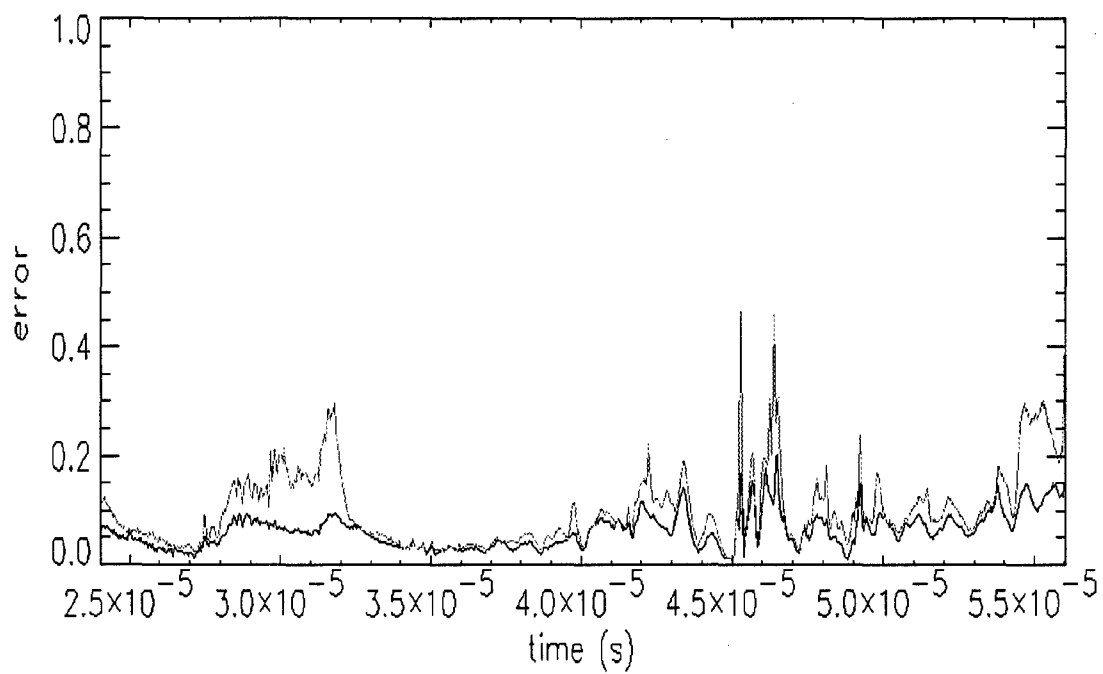


Figure 7.13: Error plot for Pulse 726027. The red line represents the neural net, and the black line represents the non-linear fitting routine.

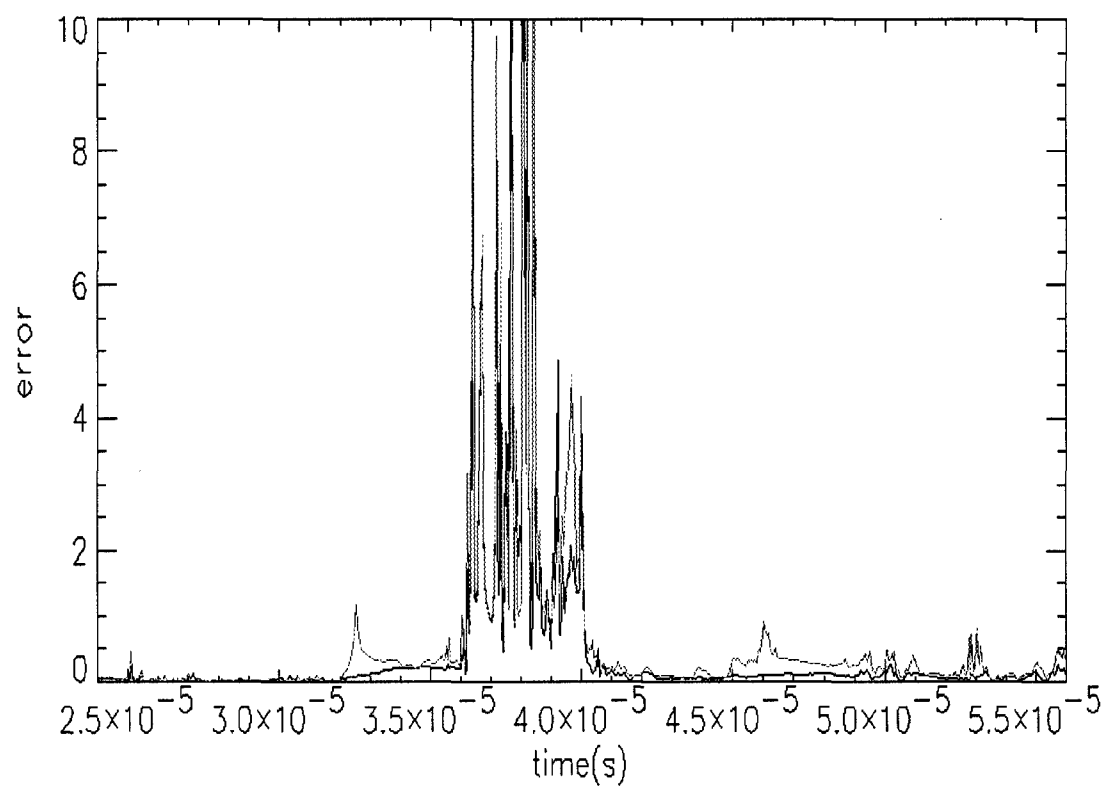


Figure 7.14: Error plot for Pulse 10201015. The red line represents the neural net, and the black line represents the non-linear fitting routine.

