UCRL-LR-136220
Thesis

Experimental Measurement of the 4-D Transverse Phase Space Map of a Heavy Ion Beam

H.S. Hopkins
(Ph.D.)

December 1997

U.S. Department of Energy
Lawrence
Livermore
National
Laboratory

Approved for public release; further dissemination unlimited

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This work was performed under the auspices of the U. S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract No. W-7405-Eng-48.

This report has been reproduced directly from the best available copy.

Available electronically at http://www.doe.gov/bridge

Available for a processing fee to U.S. Department of Energy
and its contractors in paper from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-mail: reports@adonis.osti.gov

Available for the sale to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

OR

Lawrence Livermore National Laboratory
Technical Information Department's Digital Library
http://www.llnl.gov/tid/Library.html

Experimental Measurement of the 4-D Transverse Phase Space Map of a Heavy Ion Beam

by

Harvey S. Hopkins

B.S. (United States Naval Academy) 1981
M.S. (University of California, Davis) 1987

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering-Nuclear Engineering

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Edward C. Morse, Chair
Professor David L. Judd
Professor Jasmina L. Vujic

Fall 1997

The dissertation of Harvey S. Hopkins is approved:

2 Dec 1997

Chair Date

10 Dec. 1997

Date

Jasmina Vujić 8. December 1997

Date

University of California at Berkeley

1997

Abstract

Experimental Measurement of the 4-D Transverse Phase Space Map of a Heavy Ion Beam

by

Harvey S. Hopkins

Doctor of Philosophy in Engineering-Nuclear Engineering

University of California at Berkeley

Professor Edward C. Morse, Chair

The development and employment of a new diagnostic instrument for characterizing intense, heavy ion beams is reported on. This instrument, the "Gated Beam Imager" or "GBI" was designed for use on Lawrence Livermore National Laboratory Heavy Ion Fusion Project's "Small Recirculator", an integrated, scaled physics experiment and engineering development project for studying the transport and control of intense heavy ion beams as inertial fusion drivers in the production of electric power. The GBI allows rapid measurement and calculation of a heavy ion beam's characteristics to include all the first and second moments of the transverse phase space distribution, transverse emittance, envelope parameters and beam centroid. The GBI, with appropriate gating produces a time history of the beam resulting in a 4-D phase-space and time "map" of the beam. A unique capability of the GBI over existing diagnostic instruments is its ability to measure the "cross" moments between the two transverse orthogonal directions. Non-zero "cross" moments in the alternating gradient lattice of the Small Recirculator are indicative of focusing element rotational misalignments

contributing to beam emittance growth. This emittance growth, while having the same effect on the ability to focus a beam as emittance growth caused by non-linear effects, is in principle removable by an appropriate number of focusing elements.

The instrument uses the pepperpot method of introducing a plate with many pinholes into the beam and observing the images of the resulting beamlets as they interact with a detector after an appropriate drift distance. In order to produce adequate optical signal and repeatability, the detector was chosen to be a microchannel plate (MCP) with a phosphor readout screen. The heavy ions in the pepperpot beamlets are stopped in the MCP's thin front metal anode and the resulting secondary electron signal is amplified and proximity-focused onto the phosphor while maintaining the spatial and intensity characteristics of the heavy ion beamlets. The MCP used in this manner is a sensitive, accurate, and long-lasting detector, resistant against signal degradation experienced by previous methods of intense heavy ion beam detection and imaging. The performance of the GBI was benchmarked against existing mechanical emittance diagnostics and the results of sophisticated beam transport numerical simulation codes to demonstrate its usefulness as a diagnostic tool.

A method of beam correction to remove the effects of quadrupole focusing element rotational misalignments is proposed using data obtainable from a GBI. An optimizing code was written to determine the parameters of the correction system elements based on input from the GBI. The results of this code for the Small Recirculator beam are reported on.

Edward Morse

Professor Edward C. Morse
Dissertation Committee Chair

Contents

List of Figures vi

List of Tables ix

1 Introduction 1

2 Small Recirculator Experiment Overview 5
- 2.1 Small Recirculator Description . . . 7
  - 2.1.1 Full Ring Operation . . . 7
  - 2.1.2 Partial Ring Operation . . . 8
  - 2.1.3 Potassium Ion Source . . . 10
  - 2.1.4 Electrostatic Matching Section . . . 12
  - 2.1.5 Straight Magnetic Transport Section . . . 14
  - 2.1.6 Electrostatic Bending Section . . . 15
  - 2.1.7 Multi-Instrument Diagnostic Tank . . . 16
- 2.2 Beam Physics Scaling of the Small Recirculator . . . 18
  - 2.2.1 Physical Size Scaling . . . 18
  - 2.2.2 Maintaining Key Dimensionless Beam Physics Parameters . . 19

3 Heavy Ion Beam Transport 24
- 3.1 Origin of Beam Emittance . . . 25
- 3.2 Small Recirculator Beam Transport . . . 26
- 3.3 Moment Description of the Ion Beam . . . 31
- 3.4 Definition of Emittance . . . 34

4 Pepperpot Method and Image Data Analysis 37
- 4.1 Capabilities of Existing Emittance Diagnostics . . . 37
- 4.2 The Pepperpot Method . . . 38
- 4.3 Design of the Gated Beam Imager . . . 40
- 4.4 Pepperpot Method Data Analysis . . . 42
- 4.5 Image Capture and Image Analysis Code . . . 50
- 4.6 Sample Output from the Data Analysis Code . . . 52

5 Detector Selection and Equipment Description 67

5.1 Fast Plastic Scintillator as a Heavy Ion Detector 67
5.2 Microchannel Plate Detector as a Heavy Ion Detector 71
5.3 Obtaining Time-Resolved Beam Data by Gating the MCP 76
5.4 Camera System Description 79
5.5 Operating the Gated Beam Imager 81
5.6 Applying Corrections to the GBI Images 83
5.6.1 CCD Dark Charge Correction 83
5.6.2 CCD Flat Field Response Correction 84
5.6.3 MCP Detector Flat Field Response Correction 85
5.6.4 Preparing the Images for Data Analysis 88

6 A Beam Correction System **91**
6.1 The Effect on the Beam of Rotationally Misaligned Quadrupoles . . . 91
6.2 Approximate Correction for a Single Misaligned Quadrupole 95
6.3 Correction for Random Quadrupole Rotation Misalignments 99
6.3.1 Applying GBI Data to a Beam Correction System 101
6.3.2 Determining the Optimum Correction System Parameters . . 104
6.3.3 Implementing the Optimizing Algorithm 106

7 Experimental Results **113**
7.1 GBI Setup and Calibration 115
7.2 Observing the Effect of Axial Beam Expansion 115
7.3 Data Analysis and Results at the 45° Bend Section 119
7.3.1 Comparison of GBI and Slit Scanner Measurements 120
7.3.2 Error Analysis 122
7.3.3 Comparison of GBI Data with WARP Numerical Simulation Code Results 124
7.4 Observing and Measuring the Effect of Magnetic Quadrupole Rotation Errors and Applying Approximate Corrections 126
7.5 Beam Transverse Intensity Distribution Measurement 138

8 Summary and Conclusions **141**

A GBI Design Optimizing Spreadsheet **146**

B Derivation of Beam Moment Evolution Equations **151**

C Explicit Solutions for Δx^2 and Δy^2 **156**

D GBI Equipment Settings for Acquiring Gated Images **162**

E Simulation and Data Analysis Codes **165**
E.1 "FULLAG" 165

E.2 "REDUCEDAG" 170
E.3 "CORRECTOR" 177
E.4 "PEPPERPOT" 185

List of Figures

1.1 IFE Power Plant Conceptual View 2

2.1 Small Recirculator Plan View 6
2.2 Small Recirculator Schematic 7
2.3 Experimental Configuration of the Small Recirculator (45°) 8
2.4 Experimental Configuration of the Small Recirculator (90°) 9
2.5 Ion Source Assembly 11
2.6 Ion Source Pulser Power Supply Circuit 12
2.7 Electrostatic Matching Section 13
2.8 Small Recirculator Straight Magnetic Transport Section 15
2.9 Small Recirculator 45° Bend Section 16
2.10 Multi-Instrument Diagnostic Chamber 17
2.11 "C-design" Heavy Ion Driver Schematic 19

3.1 Beam 4-D Phase Space Volume 27
3.2 2-D Phase Space Ellipse Coefficients 29
3.3 Beam Sigma Matrix 33

4.1 Nonlinear Evolution of Phase Space 39
4.2 Pepperpot Method Schematic 39
4.3 Simplified GBI Instrument Schematic 40
4.4 Phase Space Ellipse Rotation Angle Determination 42
4.5 Beamlet Data Required for Moment Calculations 43
4.6 Beam Phase Space Ellipse 44
4.7 GBI Phase Space Plot 45
4.8 GBI Beamlet Contour 48
4.9 Typical GBI Data Image 52
4.10 Identification of Beamlet Holes 53
4.11 Data Analysis Windows 54
4.12 Relationship Between Holes and Spots 55
4.13 Contour Plot of the Beamlet Intensity 56
4.14 Beamlet Intensity Surface Plot 57
4.15 Beamlet Intensity Surface Plot 58
4.16 Smoothed Beamlet Surface Plot 58

4.17 Beamlet Cross-section Plots 59
4.18 Beamlet Profile Plot 60
4.19 2-D Phase Space Plots 61
4.20 Pepperpot Rotation Angle Plot 62
4.21 Incorrect Rotation Angle 63
4.22 Cross Moment Plot($<x' - y'>$, $<x - y>$) 64
4.23 Cross Moment Plot $<x' - y>$, $<x - y'>$ 65
4.24 Beam Statistics Plots 66

5.1 BC-412 Scintillator Light Output vs. Exposure 69
5.2 Scintillator Exposure Image Sequence 71
5.3 Microchannel Plate Schematic 73
5.4 Microchannel Plate Resolution 75
5.5 PINEX MCP Gate Circuit 77
5.6 MCP Detector Gate Signals 78
5.7 GBI Equipment Interconnections 79
5.8 GBI CCD Camera In Position 80
5.9 GBI Camera Control Equipment 82
5.10 GBI Image Capture Timing Diagram 83
5.11 Dark Charge Image Line Average Plot 84
5.12 CCD Camera Flat Field Equipment Setup 86
5.13 MCP X-ray Flat Field Equipment Setup 87
5.14 MCP X-ray Flat Field, Field-of-View 88
5.15 MCP Non-Uniform Response 89

6.1 Beam and Laboratory Coordinate System 94
6.2 A Single Misaligned Quadrupole, Part A 96
6.3 A Single Misaligned Quadrupole, Part B 97
6.4 Correction of a Known Quadrupole Rotation Error, Part A 98
6.5 Correction of a Known Quadrupole Rotation Error, Part B 99
6.6 Conceptual Beam Correction System 102
6.7 Verification of Simplified Δx^2 and Δy^2 solutions 104
6.8 Beam Correction System Optimization Code Results Plots 109
6.9 Correction of Random Quadrupole Rotation Errors, Part A 111
6.10 Correction of Random Quadrupole Rotation Errors, Part B 112

7.1 Bending Dipole Voltage Scan 114
7.2 View of the GBI 116
7.3 Camera Field of View Calibration Image 117
7.4 Beam Axial Expansion Images 118
7.5 Source Pulser and Faraday Cup Waveforms 119
7.6 Normalized Emittance vs. Time at the 45° Bend 120
7.7 Beam Ion Distribution Second Moments vs. Time 121

7.8 Experimental Emittance Data Compared to "WARPxy" Simulation . 125
7.9 Experimental Envelope Data Compared with "WARPxy" Simulation 126
7.10 GBI Images with Quadrupole Rotation Errors 128
7.11 Phase Space Emittance Plots with "Aligned" Quadrupoles 129
7.12 Cross Moment Plots with "Aligned" Quadrupoles 130
7.13 Phase Space Emittance Plots with Quadrupole 8 rotated 5° counter-clockwise 131
7.14 Cross Moment Plots with Quadrupole 8 rotated 5° counter-clockwise 132
7.15 Phase Space Emittance Plots with Quarupole 13 rotated 5° clockwise 133
7.16 Cross Moment Plot with Quarupole 13 rotated 5° clockwise 134
7.17 Phase Space Emittance Plots with Quadrupoles 8 and 13 counter-rotated 135
7.18 Cross Moment Plot with Quadrupoles 8 and 13 counter-rotated . . . 136
7.19 Beam Intensity Surface Plot from the GBI 139
7.20 Beam Intensity Surface Plot from Cross Slits 139
7.21 GBI Image Pixel Row Average Plot 140

B.1 Beam and laboratory coordinate system 152

List of Tables

2.1 Nominal Electrostatic Quadrupole Voltages for an 80 keV Beam . . . 14
2.2 Small Recirculator Design Parameters 22
2.3 Comparison Between the Small Recirculator and C-design Driver . . 23

4.1 GBI Design Spreadsheet Results 41

5.1 BC-412 Scintillator Properties 68
5.2 Galileo Microchannel Plate Characteristics 73
5.3 TRIM Code Calculation Results 74

6.1 SPSA Optimization Results 110

7.1 Beam Ion Distribution Second Moments at the 45° bend 122
7.2 Pepperpot Data Analysis Sensitivity to Errors 123
7.3 Summary of Beam Parameters from Rotated Quadrupole Images . . . 137

D.1 GBI Equipment and Software Settings for GBI Measurements at 45° . 162
D.2 GBI Measurement Parameters at 45° 163
D.3 45° Integrated image file information 163
D.4 45° 100 ns gated image file information 164

Acknowledgements

I'd most like to thank my wife, Pam, for her steadfast support throughout my graduate education. Without her, I couldn't have done this work.

An overwhelming number of people provided assistance and information in support of this dissertation. Professor Ed Morse provided constructive criticism and direction when I needed it. Dr. Larry Nattrass of LLNL provided me the opportunity to work on LLNL's Heavy Ion Fusion Project. Dr. Alex Friedman provided financial support making it possible for me to work on the project and was always available with helpful suggestions. Through his efforts the Small Recirculator project was held together in a tough funding environment. Dr. Mike Cable suggested the Gated Beam Imager concept to me and Dr. Craig Sangster and Bo Nelson provided me with helpful suggestions with experimental methods. Dr. John Barnard and Dr. Steve Lund were always helpful in computational and theoretical aspects of heavy ion beam transport. I am indebted to Dr. Arthur Paul for his allowing me to use his Pepperpot data analysis code, instructing me in its use, and assisting me in modifying it.

There are many other people who have helped me and I thank them all. From the Heavy Ion Fusion Project, Dave Grote, George Craig, Larry Ahle, Gene Lauer, Bill Manning, Darryl Autrey, Art Molvik, Melissa Cisneroz, and Fred Dietrich. From Bechtel Nevada Corporation, Charlie Diamond, Ed Perreira, Dave Longinotti, Arte Debeling, Bill Fritz, and Chuck Ward. From LLNL Lasers Directorate, Dave Pendleton, Debbie Callaghan, Dean Lee, Roger Griffith, and Perry Bell. From Allied-Signal Corporation., Geoff Mant, and Rob Costa. From LBNL, Schmuel Eylon.