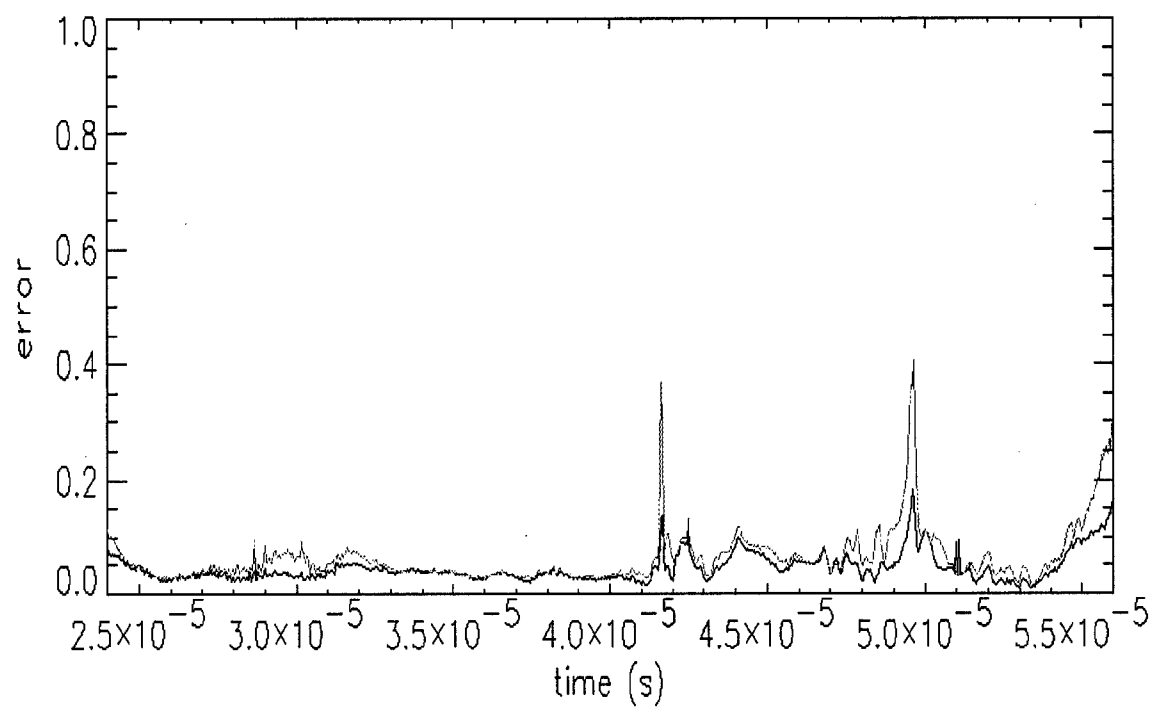


Figure 7.15: Error plot for Pulse 10306007. The red line represents the neural net, and the black line represents the non-linear fitting routine.

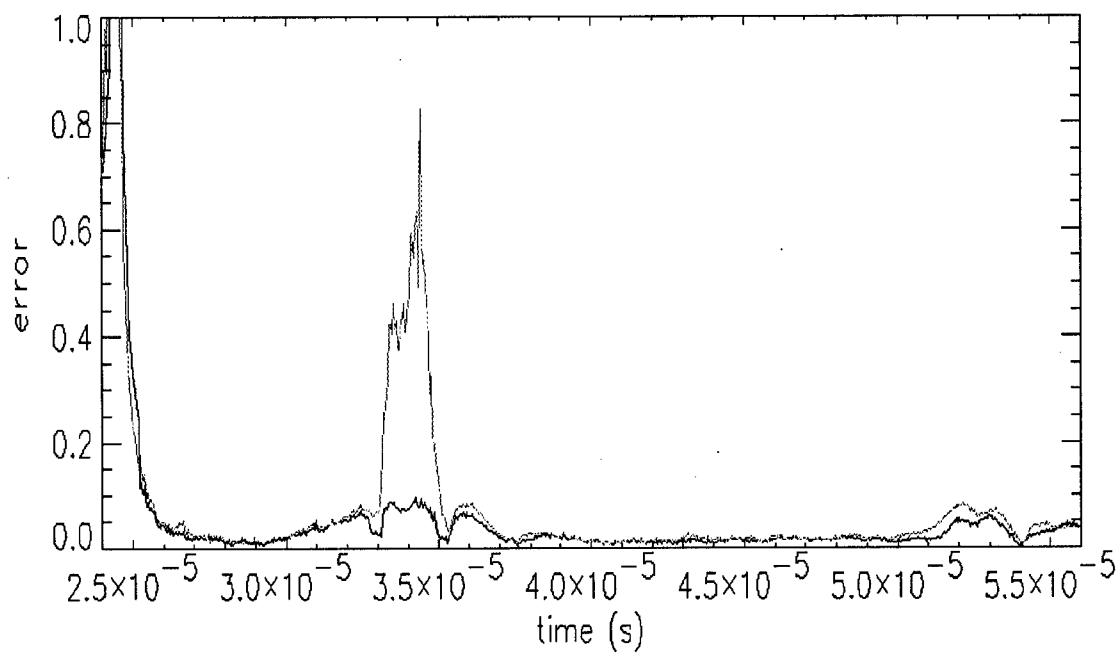


Figure 7.16: Error plot for Pulse 10417007. The red line represents the neural net, and the black line represents the non-linear fitting routine.

Notice that the neural net typically matches the error of the non-linear fit; in general, the pattern of the error is similar. During the stable periods (when the error is low), the error of the neural net is typically lower or equal to that of the non-linear fit. During the unstable period, the neural net is capable of showing large spikes in error. The general trend of the error, however, remains similar to that of the non-linear fit. An interesting study is pulse 10201015. Neither method does a particularly impressive job of fitting in this case. During the stable period (roughly 25 – 33 μ s), the neural net error is lower than that of the non-linear fit. As the unstable period begins, the neural net error spikes rapidly, and then settles into an unpredictable pattern—sometimes the error is greater than the non-linear fit, sometimes it is less. In fact, in most of the shots, the neural net error shows a characteristic jump in error as the stable period ends. Perhaps this marks the first sign of the plasma model breaking down.

Another interesting case is pulse 10417007. The error of the neural net jumps during what appears to be the stable period. There is no immediate explanation for this, although it is helpful to remember that this shot is low energy (4 kV) and thus we may not have a completely formed plasma pinch.

While the error plots show fluctuations in error, the $m=1$ comparison plots typically show very little deviation. This would imply that most of the error is a result of attempting to model the plasma as two current filaments (the $m=2$ mode). The model of two plasma filaments is attempting to represent the elongation of the plasma. As the filaments move further apart, however, the model begins to lose some accuracy. The model assumes that 50% of the current is flowing through each filament. It is likely that

in the real plasma, however, there is some current between. If this is the case, then a large separation should correspond to a large error in the neural net. Figure 7.17 shows the separation of the plasma filaments for shot 10417007. Notice that the large separation corresponds exactly with the large jump in error in Figure 7.16.