Chapter 1

Introduction

The international search for a viable technology to employ in electric power generation from controlled fusion includes the United States' Inertial Confinement Fusion (ICF) Program. Large glass lasers including NOVA and the National Ignition Facility (NIF) at the Lawrence Livermore National Laboratory (LLNL) are being operated or built as research tools for achieving ignition and gain in a fusion target.[1] Success in achieving ignition is a major hurdle to a practical, economical, and environmentally attractive source of electric power. The National Ignition Facility, using lasers as a driver, will demonstrate ignition and the response of target chamber materials to the fusion environment, as well as contributing to national security as part of the U.S. Department of Energy's Stockpile Stewardship Program.

A nominal 1000 MWe inertial fusion energy (IFE) power plant requires a high repetition rate and high driver efficiency for economic power production.[2] In the U.S., several driver options are being studied to obtain the desired characteristics of high pulse rate, efficiency, reliability, and low cost. These options include the diode-pumped solid state laser, light ion accelerator, and a heavy ion accelerator based on induction accelerator technology.[3] The heavy ion driver research is centered at LLNL and Lawrence Berkeley National Laboratory. The major issues being studied at these sites, using theoretical and computational methods, and experiments, include ion source design and performance, beam transport, beam acceleration and control, final

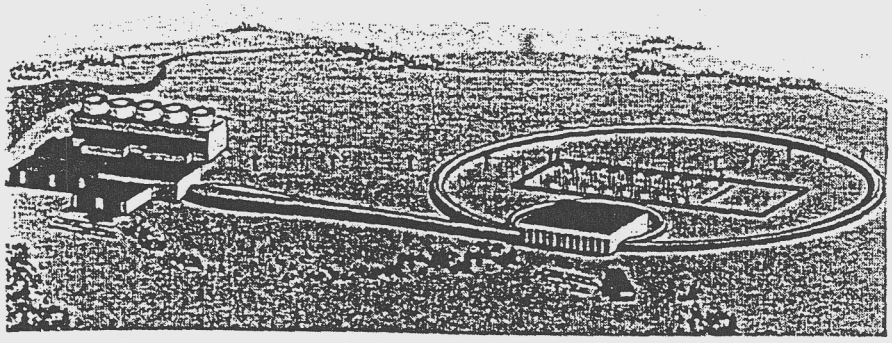


Figure 1.1: Conceptual view of an inertial fusion energy power plant.[1]

focusing element design, target chamber beam transport, and fusion target design and performance. Additionally integrated power plant systems studies done to model the plant and driver design, construction, and operation are being carried out to estimate the per kilowatt-hour cost of electricity from an inertial fusion energy plant.

Existing accelerators for high-energy physics research have had long lifetimes and high availability, and it is expected that IFE drivers could share these attributes. An induction accelerator in particular is efficient, can readily transport and amplify a high-current beam, and can easily meet the pulse rate requirement of 3-10 Hz needed for energy production.[4] Conceptual designs for IFE power plants have been studied such as the artist's conception in Figure 1.1 and the HYLIFE II reactor chamber design. The HYLIFE II design addressed issues of how to convert the output of the target to a working fluid as well as alleviating material damage from intense neutron irradiation and other critical engineering issues.[5]

The requirements for the driver are imposed by the fusion target design and to a lesser extent the geometry of the target chamber.[5–7] Current work at LLNL has focused on indirect drive targets in which a holraum with radiation converters encloses a deuterium-tritium pellet. The converters are illuminated by clusters of ion beams with a narrow cone angle. To achieve a gain of about 100 such a target is required to

receive about 5 MJ in incident ion beam energy in a suitably shaped pulse of about 10 ns in duration.[8] For an ion mass of 100-200 amu the required energy per ion is of order 5 GeV based on radiation converter stopping range of R=.02-0.2 g/cm^2.[2] The use of heavy ions and their high individual energies allows a relatively low beam current (compared to light ion beams) and non-neutralized ballistic focusing of the beam across a 4-10 meter radius target chamber. The ability to focus to a nominal 3 millimeter spot is governed by four major factors: beam space charge, beam emittance, transverse focusing field misalignments, and transverse focusing field aberrations. Neglecting the effects of the other three factors an estimate of the maximum permissible beam emittance at the final focus lens can be made. Assuming 6 meters from the final focusing lens to the target and a 3 millimeter spot size on target, ions can be aimed with maximum convergence angle of 0.3 milliradians. This angle represents the ratio between ion transverse energy and beam directed energy. For a 5 GeV beam, the transverse energy must remain less than 0.5 keV. The transverse emittance is the product of beam transverse dimension and velocity spread normalized to the beam's directed velocity. Uncorrected optics are limited to a focusing angle of about 15 milliradians and so for a 3 millimeter spot size, one obtains an emittance limit of 45 mm-mr.

Limiting the emittance growth from source to target of these heavy ion beams is a leading topic in the research and development of a heavy ion driver. Understanding the processes that cause emittance growth and developing the engineering to produce the necessary beam attributes in an economical way are of high priority. Theoretical, computational, and experimental work are in progress at LLNL with assistance from LBNL in the building and operating of the Small Recirculator, an integrated, scaled driver physics experiment and engineering technology development platform.[9] Characterizing its beam of singly-charged potassium ions is crucial to verifying its design operating specifications and benchmarking sophisticated numerical beam simulation codes that will help design a full-scale power plant driver. For this purpose an improved emittance diagnostic, the Gated Beam Imager (GBI) was developed and fielded for this thesis to obtain information on the Small Recirculator beam not

previously available.

A description of the Small Recirculator experiment and its technology is presented in Chapter 2. Chapter 3 presents a review of intense ion beam transport issues relevant to the design and operation of the GBI. Chapter 4 explains the design methodology used for the GBI and data analysis algorithm. Chapter 5 presents the work conducted to select the ion detector used in the GBI, a description of the instrument, and its operation. As an application of the unique information available from the GBI, a beam correction system concept is proposed in Chapter 6. This correction system uses inputs from the GBI to remove the effects of random quadrupole focusing magnet rotation errors from a heavy ion driver beam. Chapter 7 presents the results of GBI measurements on the Small Recirculator beam using the prototype instrument developed for this thesis and validation by comparison with existing diagnostic instrumentation and code results. Also, experimental results relevant to the beam correction system are presented.

Chapter 2

Small Recirculator Experiment Overview

As achieving ignition and conducting an experimental program on the National Ignition Facility can address among many issues, fusion target physics, chamber materials issues, and give the necessary public awareness boost to inertial fusion energy research, LLNL is building a heavy ion recirculating induction accelerator (dubbed "Small Recirculator") to study in a scaled manner the physics and engineering of producing, accelerating, transporting, and controlling an intense, space-charge-dominated heavy ion beam through multiple laps of a circular accelerator.[9] The recirculating induction accelerator concept applied to a full-scale driver promises component construction cost reduction over a linear accelerator by re-using the same accelerating and focusing elements multiple times in a beam's travel from source to target. The long path length available in a multi-lap recirculator lessens the required accelerating gradient through each induction cell allowing smaller (and presumably cheaper) components than those of a shorter path length "once-through" linear accelerator.[10]

The Small Recirculator will be the first accelerator of its kind, and with multiple lap operation will explore the physics and engineering issues of beam control needed to economically build a driver-size machine. The repeated use of accelerator elements during beam acceleration and compression on multiple laps also requires new engineering development in areas of beam control, acceleration, and diagnostics.[12] These issues are being addressed in the Small Recirculator project with the goal of an in-

tegrated, scaled system to provide experience in all aspects that apply to building and operating a driver and to provide data to benchmark computer codes to aid in the design of a power plant size induction accelerator. To date, prior to multiple lap operation, the experiment has yielded valuable data on emittance growth, and beam control of intense heavy ion beams around bends. A summary of its design is in Table 2.2 at the end of this chapter on page 22 and drawings of its completed form are shown in Figures 2.1 and 2.2.

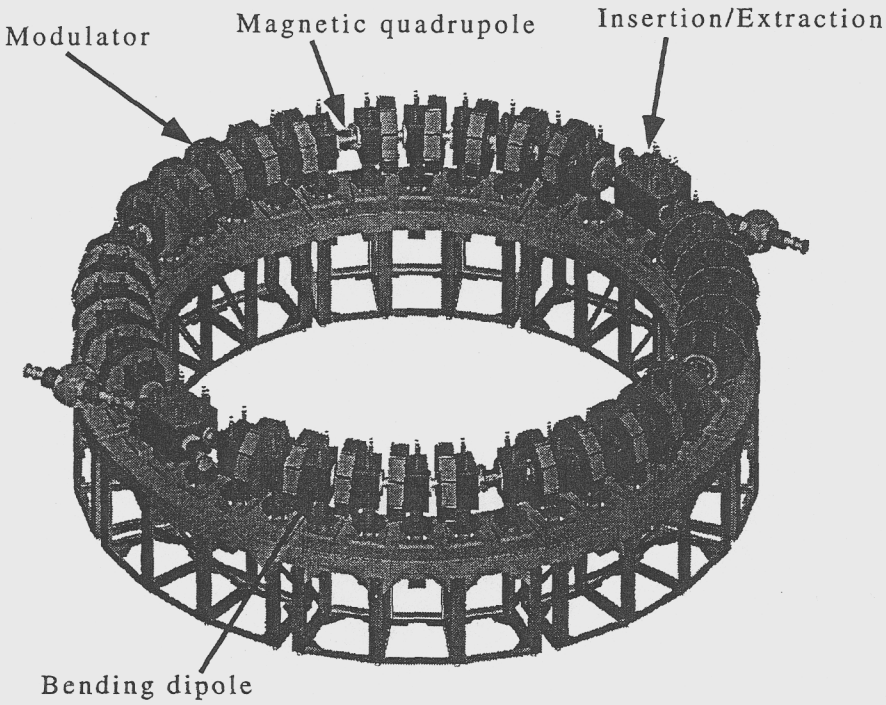


Figure 2.1: Final configuration of the Small Recirculator ring (source and matching section not pictured).

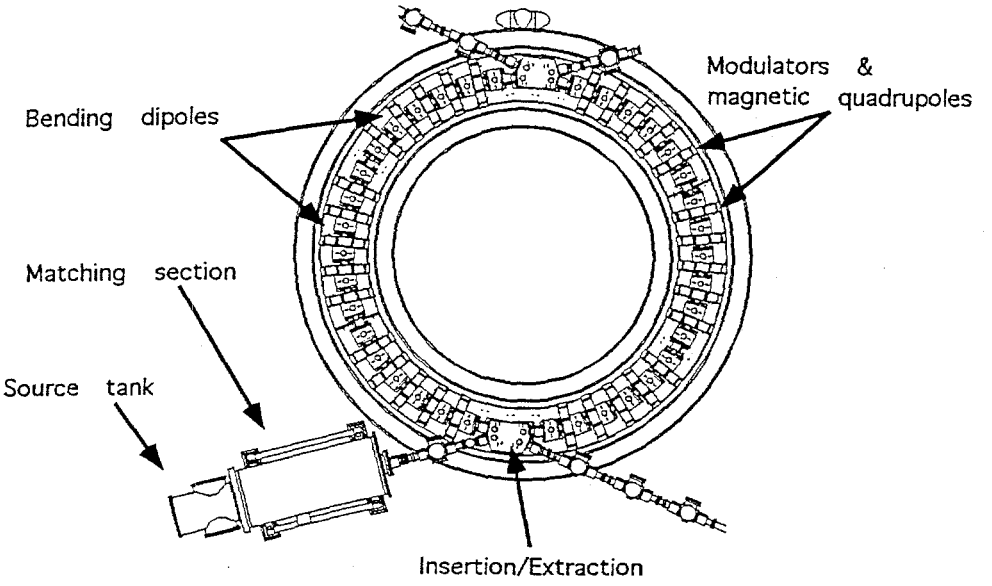


Figure 2.2: Final configuration schematic of the Small Recirculator ring

2.1 Small Recirculator Description

The Small Recirculator is being constructed on a "build and test" schedule allowing for component testing and beam experiments on that portion available at various milestones during construction. When completed, the machine will have a circumference of 14.4 m, beam pipe radius of 3.5 cm, two beam insertion/extraction sections, and an alternate-gradient focusing half-lattice period of 38 cm, allowing 40 half-lattice periods (populated by induction acceleration modules, electrostatic bending dipoles, and permanent magnet transverse focusing quadrupoles).[13]

2.1.1 Full Ring Operation

Full ring operation is expected to include the study of intense beam dynamics over 15 laps equating to a path length of about 216 meters. Sophisticated numerical modeling of the beam using LLNL's WARP code predicts satisfactory beam trans-

port through the 15 laps.[14] Data from the full ring operation as well as partial ring operation during construction will be used to validate the code's initial beam conditions that are assumed regarding the ion distribution and gage the design operation of the accelerator system. To provide for the technology development needed for a full-scale recirculating ion driver in the areas of beam control, longitudinal confinement, induction modulator performance and energy recovery, and beam manipulation (insertion/extraction) with minimal emittance growth, the beam of singly-charged potassium ions will be accelerated from 80 to 320 keV by 34 induction modules. The beam pulse, initially at a current of 2 mA, 4 μs in length and 1.1 cm average radius will be accelerated and longitudinally compressed over 15 laps resulting in a beam pulse with a current of 8 mA, 1 μs in length, and average radius of 1.3 cm. The initial phase advance of the betatron motion of the ions is $\sigma_0 = 78°$ in the absence of space-charge and $\sigma = 16°$ in the presence of space-charge. At final beam energy, the phase advances will decrease to $\sigma_0 = 45°$ and $\sigma = 12°$ per lattice period.