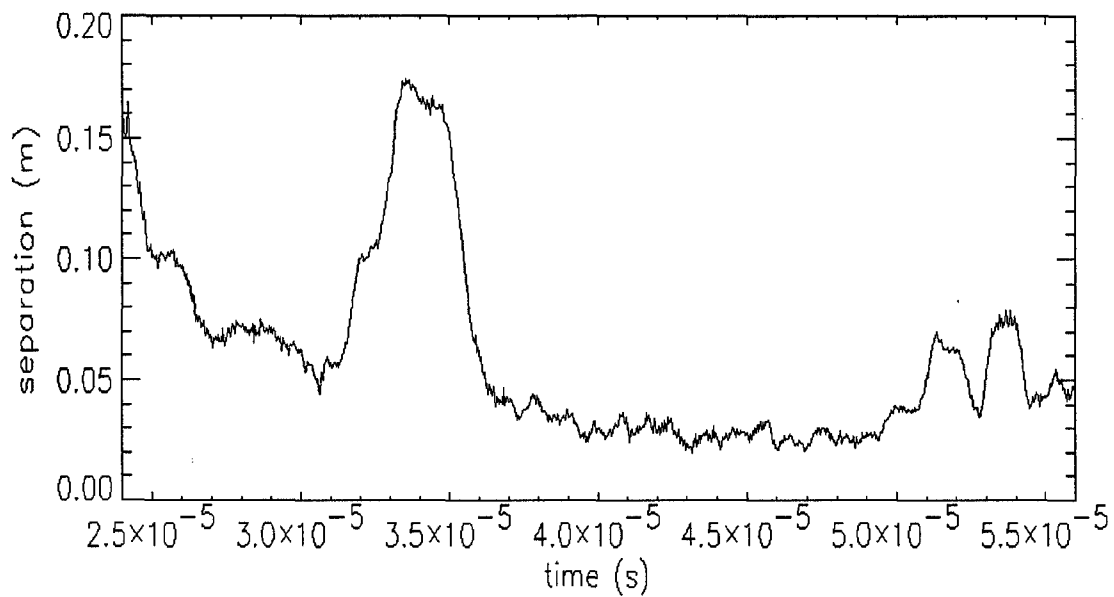


Figure 7.17: Separation of the plasma filaments, as calculated by the neural net, Pulse 10417007. Notice the large separation at roughly 35 μs .

Conclusions

The neural net is designed to match the non-linear fit as closely as possible. The non-linear fit—since it is an iterative process—can minimize error for every time point, even when the plasma is unstable. The neural net, on the other hand, is trained only on stable plasmas, and so will tend to return large errors when the physics no longer hold.

This is not a terrible thing—there is no reason to believe that the results returned by the non-linear fit routine are actually more accurate, when fitting unstable plasmas.

The above tests show that the neural net accurately determines the centroid location of the plasma filaments for almost all time points. When the plasma is stable, it appears as though the neural net returns results as accurate as the non-linear fit. The net shows occasional spikes in the error, even during relatively stable time periods. These spikes may occur when the elongation of the plasma is large.

There are several advantages to using the neural net as well. The run time on the neural net is typically 10 – 20 times faster than that of the non-linear fit. The average run time of the neural net is approximately 30 seconds, while the non-linear fit takes anywhere from 120-600 seconds. Furthermore, the neural net requires no initial guesses. This allows the net to handle a variety of different conditions without any problems. Finally, the neural net can process all data points without crashing. The non-linear fit program requires the data to be filtered so that only usable data points are analyzed. This adds a subjective user bias to the final results.

Overall, the neural net offers a fast and effective way to determine the position of the plasma filaments. The net is fast enough that it can be used as part of the diagnostics evaluated with every experimental run. When conducting a detailed analysis of the plasma position, however, it is critical to consider the error plot for that particular shot. If the neural net shows large error spikes, then the user should consult the non-linear fitting routine. The non-linear fitting routine may have a more accurate answer for that time point (or time period).

Chapter 8: Summary and Conclusions

Summary

The ZaP experiment is studying the stability of sheared-flow Z-pinches. The ZaP experiment hosts a variety of diagnostics, which include a large number of magnetic probes. The information from the magnetic probes can be used to study the magnetic field, as well as other features of the plasma. One important feature that can be determined is the position of the plasma within the experiment.

The magnetic probes must be carefully calibrated in order to guarantee reliable results. Calibrating the probes consists of creating a known magnetic field and then studying the outputs from the magnetic probes. A curve fitting routine can then be used to match the output voltage to the actual magnetic field.

The complicated magnetic field of the ZaP experiment can be represented using a Fourier analysis. Analyzing the mode structure of the plasma allows us to determine different features of the plasma such as position and shape. These relations allow us to analyze computer position codes for accuracy.

A non-linear fitting routine was designed to calculate the position and shape of the plasma. The elongation of the plasma was represented by simulating the plasma as two equal current filaments, each allowed to move independently. Two different geometries were tested in calculating the magnetic field due to the plasma filaments, but it was found that one geometry worked significantly better than the other. The non-linear fitting computer program captures the position of the plasma well during stable time

periods, but may stop working during unstable periods. For this reason, the data must be subjectively searched prior to being fed to the computer program (to eliminate data that would crash the program). The non-linear fit computer program takes a long time to process an entire shot.

A neural net was designed to replace the non-linear fit. In theory, a Feed-Forward Backpropagation (FFBP) network can simulate any non-linear function. Thus, we designed a FFBP to replace the non-linear fitting routine. The neural network had to be trained on an extensive array of target data, using the Levenberg-Marquardt training algorithm. The basic architecture of the FFBP was edited to add an extra layer and thus improve the net's ability to see patterns in the training set.

The neural net was tested on a variety of shots and compared to known data. At all times, it matches the $m=1$ mode (and thus the centroid of the plasma) very well. The net suffers from accuracy problems when the plasma goes unstable, often displaying errors larger than that of the non-linear fit computer program. The net also shows occasional spikes in errors, most likely due to the $m=2$ mode.

Conclusions

The net is accurate and much faster than the non-linear fitting routine. The results are easy to analyze, since they have not been pre-searched or edited. The net is fast enough to be run in tandem with actual experimental shots, processing the data after every shot and returning position information about the plasma. The neural net is accurate during time periods in which the plasma is stable—sometimes more accurate

than the non-linear fitting routine it was designed to replace. The neural net is not always accurate during unstable periods, but during this time it is likely the Z-pinch has broken and the “position” of the plasma has no physical meaning.