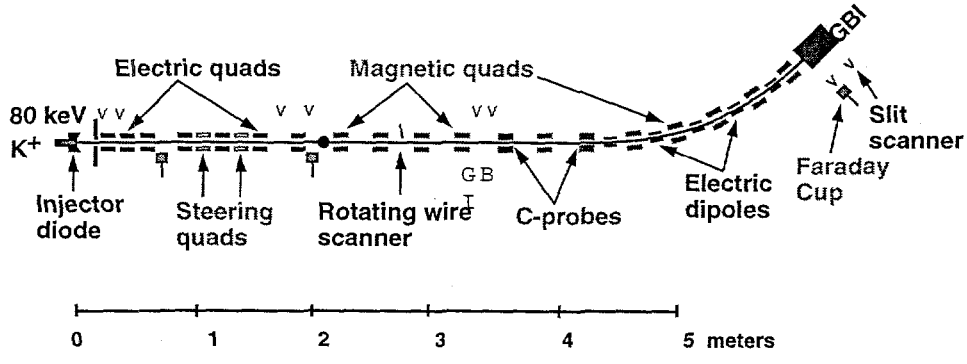


Figure 2.3: Experimental configuration of the Small Recirculator during the initial deployment of the Gated Beam Imager

2.1.2 Partial Ring Operation

During the experiments carried out in support of this thesis the configuration of the Small Recirculator was that shown in Figure 2.3 with a 45° bend. Figure 2.4

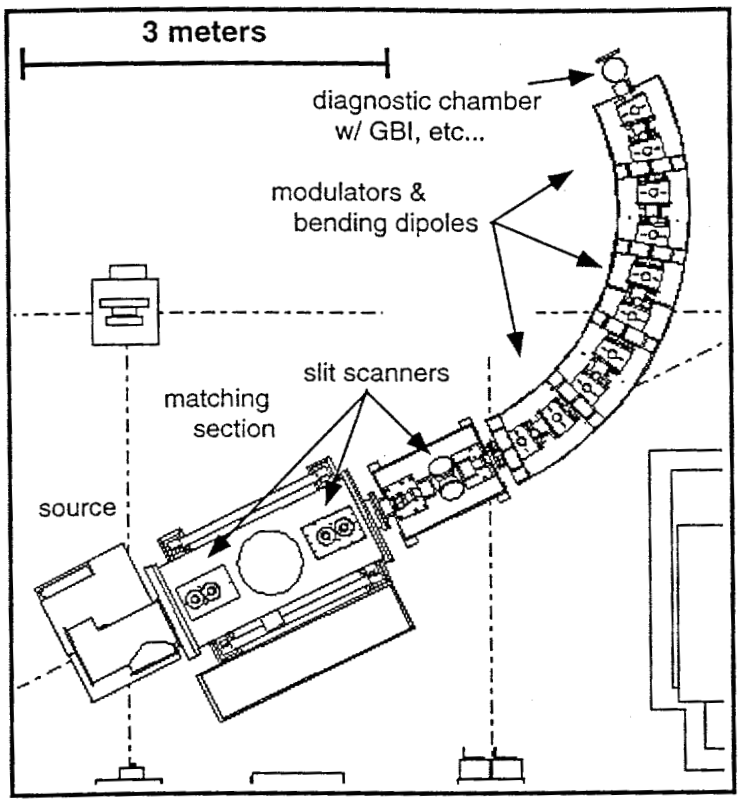


Figure 2.4: Experimental configuration of the Small Recirculator during the deployment of the Gated Beam Imager at the end of the 90° bend.

shows the Small Recirculator with a 90° bend installed. In this initial phase of construction and testing, several designs for the ion source pulser power supply were tested in order to obtain the desired beam ion extraction voltage waveform rising from 0 to 80 keV in nominally 326 ns with a flattop, 4 μs long with less than 0.1% "ringing", and a sharp cut-off at pulse end. Testing of beam transport through an electrostatic quadrupole matching section into a straight magnetic transport section was completed prior to bending the beam initially around a 45° bend in April 1997. Construction of a 90° bend in started in August 1997. Bending such an intense space-charge-dominated heavy ion beam around a curve while maintaining beam quality had never been accomplished before. Heavy ion beams are much more difficult to control than electron beams owing to their particle's large mass (a factor of 7×10^7 for

potassium ions compared to electrons), requiring much larger applied fields and less room for error. Sophisticated particle-in-cell simulations, such as WARP, are required to accurately calculate the necessary fields and the shape of the elements that produce the fields such that higher order field components that produce emittance growth are minimized.[15] The development of diagnostic instruments able to measure the subtle changes expected in the beam into and out of bends was a major effort at this stage of construction and testing. Development of the prototype GBI occurred over $2\frac{1}{2}$ years using the heavy ion beam, first in the straight magnetic transport section and then at the end of the 45° bend section. Its performance was compared with data from the mechanical 2-D emittance diagnostic known as "slit scanners" adapted from LBNL experiments to the Small Recirculator and to the moment and emittance calculations of the WARP code. For final testing, the Gated Beam Imager contained within a multi-instrument diagnostic chamber was placed at the end of the 45° bend section around which the 80 keV beam drifted without acceleration. These results are discussed in Chapter 7.

2.1.3 Potassium Ion Source

At the source end of the machine, ions are drawn from a diode source that delivers up to a 9 mA beam of singly-charged potassium ions (K^+) at 80 keV.[16] The ion source tank is mounted above the source pulser power supply housed in a tank filled with diala oil to provide cooling to power supply components and to prevent high voltage breakdown between components.[17] This allows a compact physical arrangement of the pulser circuitry but does cause a considerable effort in tuning the circuit to account for stray impedances that are caused by circuit geometry which become evident at the high voltages and frequency necessary to produce the desired waveform.

The ion source is a one centimeter diameter zeolite cylinder coated with potassium. During operation the source is heated to approximately 1200°F by ohmic heating causing potassium ions to form a plasma just in front of its surface. The 80 keV extraction voltage waveform that forms the beam is applied to a graphite Pierce

electrode surrounding the source shown in Figure 2.5. The 80 keV potential is dropped across a gap to a plate held at ground potential with a one centimeter round aperature centered on the beamline. The shape of the Pierce electrode provides a small focusing effect on the plasma and the aperature limits the current into the Small Recirculator to 2 mA.[16] The extraction voltage waveform has a design rise time of 326 ns which can be adjusted by an RC network in the power supply circuit shown in Figure 2.6. A nominal 4 μs long pulse is generated with its length determined by the receipt of a second timing signal from the accelerator control system which causes a spark gap to crowbar the remaining pulser energy to ground with a fall time of 326 ns.

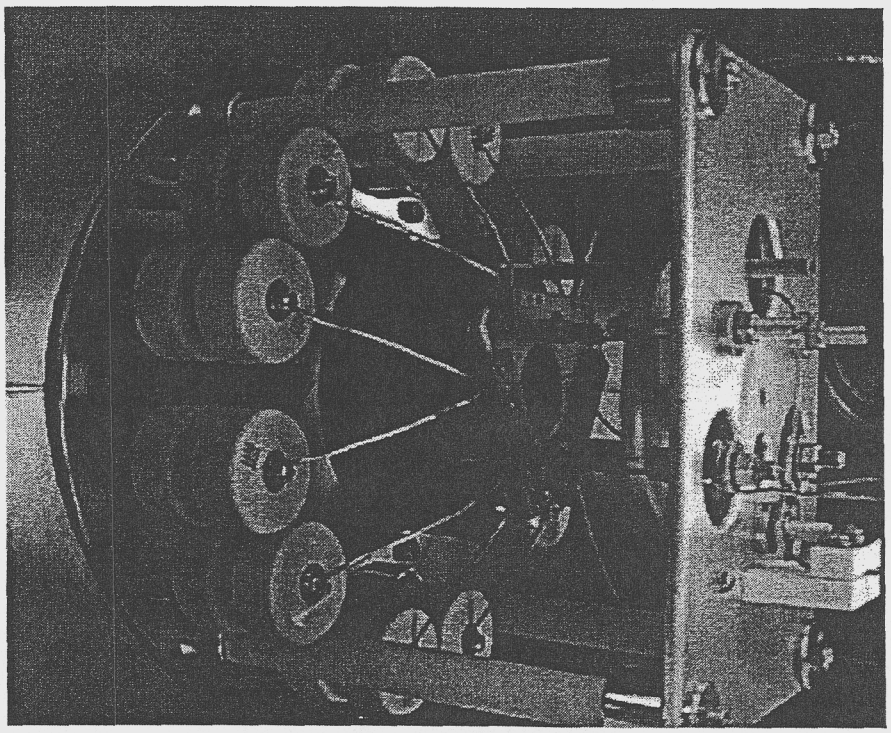


Figure 2.5: Pierce electrode (black) surrounding the potassium source (not visible) with power supply capacitors mounted on the periphery. The one centimeter diameter aperature to the matching section is to the left of the Pierce electrode

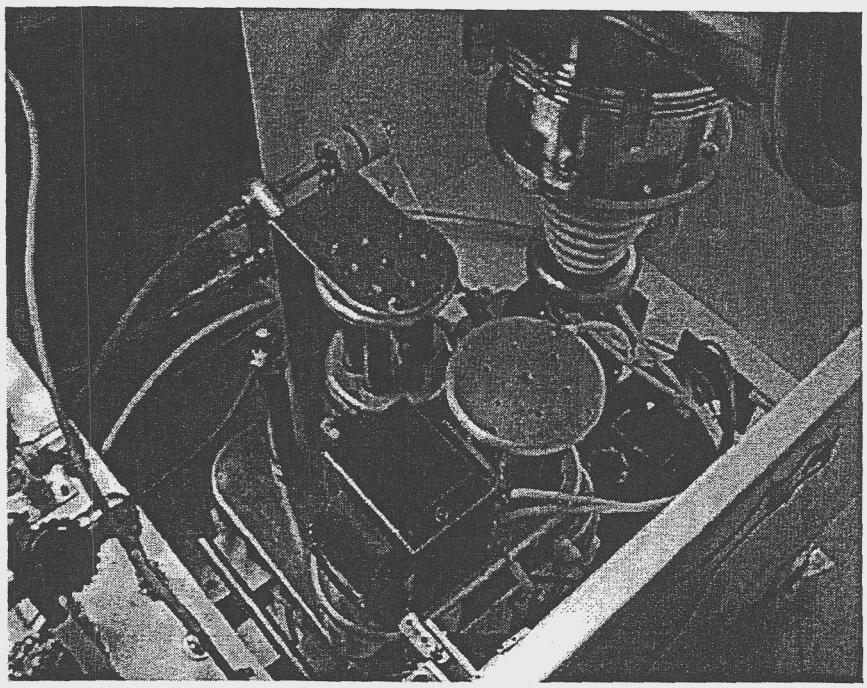


Figure 2.6: Source pulser power supply circuit in its oil tank below the ion source assembly. (Oil not present)

2.1.4 Electrostatic Matching Section

The one centimeter aperature passes the beam into the electrostatic matching section formerly used in LBNL's Single Beam Transport Experiment shown in Figure 2.7. The matching section contains seven electrostatic quadrupoles. High voltages whose values are calculated using the WARP code are applied to the quadrupoles which "matches" the beam by inducing the proper focusing in one transverse direction (e.g., x or y, with z being the axial direction) while defocusing in the orthogonal transverse direction. The correct combination of voltages and quadrupole polarity creates a beam envelope oscillation that matches the beam envelope to the orientation and spacing of the alternating gradient focusing elements of the magnetic transport section of the Small Recirculator. A correct match results in a beam envelope that remains less than the beam pipe's aperature dimension. Table 2.1 lists a typical set

of electrostatic quadrupole voltages used during experiments to attain a matched 80 keV beam.

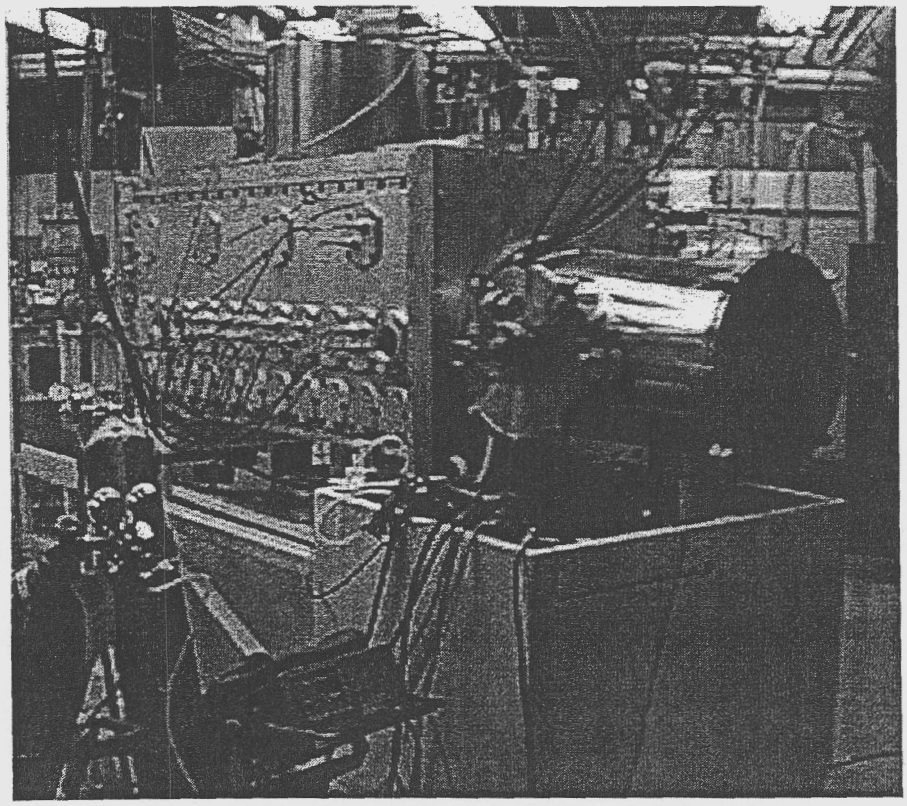


Figure 2.7: Electrostatic matching section. Cables supplying voltage to the electrostatic quadrupoles enter on the left side and the oil tank housing the ion source pulser circuit is shown below the ion source assembly tank to the right.

Faraday cup diagnostic instruments that measure total beam current as a function of time can be inserted into the matching section after electrostatic quadrupole numbers two and seven. Two sets of mechanical slit scanners, which can measure beam emittance in the y (vertical) direction only, are located between quadrupoles two and three and between quadrupoles six and seven. A set of electrostatic steering quadrupoles for moving the beam a limited amount in the $\pm x, y$ direction are located after quadrupoles four and five.

Table 2.1: Electrostatic quadrupole voltages used to "match" the Small Recirculator 80 keV beam to the magnetic transport section lattice.

Quadrupole #	Match Voltage (V)
1	±1975
2	±3836
3	±3085
4	±1955
5	±1840
6	±1655
7	±1665

2.1.5 Straight Magnetic Transport Section

Following the matching section, during the straight magnetic transport experiments, were three magnetic transport sections, each 76.2 cm long, with two six-inch diagnostic or pumping ports and connections for capacitance probes (non-intercepting beam centroid diagnostics).[18] The six-inch ports accepted a variety of diagnostic instruments including a LLNL-designed mechanical slit scanner, a rotating wire scanner for beam envelope measurements, and test assemblies used for GBI development. Small one-inch ports were used to bring out capacitance probe signals, and to accommodate GBI detector power supply leads. Each straight section consisted of two alternating gradient (or FODO, short for FOcusing/DefOcusing) half-lattice periods with two permanent magnetic quadrupole focusing elements with pole tip fields of 0.295 Tesla. Successive quadrupoles are aligned such that focusing in one transverse plane is followed by defocusing in the next quadrupole in the same plane. With a matched beam, the beam waist, (where both transverse radii of the beam are equal and the beam crossection is round), is at the center of the six-inch diagnostic ports. Figure 2.8 shows the three straight magnetic transport sections as configured during the experiments supporting this thesis.

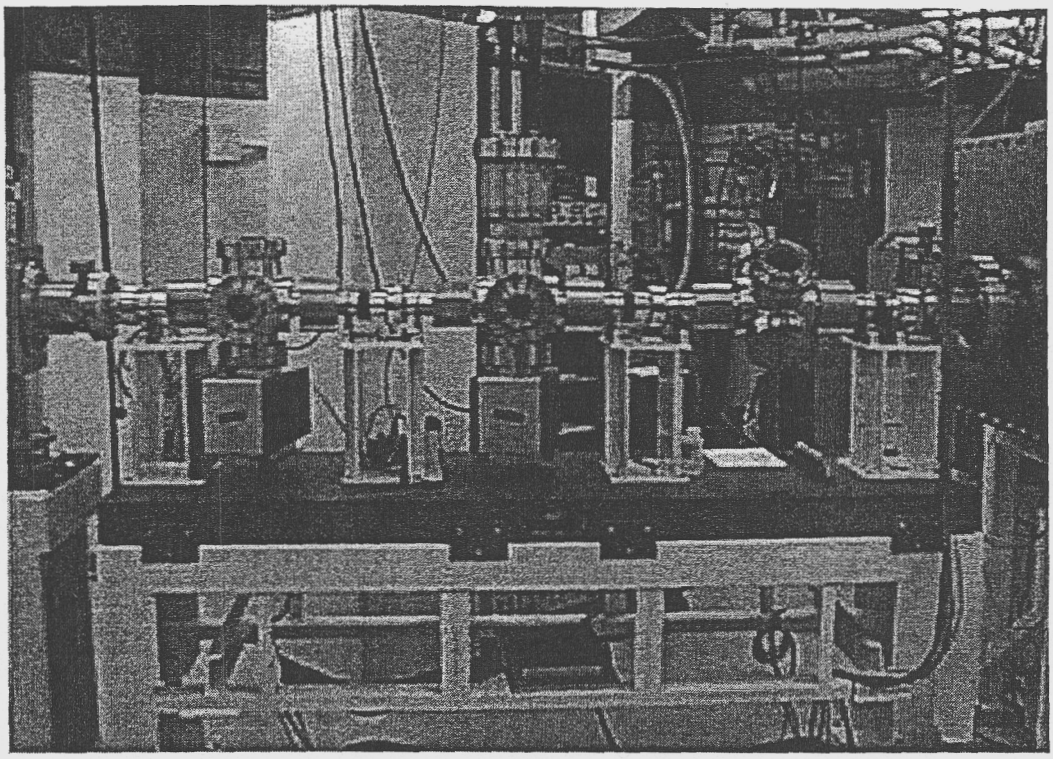


Figure 2.8: Small Recirculator straight magnetic transport section showing the 3 six-inch diagnostic ports.

2.1.6 Electrostatic Bending Section

Figure 2.9 shows the 45° bend section that first accomplished the bending of an intense heavy ion beam with full beam transport.[18] The FODO lattice of the straight transport continues in the bend with electrostatic dipole bending plates placed at the center point between successive magnetic quadrupoles. These plates are designed to bend the beam by 9° in each half-lattice period requiring 5 dipoles to bend the beam by 45°. The nominal voltage to bend the beam 9° at 80 keV is +6570 V placed on the outer plate and -6570 V on the inner plate. The shape of the faces of the bending dipoles is slightly concave on the inner electrode and convex on the outer with "ridges" along the top and bottom edges. The shape, calculated using the WARP code was optimized to minimize the energy effect which shortens/lengthens the phase advance

of particles not on the design orbit through the dipole and to minimize higher order field components.[15]

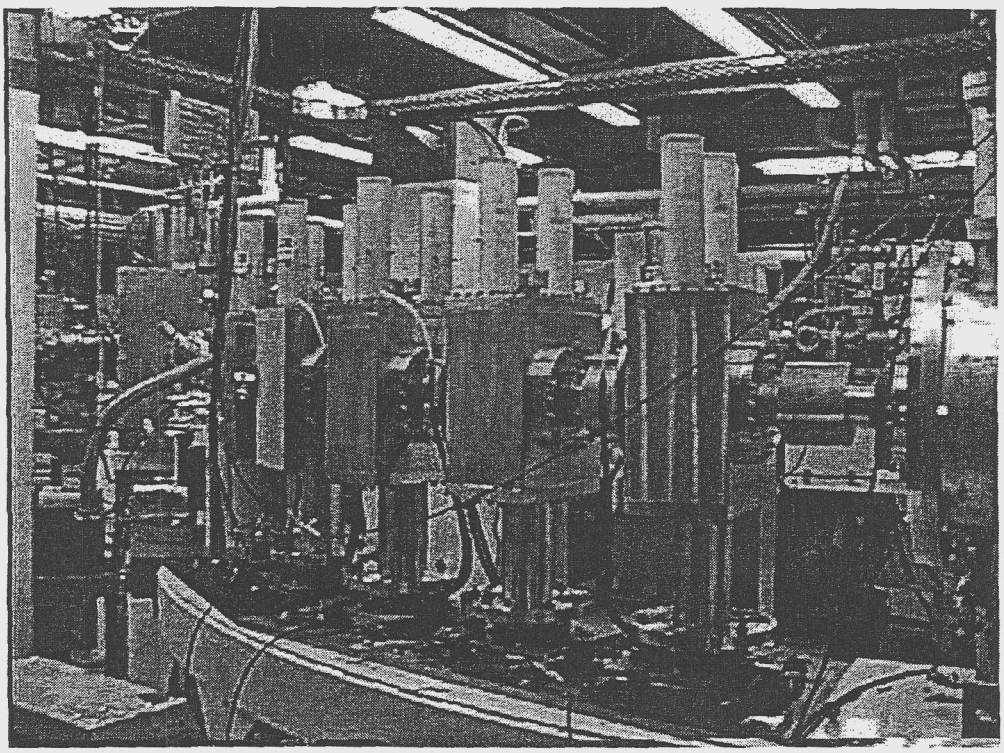


Figure 2.9: Small Recirculator 45° bend section consisting of five half-lattice periods with electrostatic bending dipole plates.

2.1.7 Multi-Instrument Diagnostic Tank

No induction modulators were placed on the 45° bend section at this point of the build and test schedule. They were being developed and bench tested during this time period. As such, no acceleration of the beam was accomplished during this phase of Small Recirculator operation and the beam drifted from extraction to the end of the beamline after the 45° bend with no longitudinal confinement of beam ions. Measurement of the beam distribution's transverse moments was made at the beamline end in the multi-instrument diagnostic chamber I designed and built for acquiring

data (Figure 2.10). The instruments contained in the chamber include a Faraday cup for measuring the total beam current, a mechanical slit scanner emittance diagnostic capable of 2-D phase space moment measurements, and the GBI diagnostic. This chamber will continue to be used at the end of the beamline as the Small Recirculator is built, eventually housing the final diagnostics following beam extraction after 15 lap operation.

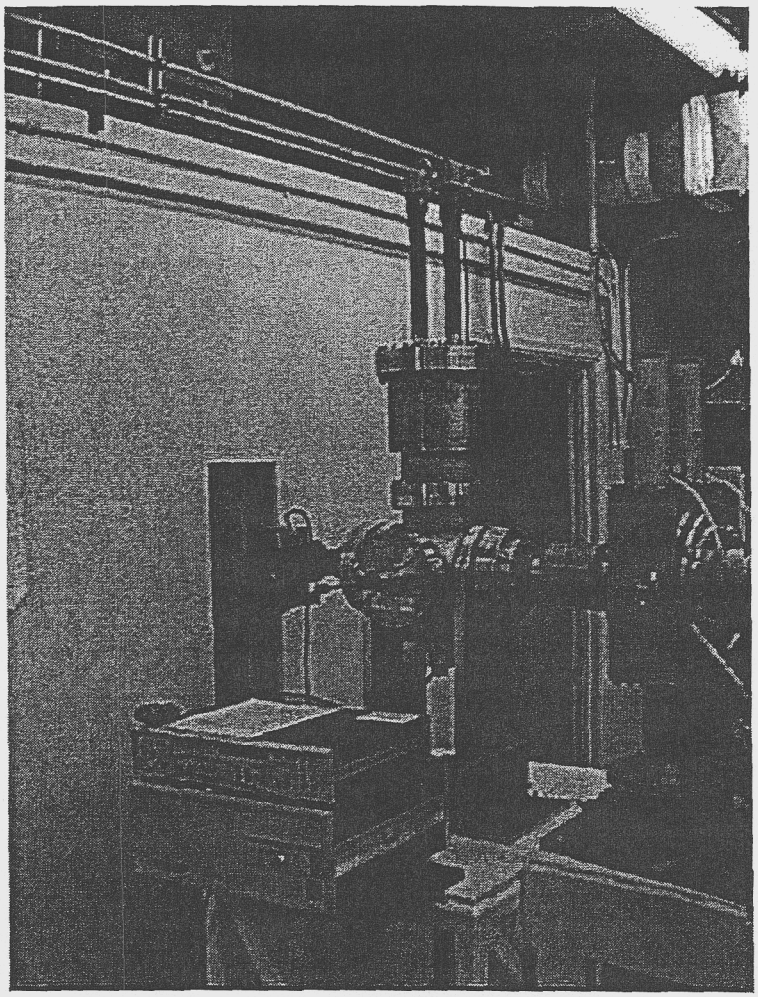


Figure 2.10: Multi-instrument diagnostic chamber at the beamline end showing the GBI CCD camera, the slit scanners mounted vertically for y-emittance measurements, and the Faraday Cup sliding manipulator on the horizontal port. The two boxes contain the controls for the GBI CCD camera. Slit scanner measurements of the x emittance is accomplished by mounting the scanners horizontally on the chamber.

2.2 Beam Physics Scaling of the Small Recirculator

That the Small Recirculator operate in the same beam physics regime as a full-scale driver is key to benchmarking the numerical simulation codes that will be used as aids to establish confidence in predicting the performance of a driver design. A LLNL design study conducted in 1991 proposed a "C-design" recirculator driver as an option for a full-scale IFE power plant.[10] A schematic is shown in Figure 2.11. Appropriate scaling of key parameters from the Low Energy Ring of the C-design driver size machine to the Small Recirculator was done to ensure the data generated in the Small Recirculator experiment would apply to a scaled-up machine.[14] A comparison of the small recirculator to the C-design Low Energy Ring is shown in Table 2.3 at the end of this chapter on page 23.

2.2.1 Physical Size Scaling

Properly scaling the physical size of the Small Recirculator was the first consideration. The physical length of both the bending dipoles and focusing magnetic quadrupoles should be greater than 3 times the pipe radius or 1.5 times the aperture between the bending dipole plates.[14] This condition ensures that the nonlinear fields associated with the finite length of the dipole and quadrupole are sufficiently small to minimize emittance growth. An additional length equal to the pipe radius is allotted on either end of each dipole and quadrupole yielding a ratio of L/r_{pipe} which is nearly identical to that of the C-design Low Energy Ring. A second scaling consideration is that the ratio of the beam pipe radius to the maximum beam radius be greater than 1.4.[14] This rule of thumb gives adequate clearance to allow for beam motion within the beam pipe and ensures the image charge forces are unimportant.

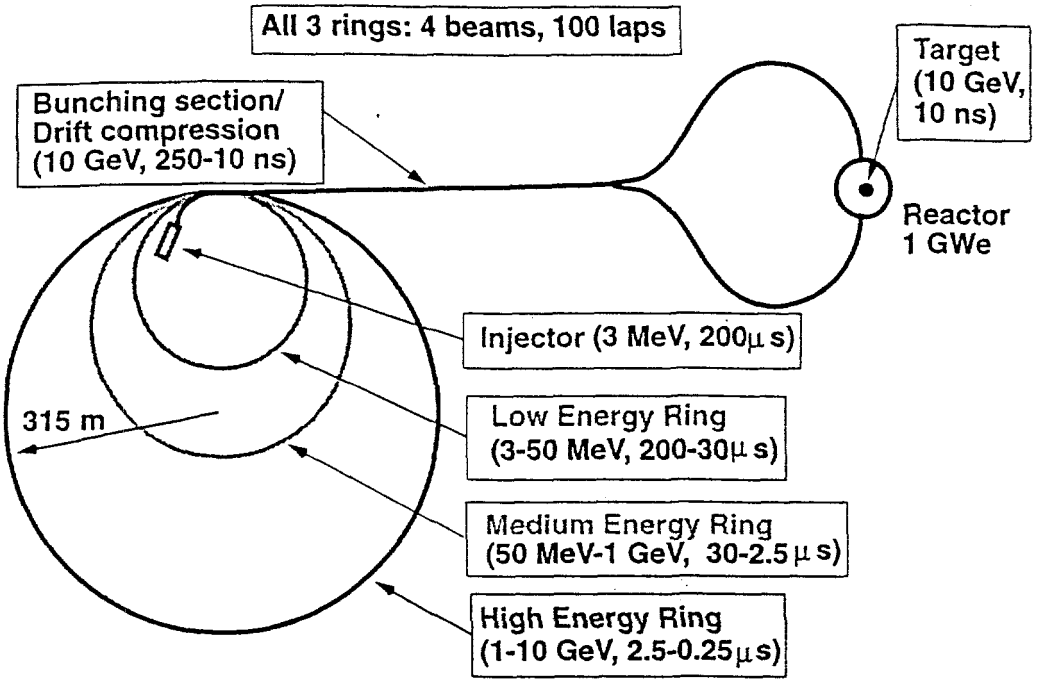


Figure 2.11: "C-design" heavy ion driver schematic

2.2.2 Maintaining Key Dimensionless Beam Physics Parameters

Maintaining the values of three dimensionless parameters which characterize the focusing, space-charge, and inertial forces, which must balance to maintain beam equilibrium, ensures that the beam physics regime of the Small Recirculator is the same as that of a full scale driver. The first parameter, the phase advance per lattice

period, σ_0 is a measure of the strength of the focusing forces relative to an ion's inertia. A resultant focusing of the beam occurs in an alternating gradient focusing system as follows. The beam is focused in the plane of the largest transverse beam envelope size and at the same time defocused in the orthogonal transverse plane of smallest envelope size. This is accomplished with quadrupole magnet fields aligned to the $x-y$ plane. Since the quadrupolar focusing force is linearly proportional to distance from the beam pipe axis, the result is net focusing through a series of magnets with focus/defocus axes rotated 90° to one another. The ions at larger envelope dimension from the axis receive a greater inward focusing force than the outward defocusing force applied to the outer ions at smaller envelope dimension in the orthogonal plane. When the beam centroid is displaced from the axis it oscillates harmonically in the focusing field and the change in phase per lattice period is the phase advance, σ_0. Experimental studies on Lawrence Berkeley National Laboratory's Single Beam Transport Experiment (SBTE) showed that beam instabilities formed at σ_0 greater than 80° which provides a constraint for both the driver and Small Recirculator.[10]

The beam perveance, κ is the second dimensionless constant and is a measure of the space-charge forces relative to inertial forces. The expression for beam perveance is:[11]

$$\kappa = \frac{qeI_0}{2\pi\epsilon_0 m_0(\beta\gamma c)^3} \tag{2.1}$$

where q is the charge state of beam ions, e is the proton charge, m_0 is the beam ion mass in atomic mass units, γ is the Lorentz factor, βc is the ion velocity, I_0 is the beam current, and ϵ_0 is he permittivity of free space. Scaling for the same perveance, ($\kappa = 10^{-3} - 10^{-4}$) as that in the C-Design Low Energy Ring is accomplished in the Small Recirculator by lowering the current, ion mass, and ion velocity appropriately.