Overall, the positional information from the plasma adds an extra dimension of information to other parameters such as density, velocity and visual pictures. Position information allows us to view the evolution of the plasma over time, which helps us understand the stability issues affecting it.

Future Work

Comparison of the neural net results to density calculations, Imacon photos, and ICCD data indicate that the neural net is probably calculating a separation of plasma filaments that is too large. In Figure 5.9, the separation stays around 5.0 cm, whereas all other experimental data suggest that the diameter of the plasma pinch is roughly 2 cm. Although the separation of the filaments might not translate exactly into the diameter of a plasma pinch (the $m=2$ mode indicates elongation, not diameter), the separation of the two filaments should be relatively close to the actual diameter.

There are several approaches to correcting this problem. Analysis of the normalized magnetic field shows that there is very little change (on the order of thousandths of T/T) in the measured signal when the plasma filaments move from a separation of 0 cm to a separation of 3 cm. This value is so small as to be indistinguishable by the neural net. Furthermore, noise in the actual signal is typically larger than this difference.

The following figure shows the normalized magnetic field versus separation. Notice that the change in normalized magnetic field is only 10% for the first 4 cm, which is possibly within the range of signal noise.

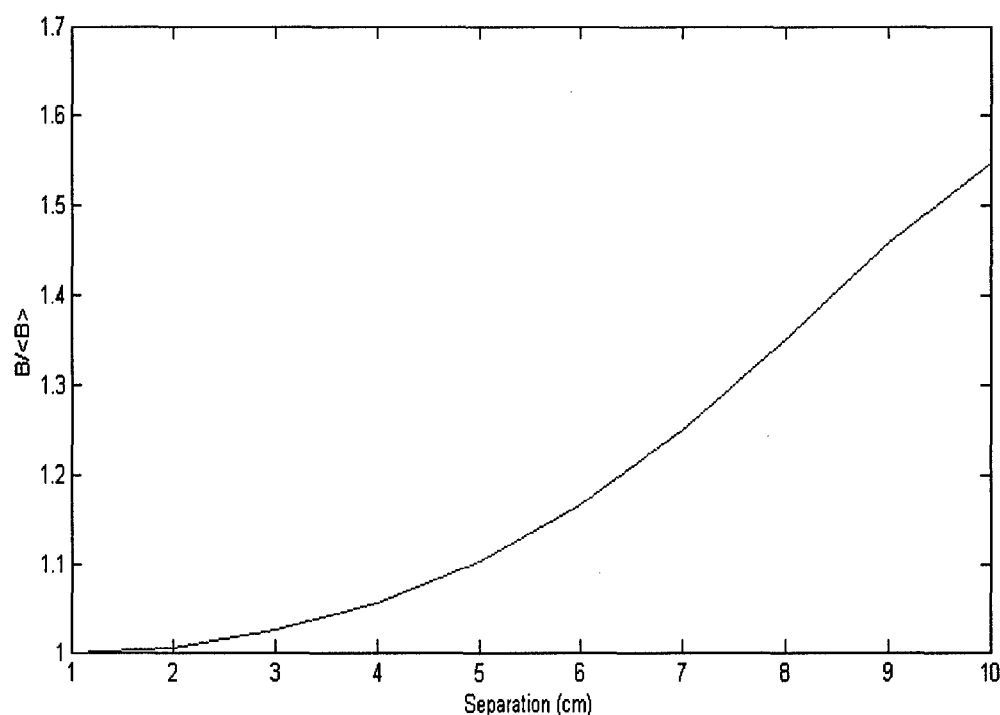


Figure 8.1: The normalized magnetic field changes only 10% when the separation changes from 0 to 5 cm. The actual signal noise may be as high as 10%, making it difficult for the neural net to return accurate separation calculations.

Closely examining the neural net reveals that the net will not output two filaments with a separation of less than 2.5 cm. When the net was given a data set consisting of vectors with identical values for all eight probes (implying a single current filament with no offset) the neural net still returned a separation of 2.5 cm. When the net was given

some artificial test data, with varying degrees of separation, the floor of 2.5 cm was again seen.

Figure 8.2 details this set of artificial data. The “time” axis represents the progression of the artificial data. For every 10 time units, the separation between the two filaments in the artificial data is increased by 2 cm. For every 100 time units, the offset of the centroid of the filaments is increased by 1 cm and the separation is reset to 0 cm. Between incrementing filament separation, the phase is rotated through 180° .

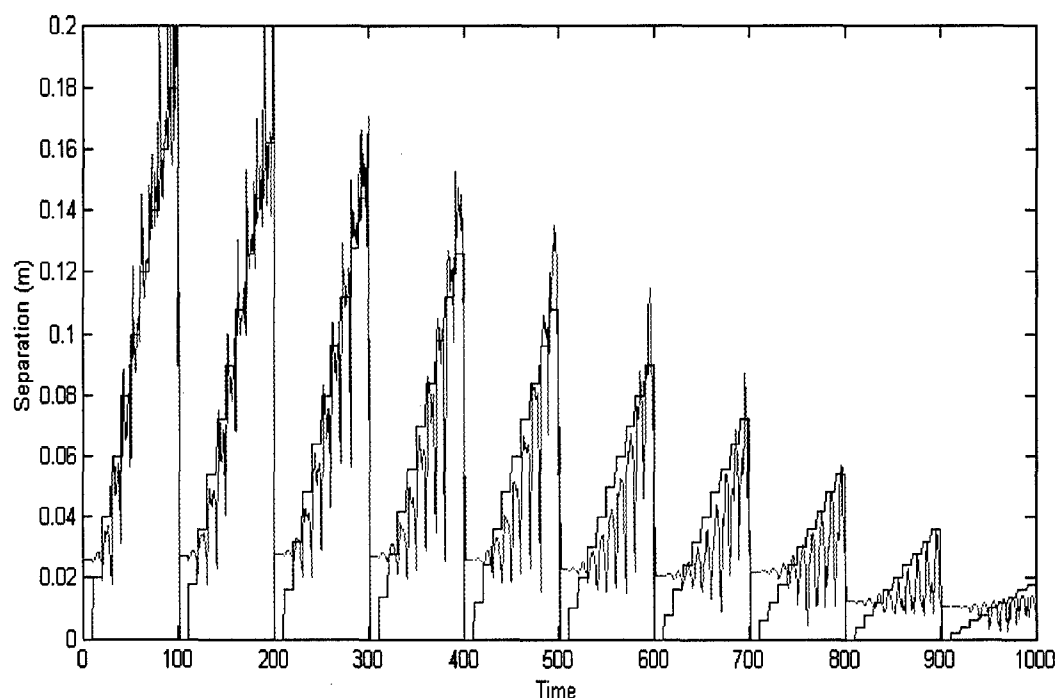


Figure 8.2: Calculated separation on synthetic data. The blue line plots the separation of the two filaments in the synthetic data. The red line plots the calculated separation from the neural net.