The third dimensionless parameter is the depressed phase advance, σ. In intense ion beams, the externally-applied focusing forces are countered by the outward force generated by space-charge and random ion thermal motion. The random ion thermal motion results in a thermal force and is characterized by the emittance. In space-

charge-dominated ion beams of high line-charge density, the space-charge forces nearly equal the applied focusing forces. The emittance force makes up the difference. An off-axis beam particle performs quasi-harmonic oscillations in the combined external focusing and internal space-charge fields. The period of oscillation decreases as space-charge forces approach the focusing forces. The ratio σ/σ_0 must be $\ll 1$ for a driver.[14] The ratio for the Small Recirculator varies from 0.21 at 80 keVto 0.27 at 320 keV.

Therefore, as the key dimensionless numbers are maintained in the scaling from a driver to the Small Recirculator, verification of the Small Recirculator's design performance will result in increased confidence in modeling a driver's performance. This along with the technology developed to successfully transport and control the beam through multiple laps will pave the way to scaling up the design to that required for a full-scale driver capable of economically delivering ion beams to a reactor chamber(s) many times a second.

Table 2.2: Small Recirculator design parameters.[13]

Ion mass (amu)	39	
Total flattop charge (C)	8×10^{-9}	
Number of laps	15	
Physical quad occupancy fraction	0.307	
Physical dipole occupancy fraction	0.291	
Pipe radius/max beam radius (x, y)	2.349, 2.348	
Pipe radius (cm)	3.493	
Circumference (m)	14.4	
Half-lattice period (cm)	36	
Half-lattice period/pipe radius	10.308	
Quad physical length/pipe radius	3.164	
Bend physical length/pipe radius	2.996	
Number of half-lattice periods	40	
Residence time (ms)	0.229	
	Initial	Final
Beam energy (keV)	80	320
Flattop pulse duration (μs)	4	1
Normalized emittance (mm-mr)	0.1	0.2
Transverse temperature (eV)	0.739	1.478
Quadrupole field (T)	0.295	0.295
Dipole electrode voltage ($\pm$kV)	6.789	27.155
σ_{0x} (°), σ_{0y} (°)	78, 78	44.554, 44.554
σ_0 (quadrupole only) (°)	72	34.201
σ_x,σ_y (°)	15.95, 15.96	11.55, 11.55
(σ_x/σ_{0x}),(σ_y/σ_{0y})	0.205, 0.205	0.259, 0.259
Beam velocity/c (β)	2.1×10^{-3}	4.2×10^{-3}
Flattop bunch length (m)	2.517	1.258
Perveance	3.57×10^{-4}	1.79×10^{-4}
Peak current (mA)	2	8
Line charge (μC/m)	3.6×10^{-3}	7.21×10^{-3}
Modulator Repetition rate (kHz)	43.69	87.38
Beam radius in x (mean,max) (cm)	1.11, 1.422	1.304, 1.487
Beam radius in y (mean,max) (cm)	1.11, 1.421	1.305, 1.488
Number of betatron oscillations per lap in x, y	4.333, 4.333	2.475, 2.475
Ear rise or fall time/flattop time	0.1	0.1
$\delta v/v$ from spatial compression	1.89×10^{-2}	9.45×10^{-3}
$\delta v/v$ from ears	6.73×10^{-4}	4.7×10^{-4}
Centroid offset from compression (cm)	0.231	0.354
Centroid offset from ears (cm)	0.008	0.018

Table 2.3: Comparison between the Small Recirculator and C-design Low Energy Ring (LER) parameters.[10]

	Small Recirculator	C-design LER
General		
Ion energy (MeV)	0.08-0.32	3-50
Pulse Duration (μs)	4-1	200-30
Circumference (m)	14	700
Current per beam (mA)	2-8	500-3300
No. of Beams	1	4
No. of laps	15	100
Beampipe radius (cm)	3.5	7.8
Lattice half-period (cm)	36	85
Vacuum (torr)	$1\text{x}10^{-8}$	5×10^{-11}
Induction modules		
Inner radius (m)	0.163	0.313
Outer radius (m)	0.211	0.455
Length (cm)	8.5	40.3
No. of cores	34	785
Cell voltage (kV)	0.50	0.60
Bends		
Effective length (cm)	13	15
No. of bends	40	2680
Maximum field	910 kv m^{-1}	0.90 T
Quadrupoles		
Length (effective length) (cm)	10 (10)	47 (23)
No. of quads	40	3139
Maximum quad field (T)	0.3	2.0
Dimensionless parameters		
Phase advance, σ_O (degrees)	78-44	80-20
σ/σ_0	0.20-0.27	0.06-0.09
Perveance, κ	$(4.0\text{-}2.0)\times10^{-4}$	$(9.0\text{-}0.9)\times10^{-4}$
Tune, ν	4.3-2.5	92-22

Chapter 3

Heavy Ion Beam Transport

The production, acceleration, and transport of high-quality intense heavy ion beams for use as drivers for inertial fusion energy is one of the key issues in determining the feasibility of the energy source.[6] The specifications for the ion beams are determined by target physics considerations. Recent simulations using the code "LASNEX" have shown that to ignite one class of indirect-drive target tailored to heavy ions, the ions need to have an energy of about 4 GeV, a total beam energy of 5.8 MJ, and a spot size on target of about a few millimeters.[8] The ions must be extracted from an ion source and manipulated within the driver in order to produce the desired beam spot size and energy deposition rate at the target. One key aspect of this process is the drift of the beam from the final focus lens across the target chamber to the target, a distance of about 4 meters. In a HYLIFE II-type IFE power plant the target chamber is necessarily large to provide a lower neutron fluence at the chamber first wall for materials damage limitations and to accommodate adequate liquid coolant/shielding within the reactor chamber.[5] This large target chamber drift distance with no confining fields (except perhaps that of partial neutralization from the beam passing through chamber background gas) places a considerable constraint on the final focus lens. Among the detrimental processes during the chamber drift is the defocusing effect of the beam ions' random transverse thermal motion characterized by the beam transverse emittance. To account for this, a driver will have a transverse emittance specification at the final focus to ensure the required spot size

on target is achieved. Measuring the emittance at the final focus and most likely at various locations along the beamline will be needed to ensure that this specification is met.

3.1 Origin of Beam Emittance

At the ion source, the extracted ions have an intrinsic thermal velocity spread which remains present in the beam from the source to target. This spread is present in both magnitude and direction of the extracted ions. During beam extraction, acceleration, and final focus, the velocity vector of each ion is changed under the influence of the applied electromagnetic fields of transverse focusing elements, the axial accelerating fields of induction modules, and by beam internal space-charge forces. In fact, the existence of a beam and not a bunch of randomly moving ions is owed to the induced directed motion of the ions along the accelerator swamping this random component. Despite focusing and acceleration of a beam, the intrinsic random ion velocity component remains and cannot be removed (except by cooling, generally deemed impractical in a driver). This is a consequence of Liouville's Theorem stating the volume of a particle ensemble in phase space cannot be reduced. The random component of an ion's velocity vector in a laboratory beam will vary considerably from the ideal thermal limit due to nonlinear focusing forces (unavoidable with practical focusing fields), and instabilities in the beam particle distribution (available free energy converted to random ion motion). Sophisticated modeling of the ion beam using the WARP code has shown that transitions between straight and curved sections of an accelerator are a major cause of emittance growth.[50] Controlling this growth is dependent on the accelerator design and performance, and is a major goal of the Small Recirculator project.[20]

In a conceptual six-dimensional Euclidean phase space, combining configuration space (x, y, z) and canonical momentum space (p_x, p_y, p_z), a beam ion is represented by a point and all the points occupy a volume. This 6-D phase space can be reduced to a 4-D transverse phase space of (x, x', y, y') whose volume is conserved with the

assumption of no coupling between the transverse (x, y) and the axial (z) motions of the particles as the beam travels through the accelerator. The canonical momenta are replaced by their corresponding transverse angles, $x' = dx/dz$, and, $y' = dy/dz$. Also, assuming a system of non-interacting particles, the movement of the beam ions is associated with the equivalent motion of points in phase space. Assuming no ions are lost from the beam, the number of points in phase space and the occupied volume is constant. The projections of the 4-D volume onto the two 2-D, x and y phase spaces results in two areas that are conserved if there is no coupling between the x and y motions of the ions. Figure 3.1 is a 2-D visualization of the 4-D phase space and the two 2-D projections. The assumption of no coupling between the orthogonal motions of the ions in a laboratory beam is most likely a weak one. Experimental detection of the loss of directed energy from the z direction to the $x - y$ direction by making measurements on the z direction energy is unlikely. More feasible is the measurement of the longitudinal directed energy coupling by measuring the amount gained in the transverse directions. This coupling can be observed by detecting an increase in the emittance in the transverse directions. Direct detection of coupling between the transverse directions is more likely as the magnitude of any energy transfer would be a much greater relative percentage of the total transverse energy.

3.2 Small Recirculator Beam Transport

Nominally, the Small Recirculator beam consists of a distribution of $5\text{x}10^{10}$ ions which are initially extracted from the ion source with an energy of 80 keV. Unless accelerated by induction modules this beam "drifts" through the lattice of focusing and bending elements of the machine. The Small Recirculator lattice is a periodic focusing system with two planes of symmetry ($x - z$ and $y - z$). In the laboratory, x is horizontal, y is vertical, and z is along the beamline which curves to the left around the bend. The paraxial motion of beam particles in the lattice (neglecting bending

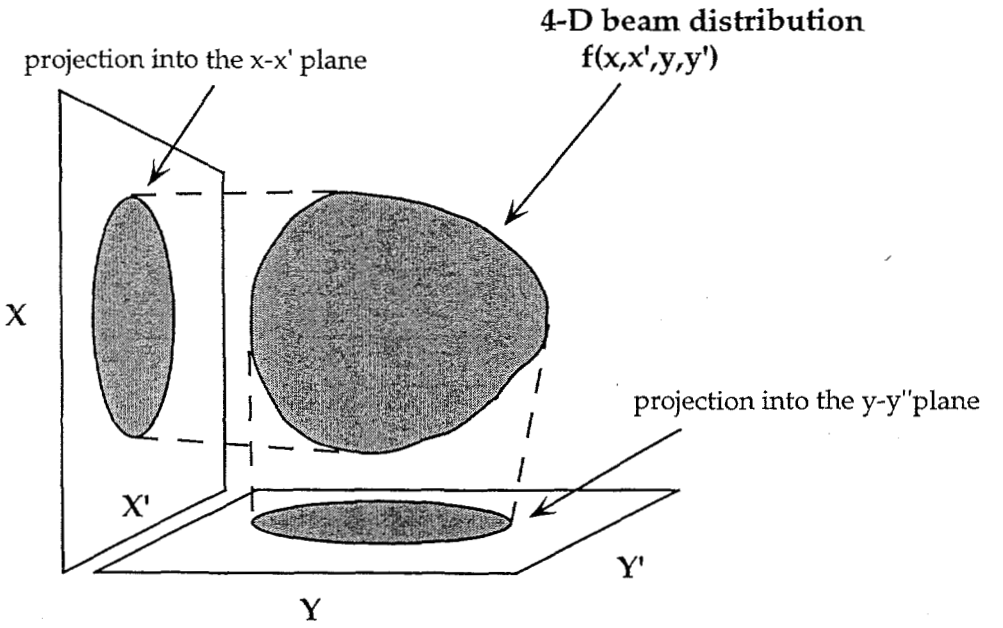


Figure 3.1: Representation of the beam's 4-D transverse phase space volume and the 2-D projections into the x and y planes.

and space-charge forces for now) is described by two equations:

$$x''(z) + k_x(z)x = 0 \qquad y''(z) + k_y(z)y = 0 \tag{3.1}$$

where $k_x(z)$ and $k_y(z)$ are the periodic focusing functions and satisfy the periodicity relation $k(z + L) = k(z)$ where L is the length of one lattice period, 76.2 cm.

Similarly, two sets of equations are necessary to describe the ion trajectories in the two planes of symmetry with space-charge taken into account now. These two sets of equations are coupled through the beam space-charge terms. Letting $X(z)$ denote the x-envelope and $Y(z)$ denote the y-envelope, the ellipse describing the transverse beam boundary obeys the equation

$$\frac{x^2}{X^2} + \frac{y^2}{Y^2} = 1 \tag{3.2}$$

and the charge density is defined by

$$\rho(z) = \{\rho_0 \quad for \quad \frac{x^2}{X^2} + \frac{y^2}{Y^2} \leq 1\,, \qquad 0 \quad for \quad \frac{x^2}{X^2} + \frac{y^2}{Y^2} > 1. \tag{3.3}$$

where

$$\rho_0 = \rho_0(z) = \frac{I}{\pi v X(z) Y(z)} \tag{3.4}$$

is constant inside the beam at a given position z, (x, y varying) but is a function of z. The electric field for this charge distribution calculated from Poisson's Equation is:

$$E_x = \frac{I}{\pi \epsilon_0 v} \frac{x}{X(X+Y)} \qquad\qquad E_y = \frac{I}{\pi \epsilon_0 v} \frac{y}{Y(X+Y)} \tag{3.5}$$

The ion trajectory equations including space-charge are:

$$x''(z) + k_x(z)x - \frac{2\kappa}{X(X+Y)}x = 0, \tag{3.6}$$

$$y''(z) + k_y(z)y - \frac{2\kappa}{Y(X+Y)}y = 0 \tag{3.7}$$

where κ is the beam perveance defined in Equation 2.1. These equations are linear in x and y, but are coupled through the space-charge terms which can be determined from the corresponding beam envelope equations.[21] The trajectory equation solutions can be represented in phase-amplitude form:

$$x(z) = A_x \omega_x(z) \cos[\psi_x(z) + \phi_x] \qquad\qquad y(z) = A_y \omega_y(z) \cos[\psi_y(z) + \phi_y] \tag{3.8}$$

where A_x, A_y, ϕ_x, and ϕ_y depend on the initial conditions (x_0,x_0') and (y_0,y_0') and remain constant throughout the motion. The terms, $\omega_x(z)$, and $\omega_y(z)$ give the amplitude dependence on z as the beam is transported through the alternate gradient focusing lattice. Differentiating $x(z)$ and $y(z)$ with respect to z gives

$$x' = A_x[\omega'_x \cos(\psi_x + \phi_x) - \omega_x \psi'_x sin(\psi_x + \phi_x)] \tag{3.9}$$

$$y' = A_y[\omega'_y \cos(\psi_y + \phi_y) - \omega_y \psi'_y sin(\psi_y + \phi_y)] \tag{3.10}$$