The next step in improving the neural net is to create a net that accurately calculates separation of the plasma filaments. Several methods are being tried to accomplish this. They include:

- 1) Adding noise to the input set when training the net.
- 2) Creating a training set that is heavily weighted towards inputs with a small separation of filaments, instead of the uniform distribution that was created before.
- 3) Creating a training set with maximum separations of 5 cm.
- 4) Creating a training set with a normal, random distribution of separations.
- 5) Combinations of the above methods.

The non-linear fitting routine also returns separation values of roughly 5.0 cm as well. This implies that the model of the two filaments recreates the actual magnetic field better when the separation is larger than in reality.

The difficulty now lies in creating a neural net that accurately models separation of the filaments while still minimizing error. Even while attempting to solve this problem, however, we are learning more about the neural net and the details of the physics it is attempting to recreate.

-
- ¹ M.O Hagler and M. Kristiansen, *An Introduction to Controlled Thermonuclear Fusion* (Lexington Books, Lexington, Ma, 1977), p. 32.
- ² U. Shumlak and C. W. Hartman, Phys. Rev. Lett. **75** (18), 3285 (1995).
- ³ U. Shumlak, AIAA 2000-3368 (2000).
- ⁴ R. J. Goldston and P. H. Rutherford, *Introduction to Plasma Physics* (Institute of Physics Publishing, Bristol, 1995), p. 132.
- ⁵ F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*, 2nd Ed. (Plenum Press, New York, 1984), Vol. 1, p. 138.
- ⁶ U. Shumlak and C. W. Hartman, Phys. Rev Lett. **75** (18), 3285 (1995).
- ⁷ I. H. Hutchinson, *Principles of Plasma Diagnostics* (Cambridge University Press, Cambridge, 1987), p. 10.
- ⁸ R. Haberman, *Elementary Applied Partial Differential Equations*, 3rd Ed. (Prentice Hall, Upper Saddle River, NJ, 1998), p. 89.
- ⁹ R. Beale and T. Jackson, *Neural Computing: An Introduction* (Adam Hilger, Bristol, 1969), p. 3.
- ¹⁰ *ibid.* p. 40.
- ¹¹ *ibid.* p. 41.
- ¹² *ibid.* p. 43.
- ¹³ M. Beale and H. Demuth, *Neural Network Toolbox User's Guide* (The MathWorks, Inc., 2000), electronic copy, p. 2-8.
- ¹⁴ *ibid.* p. 5-28.

REFERENCES

M. Beale and H. Demuth, *Neural Network Toolbox User's Guide* (The MathWorks, Inc., 2000), electronic copy.

R. Beale and T. Jackson, *Neural Computing: An Introduction* (Adam Hilger, Bristol, 1969).

F. F. Chen, *Introduction to Plasma Physics and Controlled Fusion*, 2nd Ed. (Plenum Press, New York, 1984), Vol. 1.

R. J. Goldston and P. H. Rutherford, *Introduction to Plasma Physics* (Institute of Physics Publishing, Bristol, 1995).

R. Haberman, *Elementary Applied Partial Differential Equations*, 3rd Ed. (Prentice Hall, Upper Saddle River, NJ, 1998).

M. O. Hagler and M. Kristiansen, *An Introduction to Controlled Thermonuclear Fusion* (Lexington Books, Lexington, Ma, 1977).

I. H. Hutchinson, *Principles of Plasma Diagnostics* (Cambridge University Press, Cambridge, 1987).

U. Shumlak and C. W. Hartman, *Phys. Rev. Lett.* **75** (18), 3285 (1995).

U. Shumlak, AIAA 2000-3368 (2000).

Appendix A: Chord Probe Calibration Code

Common shared_data, b_helm

A = fltarr(3)

shot_start = lon64arr(3)

constant_matrix = fltarr(3,3)

shot_start[0] = 11119043 ;A1

shot_start[1] = 11119011 ;A2

shot_start[2] = 11119026 ;A3

!p.multi = [0,1,3]

for I = 0,2 do begin

 window, I

 ss, shot_start[I]

 signal1 = data('sub_baseline_string("\\DIGITIZERS::CHORD_A1_RAW",11119008)',
 xaxis=time1)

 signal2 = data('sub_baseline_string("\\DIGITIZERS::CHORD_A2_RAW",11119008)',
 xaxis=time2)

 signal3 = data('sub_baseline_string("\\DIGITIZERS::CHORD_A3_RAW",11119008)',
 xaxis=time3)

 b_helm = data('\\DIGITIZERS::B_HELMHOLTZ', xaxis=time)

 ;take data from 0 to 2 ms

 start = min(where(time GE 0))

 finish = min(where(time GE 0.002))

 b_helm = b_helm[start:finish]

 start = min(where(time1 GE 0))

```

finish = min(where(time1 GE 0.002))

signal1 = signal1(start:finish)

elements = n_elements(signal1)

b_helm = interpol(b_helm, elements)

time1 = time1[start:finish]

weights = make_array(elements, value=1.0)

A[1] = 1

A[2] = 1

;direction 1

A[0] = 1

yfit = curvefit(time1, signal1, weights, A, function_name = 'cal_fit')

plot,signal1

oplot,yfit,color=2

constant_matrix[I,0] = A[0]

;direction 2

A[0] = 1

signal2 = signal2[start:finish]

yfit = curvefit(time1, signal2, weights, A, function_name = 'cal_fit')

plot,signal2

oplot,yfit,color=2

constant_matrix[I,1] = A[0]

;direction 3

A[0] = 1

signal3 = signal3[start:finish]

yfit = curvefit(time1, signal3, weights, A, function_name = 'cal_fit')

```

```

    plot,signal3

    oplot,yfit,color=2

    constant_matrix[I,2] = A[0]

;calculations to check accuracy of calculated constants

    b_matrix = make_array(elements,3,value=0.0)

    b_matrix[* ,2] = b_helm

    measured = constant_matrix##b_matrix

    inv_matrix = invert(constant_matrix, status, /double)

    measured_real = flarr(elements,3)

    measured_real[* ,0] = signal1

    measured_real[* ,1] = signal2

    measured_real[* ,2] = signal3

    final_check = inv_matrix##measured_real

endfor

;print constants to file

    openw, 1, 'probe_A.txt'

    printf, 1, 'Probe A'

    printf, 1, inv_matrix

    close, 1

    !p.multi = 0

    window,3

    plot, b_helm

    oplot, final_check[* ,0],co=2

    oplot, final_check[* ,1],co=3

    oplot, final_check[* ,2],co=4

end

```


Appendix B: Mode Simulation Code

```
;pro Justin_mode
```

```
;This program will calculate the ratio of the m_1 magnitude to the m_0 magnitude based on offset
;locations of the current. It also calculates ratios based on m = 2 mode.
;Assumes that magnetix flux is conserved.
```

```
Common Share1, R, phi, B_theta, mag_const, nppts, fract
```

```
;Mu is the Permeability of free space, in N/A^2
```

```
Mu = 4E-7 * !Pi
```

```
;I_Avg is the average current of the plasma filaments, for numerical iteration purposes
```

```
I_Avg = 5000
```

```
;fract is the fraction of total current running in the first filament
```

```
fract = 0.5
```

```
;mag_const is the value of all the "constant" terms in the equation for the magnetic field around a wire.
;the equation is:  $B = \mu * I / (2 * \pi * r)$ . The constant terms are  $\mu * I / (2 * \pi)$ 
```

```
mag_const = Mu*I_avg/(2*!Pi)
```

```
;azimuthal position of probes
```

```
phi = [ 0.0, 45.0, 90.0, 135.0, 180.0, 225.0, 270.0, 315.0 ] *(!Pi/180.0)
```

```
;number of probes is NPPTS
```

```
nppts=n_elements(phi)
```

```
;R is the radius to the outer electrodes, in meters
```

```
R = 0.1
```

```
;inc is the number of increments used for m=1 calculations
```

```
inc = 10
```

```
;rot is the number of azimuthal rotations for the 2 plasma filaments in the m=2 mode
```

```
rot = 4
```

```
;div is the number of divisions used for m=2 calculations
```

```
div = 10
```

;c, alpha and beta are used for r_new calculations

```
c = fltarr(nppts)
```

```
alpha = fltarr(nppts)
```

```
beta = fltarr(nppts)
```

;gamma is the angle between the upper plasma filament and the x-axis

```
gamma = 0.0
```

;r_new is the new radius from the filament to the probe

```
r_new1 = fltarr(inc,div,rot,nppts)
```

```
r_new2 = fltarr(inc,div,rot,nppts)
```

```
beta_s = fltarr(inc,div,rot,nppts)
```

;b_theta is the magnetic field value at each of the probes, in the theta direction

```
b_theta = fltarr(inc,div,rot,nppts)
```

```
;theta_offset = fltarr(nppts,9,rot,delta)
```

;delta_r is the half of the distance between the two plasma filaments (m=2)

```
delta_r = 0.0
```

;r_offset is the distance from the center of the filaments to the center of the test section

```
r_offset = 0.0
```

```
mode_0 = fltarr(inc,div,rot)
```

```
mode_1 = fltarr(inc,div,rot)
```

```
mode_2 = fltarr(inc,div,rot)
```

```
mode_3 = fltarr(inc,div,rot)
```

```
phase2 = fltarr(inc,div,rot)
```

```
a1 = fltarr(inc,div,rot)
```

```
b1 = fltarr(inc,div,rot)
```

```
a2 = fltarr(inc,div,rot)
```

```
b2 = fltarr(inc,div,rot)
```

```
a3 = fltarr(inc,div,rot)
```

```
b3 = fltarr(inc,div,rot)
```

```
;ratio1 is the ration of m_1 over m_0
```

```
ratio1 = fltarr(inc,div,rot)
```

```
;ratio2 is the ration of m_2 over m_0
```

```
ratio2 = fltarr(inc,div,rot)
```

```
;ratio3 is the ratio of m_3 over m_0
```

```
ratio3 = fltarr(inc,div,rot)
```

```
;Y is an array used to store data values in curvefit attempts
```

```
X = fltarr(2,nppts)
```

```
;the law of cosines determines the length of r_new and the distance from the filaments to the probe
```

```
;the first loop moves the R_offset outward by a given increment
```

```
For Row = 0,(inc-1) do begin
```

```
;the next loop will move the two elements of the current apart a distance delta_r from the original center
```

```
    delta_r = 0
```

```
    for I = 0,(div-1) do begin
```

```
        ;the next loop will rotate the filaments through 180 degrees
```

```
        gamma = 0.0
```

```
        for J = 0, (rot-1) do begin
```

```
            c = sqrt(R^2 + r_offset^2 - 2*R*r_offset*cos(phi) )
```

```
            alpha = asin(R*sin(phi)/c)
```

```
            beta = alpha - !pi + gamma
```

```
            r_new1(Row,I,J,*) = sqrt(c^2 + delta_r^2 - 2.0*c*delta_r*cos(beta))
```

```
            r_new2(Row,I,J,*) = sqrt(c^2 + delta_r^2 - 2.0*c*delta_r*cos(!pi-beta))
```

```
            gamma = gamma + (!Pi/rot)
```

```
        endfor
```

```
    delta_r = delta_r + (1.0/div) * (R - r_offset)
```

```

        delta_r = delta_r + (1.0/div) * R

    endfor

    r_offset = r_offset + (1.0/inc) * R

endfor

b_theta = mag_const*(1/r_new1 + 1/r_new2)

;Next are the mode calculations, derived from Fourier series. These only calculate the magnitudes
;m=0 mode is calculated by taking the average of the b_theta values at each sensor
;m=1,2 and 3 are calculated using Fourier series relations

For I = 0,(inc-1) do begin
    For J = 0,(div-1) do begin
        For K = 0,(rot-1) do begin
            ;For L = 0,(nppts-1) do begin

                mode_0(I,J,K) = total(b_theta(I,J,K,*))

                a1(I,J,K) = total(b_theta(I,J,K,*)*sin(phi))

                b1(I,J,K) = total(b_theta(I,J,K,*)*cos(phi))

                a2(I,J,K) = total(b_theta(I,J,K,*)*sin(2.*phi))

                b2(I,J,K) = total(b_theta(I,J,K,*)*cos(2.*phi))

                a3(I,J,K) = total(b_theta(I,J,K,*)*sin(3.*phi))

                b3(I,J,K) = total(b_theta(I,J,K,*)*cos(3.*phi))

            endfor
        endfor
    endfor

    mode_0 = mode_0/nppts

;a1/b1 calculation