Eliminating ψ and ϕ using the identity $\cos^2(\psi+\phi)+\sin^2(\psi+\phi)=1$ yields the equation of the 2-D phase space trajectories of an ion.[22]

$$\hat{\gamma}x^2+2\hat{\alpha}xx'+\hat{\beta}x'^2={A_x}^2 \qquad \hat{\gamma}y^2+2\hat{\alpha}yy'+\hat{\beta}y'^2={A_y}^2 \tag{3.11}$$

These are equations of ellipses whose shape and orientation are determined by the amplitude factor, A, and coefficients $\hat{\gamma}$, $\hat{\alpha}$, and $\hat{\beta}$ which are determined by $\omega(z)$, and $\omega'(z)$. All ions in the beam with the same amplitude factor but differing ϕ lie on the same ellipse. Ions with different amplitude factors lie on other ellipses scaled in size but similar in shape. The relationship between the coefficients and the ellipse dimensions are shown in Figure 3.2. For the $x-x'$ plane, and similarly for the $y-y'$ plane,the area of the largest ellipse is $A_0=\pi\epsilon_x$ where ϵ_x is the beam emittance in the x direction. Therefore, the equation of the largest ellipse can be written as

$$\hat{\gamma}x^2+2\hat{\alpha}xx'+\hat{\beta}x'^2=\epsilon_x^2 \tag{3.12}$$

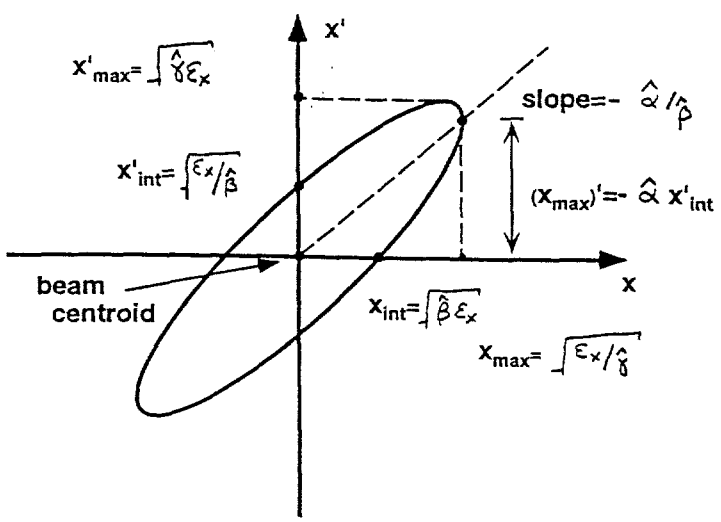


Figure 3.2: Relationship between phase space ellipse coefficients and the ellipse dimensions.

As the beam drifts through the Small Recirculator lattice the emittance would be constant if the external focusing and bending fields and the space-charge fields were linear.[23] With acceleration, the emittance decreases since $x' = dx/dz$, but the normalized emittance remains constant as it includes factors which account for the increase in ion velocity. A discussion of the use of different emittance definitions follows later in this chapter. The only ion distribution that results in linear space-charge fields is the Kapchinsky-Vladimirski (K-V) distribution of the form $f(x, x', y, y') = F_0\delta(F - F_0)$.[24] The point representation of this distribution is that all the points in a 4-D phase space representing ions lie on the surface of the 4-D hyperellipsoid of Figure 3.1. The projection of the hyperellipsoid onto the 2-D phase spaces results in ellipses with a uniform charge density.

A fundamental question of beam theory is to find equilibrium states where the ion distribution remains stationary (i.e., does not change with distance along the accelerator). Vlasov theory is a tool used in the absence of collisions to find equilibrium states. A stationary distribution represents a state of minimum total energy. A beam with higher total energy contains free energy which is available to be converted into random ion energy, thus increasing the emittance of the beam. This conversion is brought about by collisional processes, instabilities, and non-constant space-charge distributions and nonlinear focusing fields.[25]

Analytical techniques using the Vlasov equation to understand the stationary state or the evolution of a non-stationary state are useful only in axisymmetric cases and in the case of the K-V distribution.[26] Computational particle simulations must be used for other distributions such as the constant density with a semi-gaussian spread in ion velocities used to initialize WARP simulations.[15] The initial magnitudes of the velocity spreads are, to date, only estimates in the code. A result of code simulations has been to show that the transverse emittance growth of a Small Recirculator beam as a result of the insertion/extraction into the ring is very sensitive to the magnitude of the axial velocity (v_z) spread. Determining the proper spread by comparing experimental measurements to the code is of vital importance to code validation.

The results of the analytical analysis of the K-V distribution can be used on unknown (laboratory) distributions, which are likely not K-V, using the concept of "equivalent beams".[26,27] Treating the laboratory beam as an equivalent K-V beam has yielded valuable tools for characterizing accelerator performance and as a tool for designing and predicting accelerator performance.

3.3 Moment Description of the Ion Beam

Sophisticated heavy ion beam simulation codes, such as LLNL's WARP, model the evolution of the beam distribution through a model recirculator using macro-particles, each representing a large number of ions.[15] Choosing the appropriate parameters to use as code benchmarks involves surveying what can be measured in an experiment. The desirable (and reasonable) measurements include ion energy, beam envelope parameters, and transverse phase space first and second moments. These can be compared directly with code results or figures of merit such as the beam transverse emittance in the x and y directions. Relating the beam's representation in the conceptual 4-D phase space as a hyperellipsoid to its moments is seen with a matrix representation in Figure 3.1.[28] In this context, σ is not the beam phase advance, but the hyperellipsoid's sigma matrix. The difference between uses of the notation is apparent in context.

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} & \sigma_{24} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} & \sigma_{34} \\ \sigma_{41} & \sigma_{42} & \sigma_{43} & \sigma_{44} \end{pmatrix} \tag{3.13}$$

The equation of the hyperellipsoid is given by $\Gamma^T \sigma^{-1} \Gamma = 1$ where the 4-component vector, $\Gamma = (x, x', y, y')$ and the dimensions of the hyperellipsoid are related to the

matrix elements by:

$$x_{max} = \sqrt{\sigma_{11}} \qquad x'_{max} = \sqrt{\sigma_{22}} \qquad y_{max} = \sqrt{\sigma_{33}} \qquad y'_{max} = \sqrt{\sigma_{44}} \tag{3.14}$$

Assuming a K-V beam distribution, the relationship of the matrix elements to the second moments of the beam is:

$$\sigma_{11} = 4 < x^2 > \qquad \sigma_{12} = 4 < xx' > \equiv 4 < x'x > = \sigma_{21} \qquad \sigma_{22} = 4 < x'^2 > \tag{3.15}$$

$$\sigma_{13} = 4 < xy > \equiv 4 < yx > = \sigma_{31} \qquad \sigma_{23} = 4 < x'y > \equiv 4 < yx' > = \sigma_{32} \tag{3.16}$$

$$\sigma_{14} = 4 < xy' > \equiv 4 < y'x > = \sigma_{41} \qquad \sigma_{24} = 4 < x'y' > \equiv 4 < y'x' > = \sigma_{42} \tag{3.17}$$

$$\sigma_{33} = 4 < y^2 > \qquad \sigma_{34} = 4 < yy' > \equiv 4 < y'y > = \sigma_{43} \qquad \sigma_{44} = 4 < y'^2 > \tag{3.18}$$

The x and y cross terms represent coupling between the motion of beam ions in the x and y planes. These elements are zero if no coupling is present or non-zero with coupling caused by, for example, a rotationally misaligned quadrupole focusing magnet. The second moments of the distribution are defined in the conventional manner, e.g.,

$$< x^2 > = \iiiint x^2 \, \rho(x, x', y, y') \, dx \, dx' \, dy \, dy' \tag{3.19}$$

with $\rho(x, x', y, y') \, dx \, dx' \, dy \, dy'$ representing the beam ion distribution and having the normalization

$$\iiiint \rho(x, x', y, y') \, dx \, dx' \, dy \, dy' = 1 \tag{3.20}$$

Without coupling between the ions' x and y motions, the projections of the hyperellipsoid into the $x - x'$ and $y - y'$ planes in Figure 3.1 are ellipses which contain the x and y motions of the beam particles. The applicability of Liouville's Theorem to the 4-D and to these 2-D phase spaces is predicated on no coupling between the orthogonal directions of motion. The separable nature of the x and y phase space ellipses is seen in the sigma matrix with no coupling, Equation 3.21.

$$\sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & 0 & 0 \\ \sigma_{21} & \sigma_{22} & 0 & 0 \\ 0 & 0 & \sigma_{33} & \sigma_{34} \\ 0 & 0 & \sigma_{43} & \sigma_{44} \end{pmatrix} \tag{3.21}$$

Figure 3.3 shows the relationships between the sigma matrix elements and the 2-D phase space ellipse dimensions.

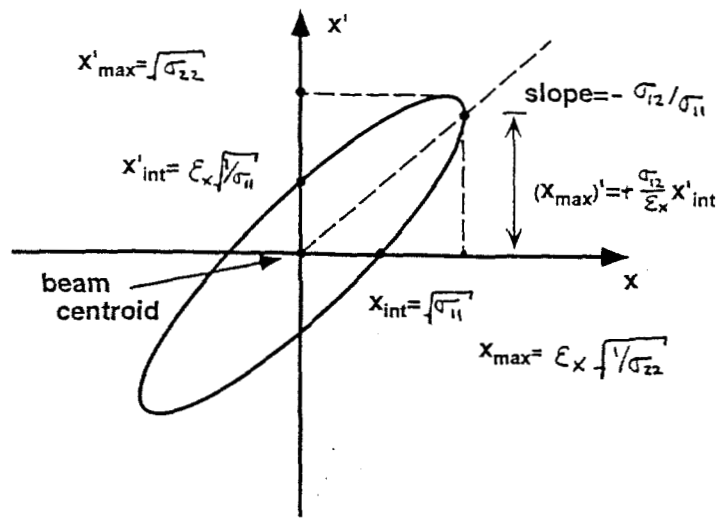


Figure 3.3: Relationship between the sigma matrix elements and the 2-D phase space ellipse dimensions.

A beam's phase space distribution can be described by its first and second moments. The actual beam distribution in a laboratory beam is not known. The usefulness of gross measurements such as the beam first and second moments is in the concept of equivalent beams. Two beams can be considered equivalent if they are composed of the same particle species, current, kinetic energy, and second moments. Detailed differences in their distributions are unimportant as long as the above constraints are met. This is fortunate as it allows comparison of a laboratory beam

of unknown distribution with that of a known and probably different distribution used in a numerical simulation. The initial distribution for the WARP code that is most commonly used is one that is uniform in x, y, z, with a semi-gaussian spread in v_x, v_y, and v_z.[15] MOMENT, a simpler beam moment evolution code, uses a K-V distribution.[29] Therefore, a simulated beam and a laboratory beam with the same second moments and other quantities at the same time and location in the accelerator can be considered equivalent, and is one basis for comparing experimental data with numerical simulation results.

3.4 Definition of Emittance

The figure of merit most commonly used to describe beam quality is the transverse emittance of the beam. Beam "brightness" is another quantity used as a figure of merit when there is concern about the amount of beam current as well as the emittance. In the Small Recirculator, with full beam transport verified with Faraday cups, the emittance is the relevant quantity. It is closely related to the area of the two-dimensional projections of the 4-D phase space hyperellipsoid. Lapostalle and Sacherer introduced rms quantities in order to compare beams with differing distributions.[26,27] The rms beam width is given by

$$\tilde{x} = < x^2 >^{\frac{1}{2}} \tag{3.22}$$

The rms values of other quantities $(\tilde{x}', \tilde{y}, \tilde{y}')$ are similarly calculated. Statistically, the definition of the "rms transverse emittance" in the x direction is

$$\tilde{\epsilon}_x = \left[< x^2 > < x'^2 > - < xx' >^2 \right]^{\frac{1}{2}} \tag{3.23}$$

where the term $< xx' >$ represents an $x - x'$ correlated particle flow term that occurs when the beam is converging or diverging under the influence of external focusing forces.[26,27] It represents an inward or outward flow term in the transverse kinetic energy. The difference between the total transverse kinetic energy, $< x^2 > < x'^2 >$, and

the flow energy represented by $<xx'>$ is the small random transverse kinetic energy from which the emittance rises. Correspondingly for the y direction:

$$\tilde{\epsilon}_y = \left[<y^2><y'^2> - <yy'>^2\right]^{\frac{1}{2}} \tag{3.24}$$

Another common definition I will adopt for clarity is the "rms edge emittance" which is four times the rms emittance[26] :

$$\epsilon_x = 4\tilde{\epsilon}_x = 4\left[<x^2><x'^2> - <xx'>^2\right]^{\frac{1}{2}} \tag{3.25}$$

In terms of the hyperellipsoid sigma matrix:

$$\epsilon_x = \left[\sigma_{11}\sigma_{22} - \sigma_{12}^2\right]^{\frac{1}{2}} \tag{3.26}$$

As x' is defined as dx/dz, the instantaneous transverse angle with respect to motion along the accelerator, it is inversely proportional to the beam energy. A correction factor of $\beta\gamma$, the usual relativistic quantities, is applied to the emittance to give the normalized rms edge emittance which is constant with beam acceleration:[30]

$$\epsilon_n = \beta\gamma\epsilon_x \tag{3.27}$$

This is a figure of merit that can be easily used to compare the beam emittance if measurements are made at two points in an accelerator with acceleration in between.

There are ten independent transverse second moments. The existence of non-zero cross moments can be indicative of rotational misalignments of accelerator quadrupolar focusing components causing coupling between the x and y ion motions. The effects of misalignments are more fully discussed in Chapter 6. These $x - y$ cross moments cannot be measured with a slit scanner diagnostic as this instrument averages beam information along the length of its slits. A useful quantity that becomes non-zero with $x - y$ coupling is indicative of the angular momentum developed by the

beam about the z-axis. This quantity is defined by:

$$l = \beta(<x'y> - <xy'>) \tag{3.28}$$

Detection of beam rotation can indicate alignment problems in the accelerator which will have the ultimate effect of increasing the final spot size on target over that of a similar non-rotating beam.

The GBI measures these $x - y$ cross moments and extends the characterization of the Small Recirculator beam and understanding of possible causes of unpredicted beam behavior beyond that of previous instruments.

Chapter 4

Pepperpot Method and Image Data Analysis

The motivation to design a new diagnostic instrument for measuring the characteristics of a heavy ion beam was fostered by the limitations of previous instruments. The new instrument developed in support of this thesis is the Gated Beam Imager (GBI) and is based on the pepperpot beam diagnostic method which has previously been used to diagnose electron beams.[28] The diagnostics in predominant use to determine a heavy ion beam distribution's first and some of the second moments in transverse phase space have been capacitance probes and mechanical slit scanners.