```

```
a1 = 2.0 * (a1/nppts)
b1 = 2.0 * (b1/nppts)
mode_1 = sqrt(a1^2 + b1^2)
;a2/b2 calculation
a2 = 2.0 * (a2/nppts)
b2 = 2.0 * (b2/nppts)
mode_2 = sqrt(a2^2 + b2^2)
phase2 = atan(a2,b2) * 180/!Pi
;a3/b3 calculation
a3 = 2.0 * (a3/nppts)
b3 = 2.0 * (b3/nppts)
mode_3 = sqrt(a3^2 + b3^2)
;ratio calculations
ratio1 = mode_1/mode_0
ratio2 = mode_2/mode_0
ratio3 = mode_3/mode_0
```

Appendix C: Non-Linear Fit Routine Code

```
;pro auto_cut_mode
```

```
;This program will determine the radial position of the plasma currents, as well as their angle from the center.
```

```
Common Share1, R, phi, B_theta, mag_const, nppts, fract
```

```
;Mu is the Permeability of free space, in N/A^2
```

```
Mu = 4E-7 * !Pi
```

```
;I_Avg is the average current of the plasma filaments, for numerical iteration purposes
```

```
I_Avg = 5000
```

```
;fract is the fraction of total current running in the first filament
```

```
fract = 0.5
```

```
;mag_const is the value of all the "constant" terms in the equation for the magnetic field around a wire.  
;the equation is:  $B = \frac{\mu * I}{2 * \pi * r}$ . The constant terms are  $\frac{\mu * I}{2 * \pi}$ 
```

```
mag_const = Mu*I_avg/(2*!Pi)
```

```
;azimuthal position of probes
```

```
phi = [ 0.0, 45.0, 90.0, 135.0, 180.0, 225.0, 270.0, 315.0 ] *(!Pi/180.0)
```

```
;number of probes is NPPTS
```

```
nppts=n_elements(phi)
```

```
;R is the radius to the outer electrodes, in meters
```

```
R = 0.1
```

```
;The next section will read actual data and use a curvefit to determine the location of the plasma  
;It uses an approximation of the geometry at each iteration to
```

```
ss, 10515009
```

```
elements = N_elements(data(\b_p0_0'))
```

```
;B_data holds the test data for each magnetic sensor, taken from the actual probes
```

```
B_data = fltarr(nppts, elements)
```

```

B_data(0,*) = data(\b_p0_0',xaxis=time)
B_data(1,*) = data(\b_p0_45',xaxis=time)
B_data(2,*) = data(\b_p0_90',xaxis=time)
B_data(3,*) = data(\b_p0_135',xaxis=time)
B_data(4,*) = data(\b_p0_180',xaxis=time)
B_data(5,*) = data(\b_p0_225',xaxis=time)
B_data(6,*) = data(\b_p0_270',xaxis=time)
B_data(7,*) = data(\b_p0_315',xaxis=time)

```

;check is an array that registers whether the data is usable.

;finish is the final time value to be calculated

```
finish = 7850 ;150 usec
```

```
start = min(where(B_data(0,*) GE 0.03 and B_data(1,*) GE 0.03 and B_data(2,*)GE$ 0.03 and
B_data(3,*) GE 0.03 and B_data(4,*) GE 0.03 and B_data(5,*) GE 0.03 and B_data(6,*) GE 0.03 $
and B_data(7,*) GE 0.03))
```

```
check = where (B_data(0,*) GE 0 and B_data(1,*) GE 0 and B_data(2,*) GE 0 and$ B_data(3,*) GE 0 $
and B_data(4,*) GE 0 and B_data(5,*) GE 0 and B_data(6,*) GE 0 and$ B_data(7,*) GE 0)
```

```
gate = where(check GE start and check LE finish)
```

```
check = check(gate)
```

```
n_check = N_elements(check)
```

;create vectors to hold data based on size limitations: elements

;Radius is the array that holds radial positions determined by yfit3 (althernative geometry)

```
Radius1 = fltarr(elements)
```

```
Radius2 = fltarr(elements)
```

;Angle is the array that holds angle values determined by yfit3

```
Angle1 = fltarr(elements)
```

```
Angle2 = fltarr(elements)
```

```
;Error1 corresponds to yfit1, Error2: yfit2, etc.
```

```
Error1 = fltarr(elements)
```

```
Error3 = fltarr(elements)
```

```
actual_offset1 = fltarr(elements)
```

```
iterations = fltarr(elements)
```

```
;loop through time, and at each time point use curvefit to determine position of the plasma
;the initial conditions (matrix "A") will be a guess the first time through, and then will be set to the values
of the
;last data point
```

```
weights = [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]
```

```
Guess1 = 0.001
```

```
Guess2 = 0.02
```

```
Guess3 = 0.1
```

```
Guess4 = 0.05
```

```
A1 = [Guess1, Guess4]
```

```
B = [0.01, 0.01, 0.1, -0.1]
```

```
X = phi
```

```
;proceed through 150 usec
```

```
for I = 0,n_check-1 do begin
```

```
    ;normalize the B(theta) values by the average field
```

```
    Y = B_data[*,check(I)]/(total(B_data[*,check(I)))/8)
```

```
    ;yfit1 is the approximation using no m=2 analysis
```

```
    yfit1 = CURVEFIT(X,Y, weights, A1, ITMAX = 1000, TOL = 1e-11,$ FUNCTION_NAME =$
    'non_linear1'/NODERIVATIVE, CHISQ = chi1)
```

```
    B[0] = A1[0]
```

```
    B[1] = A1[0]
```



```

;yfit3 is an alternative approach to solving the geometry

yfit3 = CURVEFIT(X, Y, weights, B, ITER = iter, ITMAX = 1000, TOL = $
1e-5,FUNCTION_NAME = 'comparison_geometry',/NODERIVATIVE, CHISQ = chi3)

;store data from yfit1

actual_offset1[check(I)] = A1[0]

Error1[check(I)] = chi1

iterations[check(I)] = iter

;store data from yfit3

Radius1[check(I)] = B[0]

Radius2[check(I)] = B[1]

Angle1[check(I)] = B[2]

Angle2[check(I)] = B[3]

Error3[check(I)] = chi3

endfor

;Calculate the bisector for the yfit3 data, which will give an m=1 approximation

x1 = radius1*cos(angle1)

x2 = radius2*cos(angle2)

y1 = radius1*sin(angle1)

y2 = radius2*sin(angle2)

x3 = (x1 + x2)/2

y3 = (y1 + y2)/2

m = (sqrt(x3^2 + y3^2)) < 1.0

q = atan(y3,x3)

separation = sqrt((x1 - x2)^2 + (y1 - y2)^2)

;change all radius values to positive, add 180 to associated angles

a = where(radius1 LT 0.0)

```

```
angle1(a) = angle1(a) + !Pi
```

```
radius1(a) = -1*radius1(a)
```

```
a = where(radius2 LT 0.0)
```

```
angle2(a) = angle2(a) + !Pi
```

```
radius2(a) = -1*radius2(a)
```

```
;for use with error_calcs.pro
```

```
target2 = fltarr(4,n_elements(x1))
```

```
target2(0,*) = x1
```

```
target2(1,*) = y1
```

```
target2(2,*) = x2
```

```
target2(3,*) = y2
```

```
end
```

```
pro non_linear1, X, A, F, pder
```

```
nppts = 8
```

```
Mu = 4E-7 * !Pi
```

```
I_Avg = 5000
```

```
R = 0.1
```

```
mag_const = Mu*I_avg/(2*!Pi)
```

```
;A[0] = r_offset, distance from origin to center of plasma filaments
```

```
;A[1] = theta, angle between x axis and a the center of the plasma
```

```
c1 = sqrt(R^2 + A[0]^2 - 2*R*A[0]*cos(X-A[1]))
```

```
Sum1 = total(1/c1)
```

```
F = nppts * (1/c1) * (1/Sum1)
```

```
end
```

```
pro comparison_geometry, X, A, F, pder

    ;A[0] is Radius1
    ;A[1] is Radius2
    ;A[2] is Angle1
    ;A[3] is Angle2

    nppts = 8

    R = 0.1

    Beta1 = X - A[2]

    Beta2 = A[3] - X

    r_new1 = sqrt(R^2 + A[0]^2 - 2*R*A[0]*cos(Beta1))

    r_new2 = sqrt(R^2 + A[1]^2 - 2*R*A[1]*cos(Beta2))

    Sum1 = 0.5*total(1/r_new1) + 0.5*total(1/r_new2)

    F = nppts * (0.5/r_new1 + 0.5/r_new2) * (1/Sum1)

end
```

Appendix D: Neural Net Code

The weight and bias matrices are stored in the ZaP data base. They are copied directly from MATLAB.

```
;This program recreates the neural net that was trained using MATLAB software. It is a ;3 layer Feed-
Forward Backpropagation network,
;with 25 neurons in the first two layers and 4 in the last. The first two layers are tansig ;transfer functions,
the last is a purelin.
;The values for the weights and biases were taken from the MATLAB network object, ;which was trained
using trainLM. The training set was created
;using the target_generation.m file, which is identical to the target_generation3.pro file
```

```
;read data
```

```
ss,20214013
```

```
elements = N_elements(data(\b_p0_180'))
```

```
B = dblarr(elements,8)
```

```
Bn = dblarr(elements,8)
```

```
B(*,0) = data(\b_p0_0',xaxis=time)
```

```
B(*,1) = data(\b_p0_45',xaxis=time)
```

```
B(*,2) = data(\b_p0_90',xaxis=time)
```

```
B(*,3) = data(\b_p0_135',xaxis=time)
```

```
B(*,4) = data(\b_p0_180',xaxis=time)
```

```
B(*,5) = data(\b_p0_225',xaxis=time)
```

```
B(*,6) = data(\b_p0_270',xaxis=time)
```

```
B(*,7) = data(\b_p0_315',xaxis=time)
```

```
;normalize data
```

```
For N = 0,elements-1 do begin
```

```
    Bn[N,*] = B[N,*]/(total(B[N,*])/8.0)
```

endfor

;create weight and bias matrices as defined by MATLAB Neural net toolbox

W1 = dblarr(8,25)

W2 = dblarr(25,25)

W3 = dblarr(25,4)

B1 = dblarr(1,25)

B2 = dblarr(1,25)

B3 = dblarr(1,4)

W1 = mdsvalue(\neural_net:W1')

W2 = mdsvalue(\neural_net:W2')

W3 = mdsvalue(\neural_net:W3')

B1 = mdsvalue(\neural_net:B1')

B2 = mdsvalue(\neural_net:B2')

B3 = mdsvalue(\neural_net:B3')

;The actual calculations: multiply by the weight matrix, add the bias, and then run through the transfer function

;The transfer function for the first two sets is a tansig, the last one is simply a linear transfer function, n=n.

A1 = W1##Bn

for I = 0,elements-1 do begin

A1(I,*) = A1(I,*) + B1

endfor

A1 = 2/(1+exp(-2*A1))-1

A2 = W2##A1

for I = 0,elements-1 do begin

```

        A2(I,*) = A2(I,*) + B2

    endfor

    A2 = 2/(1+exp(-2*A2))-1

    target = W3##A2

    for I = 0, elements-1 do begin

        target(I,*) = target(I,*) + B3

    endfor

;calculations

    m1_approx = dblarr(elements,2)

    m1_approx(*,0) = (target(*,0) + target(*,2))/2

    m1_approx(*,1) = (target(*,1) + target(*,3))/2

    m1_approx_phase = atan(m1_approx(*,1),m1_approx(*,0))

    m1_approx = sqrt(m1_approx(*,0)^2 + m1_approx(*,1)^2)

    m1 = data(\m_1_p0)

    m0 = data(\m_0_p0)

    m1_real = m1/m0*.1

end

```