4.1 Capabilities of Existing Emittance Diagnostics

Mechanical slit scanners, in which two thin slits are repositioned across the beam between successive shots are able to measure the second moments ($<x^2>$,$<xx'>$,$<x'^2>$ or, $<y^2>$,$<yy'>$,$<y'^2>$) in one transverse plane at a time while averaging the beam information in the orthogonal plane.[31] Thus, only the emittance in one plane at this particular place in the beam line can be obtained without disassembling the accelerator or having a second scanner at the same location, or some other extravagant solution. As a slit averages the beam in the direction of its length, a slit scanner cannot measure any of the cross moments, ($<xy>, <x'y>, <xy'>, <x'y'>$) which are important to detecting a decrease in the accelerator performance due to a focusing field degradation, or the presence of a component physical misalignment such as

a focusing quadrupole rotation error about the beam axis. These types of problems induce coupling between the x and y motions of the ions in an alternating gradient accelerator lattice causing an "effective" emittance growth in the beam. I call this "effective" emittance growth as the growth is genuine in its effect on the focusability of the ion beam. Because it is caused by linear forces coupled in the x and y directions, it can, in principle, be removed by the appropriate application of counteracting linear forces. Normally, emittance growth is associated with nonlinear forces acting on the beam distribution which entrains unoccupied phase space into the distribution making it for practical purposes impossible to sort out the actual distribution through diagnostic measurements.[25] Figure 4.1 shows the entrainment of unoccupied phase space within the distribution as a result of nonlinear forces. The dotted line envelope shows the resolvable phase space ellipse.

The GBI is able to measure all the first and second phase space moments of the ion beam simultaneously, including the $x-y$ cross moments. With appropriate time gating it can produce the information for a 4-D and time history "map" of the beam using successive beam pulses. When installed after a bending dipole such as after the 45° bend section of the Small Recirculator it was found useful as a qualitative energy analyzer to observe energy changes longitudinally along the beam pulse.

4.2 The Pepperpot Method

In the pepperpot method where a mask with small holes (the pepperpot) is introduced into the beam, the beam ions are stopped by the hole plate except where they pass through the holes forming small beamlets. These beamlets pass through a drift region where they freely expand from space-charge and emittance forces. The beamlets are intercepted by a detector which is excited and the spots are observed with a CCD camera, and digitized for analysis. Figure 4.2 is a schematic of the method.

Without the hole plate, the gross behavior of the beam in the drift space is similar

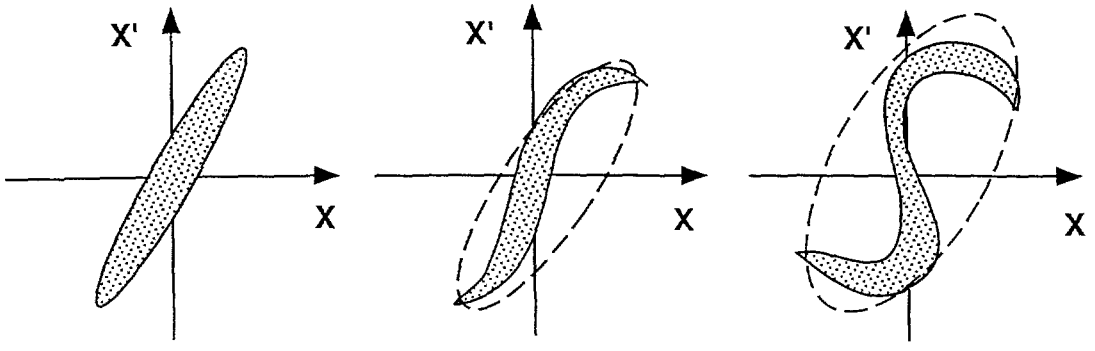


Figure 4.1: Entrainment of unoccupied space into the phase space area of the beam due to the action of nonlinear forces as the beam progresses through the accelerator. The dotted ellipses are fitted to shaded areas to illustrate how the resolvable emittance grows although the shaded areas are constant in area.

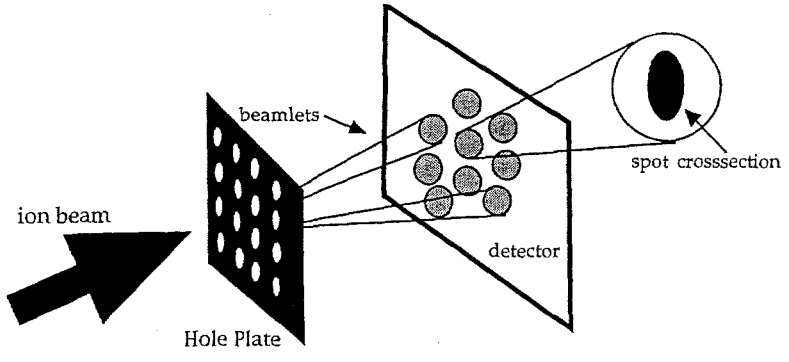


Figure 4.2: Schematic of the pepperpot method

to the beamlet but the small increase in beam dimension due to emittance forces is lost in the greater expansion due to space-charge. The small holes allow the emittance-caused transverse growth to dominate over space-charge expansion and thus be more easily measured as well as creating a known initial circular cross-section to compare to the beamlet cross-section at the detector. Creating the beamlets also allows information from the central regions of the beam to be used in determining its properties. Figure 4.3 is a simplified schematic of the Gated Beam Imager.

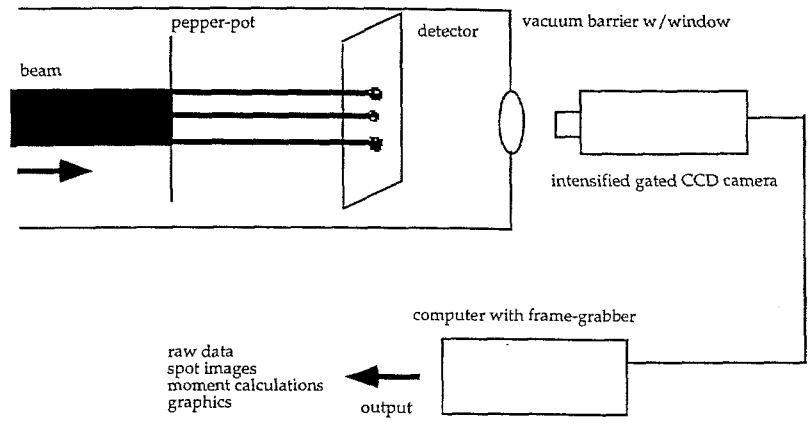


Figure 4.3: Simplified GBI instrument schematic

4.3 Design of the Gated Beam Imager

Several dimensions of the GBI such as pepperpot hole size, hole spacing, and beamlet drift length must be sized to achieve the goal of maximizing the beamlet's growth due to emittance while minimizing the growth from remaining space-charge forces. Additionally, adequate signal strength must be present at the CCD camera and the beamlet image spot size must be much greater than the camera system resolution so that statistical averages of the CCD pixels are valid. These requirements place constraints on the pepperpot hole size, hole spacing, and beamlet drift length. For instance, small beamlet size is desirable to limit space-charge forces but small beamlets reduce the detectable signal to the camera. Also, the hole spacing must be close enough to allow enough beamlets for sufficient statistical averaging but the spacing is limited by the requirement that the expanding beamlet image spots not overlap at the detector. The drift distance, as well, is limited by spot overlap due to the quadratic growth with distance of the beamlet from space-charge. Additionally, the drift distance must be sufficient to allow adequate beamlet spreading from emittance but prevent excessive space-charge spreading. An optimization of these competing parameters was done by developing a spread sheet to predict the performance of the GBI based on reasonable estimates of instrument dimensions and

constraints imposed by the Small Recirculator design. The results used in the design of the GBI are shown in Table 4.1. The criteria used to define an acceptable design were that beamlet spread due to space-charge be less than 10% and that no beamlet spots could overlap due to expansion. The spreadsheet also makes estimates of the amount of optical signal available to the CCD camera based on the input design parameters. A copy of the spreadsheet is contained in Appendix A and details of the calculations are available in Reference 55.

Table 4.1: Results of the GBI design optimizing spreadsheet.

Input Data			
Beam current	2 mA	Hole diameter	100 μ
Beam radius	0.5 cm	Hole spacing	2.5 mm
Beam energy	80 keV	Drift distance	15 cm
Beam length	4 μs	Beam emittance	10 mm-mr
Beamlet Calculations			
Max acceptance angle through hole			12.34°
Beamlet spread due to space charge			8.2%/8.2μ
Beamlet spread from emittance			214%/214μ
Beamlet diameter at detector			322.5μ
Distance between beamlets			1.96 mm

The optimization calculations assume that the mean direction of the beam ions is parallel with the recirculator axis (z). In general, the beam will be converging in one transverse plane while diverging in the other transverse plane through the FODO lattice. The random transverse thermal velocity of the ions will be superimposed on top of this directed transverse motion. While this directed transverse motion may not be overly significant to the design of the instrument, it is important to measure it in order to obtain an accurate calculation of the transverse emittance.

4.4 Pepperpot Method Data Analysis

A diverging beam appears in phase space as a rotation of the major axis of the phase space ellipse containing the beam ion distribution into the first and third quadrant. Figure 4.4 shows a schematic of a diverging beam's phase space ellipse and the contribution to it of two beamlets created by a pepperpot.

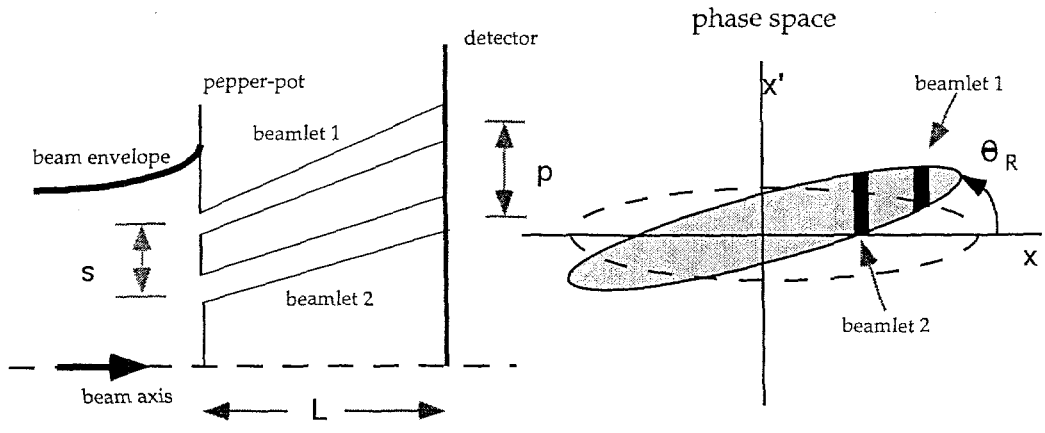


Figure 4.4: Determination of the phase space ellipse rotation angle.

Using some geometry and the assumption that the divergence angle is small yields the expression for the emittance ellipse rotation angle, "θ_R":

$$\theta_R = \arctan\left(\frac{p-s}{L}\right) \tag{4.1}$$

The hole spacing, s, and drift distance, L, are known and the determination of the beamlet spot spacing, p, requires a calibration of the image pixel size so the image analysis software can measure the distance between spots. To improve the accuracy of the measurement the orthogonal distances between all the beamlet spots can be measured and θ_R calculated from the average image spot separation.

A schematic of a single beamlet is shown in Figure 4.5. The beamlet divergence angle is calculated based on the pepperpot hole size, spot size, and drift distance. For

the nth beamlet (from a pepperpot array of N holes) the divergence angle is:

$$x'_n \equiv \frac{dx_n}{dz} = \frac{1}{L}(x_n - r_h) \tag{4.2}$$

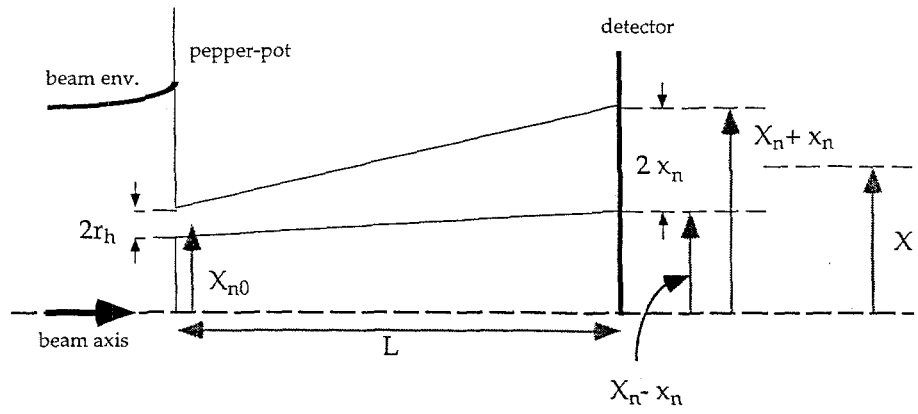


Figure 4.5: Beamlet data needed for beam moment and emittance calculations

As seen in Figure 4.6 each beamlet from the pepperpot contributes two points, (X_{n0}, X'_{n1}) and (X_{n0}, X'_{n2}) to defining the phase space ellipse. The pepperpot hole radius is assumed small compared to the beam radius. The maximum and minimum boundaries of the phase space ellipse at a given position is given by

$$X'_{n2} = \frac{(X_n + x_n) - X_{n0}}{L} \qquad X'_{n1} = \frac{(X_n - x_n) - X_{n0}}{L} \tag{4.3}$$

The points from all beamlets will roughly define the phase space ellipse. Figure 4.7 shows data plotted from a GBI measurement on the Small Recirculator where each set of two points derived from one beamlet is plotted with a vertical line connecting

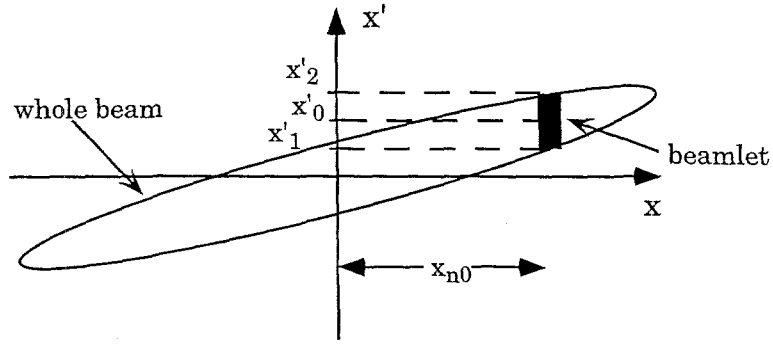


Figure 4.6: Beam phase space ellipse.

them. The area of the phase space ellipse is closely related to the beam emittance as discussed in Chapter 3.

The ellipse is generated using the first and second moments of the beam distribution. The moments of the entire beam are obtained by averaging the information obtained from each of the N total beamlets. For measurements made on the Small Recirculator beam, N≈70. The moments are calculated by discretizing the moment formula:

$$<\phi>=\iiiint \phi\, \rho(x,x',y,y')\, dx\, dx'\, dy\, dy' \tag{4.4}$$

with the normalization

$$\iint \rho(x,x',y,y')\, dx\, dx'\, dy\, dy' = 1 \tag{4.5}$$

therefore the explicit forms for the first and second moments are obtained from the beamlet spots by the following calculations:

$$< x >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x_n}{I_N} \qquad < x' >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x'_n}{I_N} \tag{4.6}$$

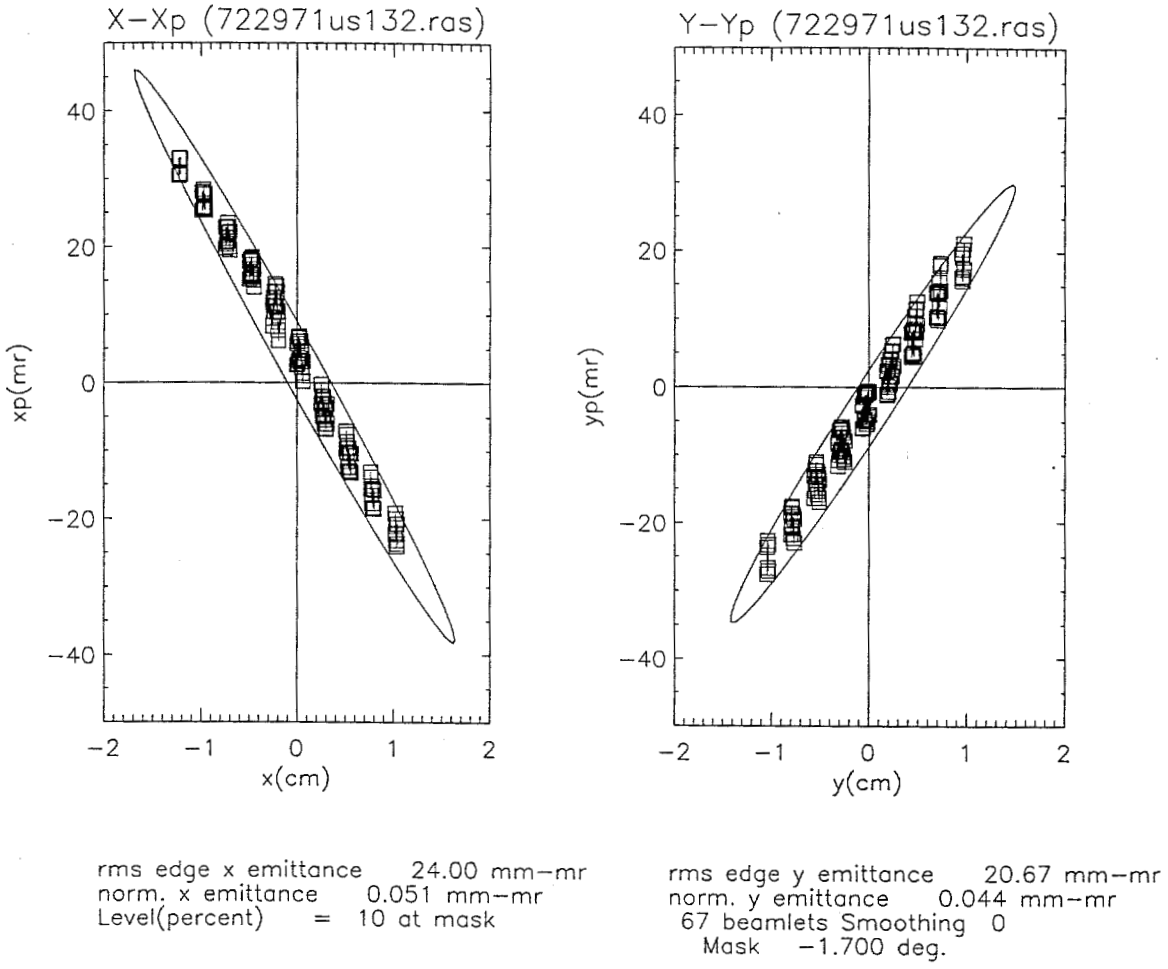


Figure 4.7: Phase space plot of a GBI measurement from the Small Recirculator. The ellipse rotation angles correspond to the FODO lattice focusing in the x-plane while defocusing in the y-plane at the GBI location.

$$< y >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n y_n}{I_N} \qquad < y' >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n y'_n}{I_N} \tag{4.7}$$

$$< x^2 >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x_n^2}{I_N} \qquad < y^2 >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n y_n^2}{I_N} \tag{4.8}$$

$$< xx' >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x_n x'_n}{I_N} \qquad < yy' >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n y_n y'_n}{I_N} \tag{4.9}$$

$$< x'^2 >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x'^2_n}{I_N} \qquad < y'^2 >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n y'^2_n}{I_N} \tag{4.10}$$

$$< xy >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x_n y_n}{I_N} \qquad < x'y' >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x'_n y'_n}{I_N} \tag{4.11}$$

$$< x'y >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x'_n y_n}{I_N} \qquad < xy' >= \frac{\frac{1}{N}\sum_{n=1}^{N} I_n x_n y'_n}{I_N} \tag{4.12}$$

$$I_n = \frac{1}{P_n}\sum_{p=1}^{P_n} V_p \qquad I_N = \frac{1}{N}\sum_{n=1}^{N} I_n \tag{4.13}$$

where N is the total number of beamlets, n is the beamlet index, P_n is the total number of pixels contained in the nth beamlet's digitized image, V is the digitized pixel value, p is the pixel index, I_n is the average intensity of the nth beamlet, and I_N is the average intensity of all N beamlets. In the above formulas for the second moments the contribution from non-zero first moments (beam centroid not on the accelerator axis) is not removed. For a beam not on axis in phase space the following calculation is used to remove this contribution:

$$\Delta x^2 =< x^2 > - < x >^2 \qquad \Delta y^2 =< y^2 > - < y >^2 \tag{4.14}$$

$$\Delta x'^2 =< x'^2 > - < x' >^2 \qquad \Delta y'^2 =< y'^2 > - < y' >^2 \tag{4.15}$$

$$\Delta xx' =< xx' > - < x >< x' > \qquad \Delta yy' =< yy' > - < y >< y' > \tag{4.16}$$

$$\Delta xy =< xy > - < x >< y > \qquad \Delta x'y' =< x'y' > - < x' >< y' > \tag{4.17}$$

$$\Delta x'y =< x'y > - < x' >< y > \qquad \Delta xy' =< xy' > - < x >< y' > \tag{4.18}$$

where the Δ symbol introduced by Barnard will be used henceforth to indicate a second moment with the centroid contribution removed.[29] The equation for the ellipse fitted to the pepperpot data is

$$\gamma x^2 + 2\alpha xx' + \beta x'^2 = \epsilon_x \tag{4.19}$$

where

$$\gamma = \frac{\Delta x^2}{\epsilon_x} \quad \alpha = \frac{-\Delta x x'}{\epsilon_x} \quad \beta = \frac{\Delta x'^2}{\epsilon_x} \tag{4.20}$$

The transverse rms edge emittance, ϵ_x, is obtained from

$$\epsilon_x = 4\left(\Delta x^2 \Delta x'^2 - (\Delta x x')^2\right)^{\frac{1}{2}} \tag{4.21}$$

The previous discussion of measuring second moments by measuring the half-widths of the beamlet spots assumed the spots were of regular shape (ellipses or circles) and the "edges" of the spots identifiable. The beamlets originate with round cross-section at the pepperpot but become elliptical due to different focusing in the x and y directions from the FODO lattice prior to the pepperpot, which distorts them into ellipse-like shapes at the detector.

The raw data available from the GBI consists of a two-dimensional array of digitized pixel values in the field of view of the CCD camera observing the detector. Pixel values range in value from 0 to 65535 (with 16-bit digitization of the CCD well). Generally, a contour of equal pixel value identified as the beamlet image edge will not be an ellipse. To make a spot size measurement, an ellipse is fitted to the CCD display intensity contour of interest and the measurement of major and minor axes based on this.

Practically this can be done by sampling the contour of interest (most likely the contour whose value is 10% of the spot's peak pixel value), fitting an ellipse and then rotating the ellipse to its upright position such that the major and minor axes lie on the x and y axes. This is done as follows: A thin band of N points (x_j, y_j) at the 10% contour level is chosen as in Figure 4.8 on page 48. The coefficients for an elliptical approximation to this band are calculated from:

$$\bar{x^2} = \frac{2}{N}\sum_{n=1}^{N} {x_j}^2 \quad \bar{y^2} = \frac{2}{N}\sum_{n=1}^{N} y_j^2 \quad \bar{xy} = \frac{2}{N}\sum_{n=1}^{N} x_j y_j \tag{4.22}$$

The centroid (x_c, y_c) of the ellipse is given by:

$$(x_c, y_c) = (\frac{1}{N}\sum_{n=1}^{N} x_j \quad \frac{1}{N}\sum_{n=1}^{N} y_j) \tag{4.23}$$

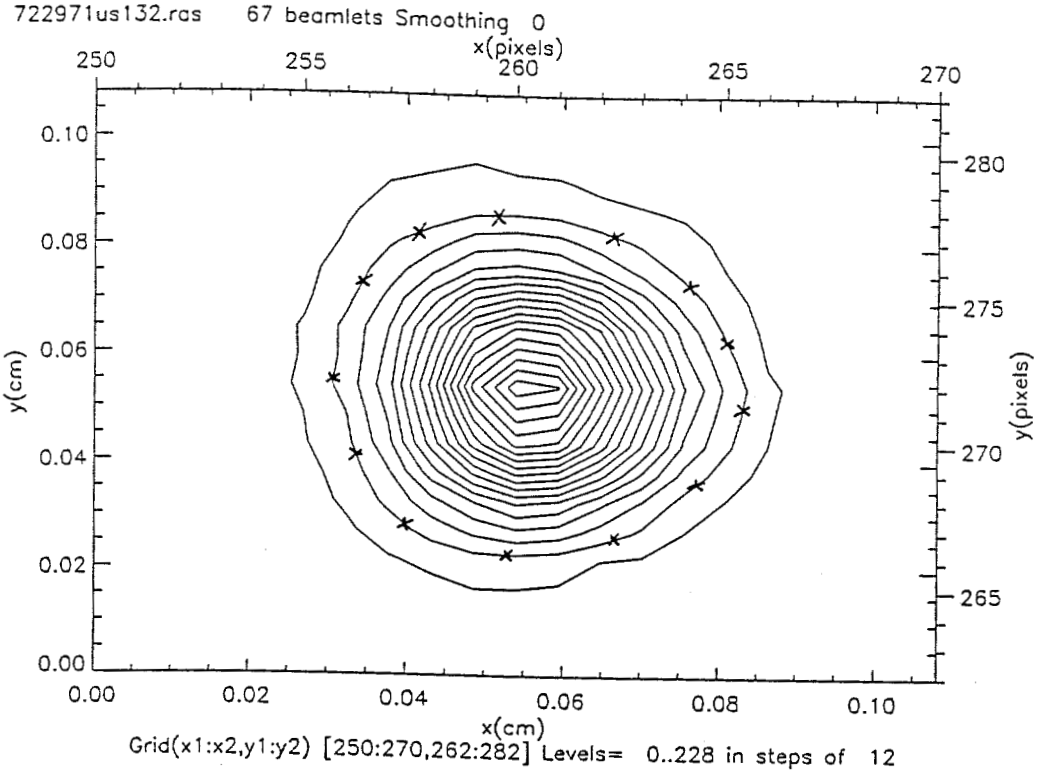


Figure 4.8: Contour of a single beamlet from a GBI measurement

These averages, $(\bar{x^2}, \bar{y^2}, \bar{xy})$ can be equated to the elements of an ellipse's sigma matrix. The equation of the ellipse is:

$$\gamma x^2 + 2\alpha xy + \beta y^2 = 1 \tag{4.24}$$

and

$$\sigma = \begin{bmatrix} \sigma_{11} & \sigma_{21} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} = \begin{bmatrix} \beta & -\alpha \\ \alpha & \gamma \end{bmatrix} = \begin{bmatrix} \bar{x^2} & \bar{xy} \\ \bar{yx} & \bar{y^2} \end{bmatrix} \tag{4.25}$$

thus,

$$\sigma_{11} = \bar{x^2} \qquad \sigma_{22} = \bar{y^2} \qquad \sigma_{12} = \bar{xy} = \sigma_{21} = \bar{yx} \tag{4.26}$$

The general rule for transforming the sigma matrix (e.g. a pure rotation) is

$$\sigma^* = R\ \sigma\ R^T \tag{4.27}$$

The transformation for a general rotation through angle, θ is

$$\sigma^* = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \sigma_{11} & \sigma_{21} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{4.28}$$

$$\sigma^* = \begin{bmatrix} \cos^2\theta\sigma_{11} + 2\sin\theta\cos\theta\sigma_{12} + \sin^2\theta\sigma_{22} & -\sin\theta\cos\theta\sigma_{11} + (\cos^2\theta - \sin^2\theta)\sigma_{12} + \sin\theta\cos\theta\sigma_{22} \\ -\sin\theta\cos\theta\sigma_{11} + (\cos^2\theta - \sin^2\theta)\sigma_{12} + \sin\theta\cos\theta\sigma_{22} & \sin^2\theta\sigma_{11} + 2\sin\theta\cos\theta\sigma_{12} + \cos^2\theta\sigma_{22} \end{bmatrix} \tag{4.29}$$

To find the major and minor widths of the beamlet the required transformation is one that rotates the ellipse into its upright coordinate system. The correct rotation angle that puts the ellipse upright can be solved from:

$$\sigma^*_{12} = \sigma^*_{21} = 0 \qquad => \qquad \theta = \frac{1}{2}\arctan\left(\frac{2\sigma_{12}}{\sigma_{11} - \sigma_{22}}\right) \tag{4.30}$$

therefore the upright beamlet spot ellipse sigma matrix is:

$$\sigma^* = \begin{bmatrix} A^2 & 0 \\ 0 & B^2 \end{bmatrix} \tag{4.31}$$

with major and minor axes of the fitted ellipse given by

$$A = (\cos^2\theta\sigma_{11} + 2\sin\theta\cos\theta\sigma_{12} + \sin^2\theta\sigma_{22})^{1/2} \tag{4.32}$$

$$B = (\sin^2\theta\sigma_{11} + 2\sin\theta\cos\theta\sigma_{12} + \cos^2\theta\sigma_{22})^{1/2} \tag{4.33}$$

These values (A, B) are the widths of the beamlet spots used in the calculation of the ion divergence angles and the ion distribution's first and second moments.

4.5 Image Capture and Image Analysis Code

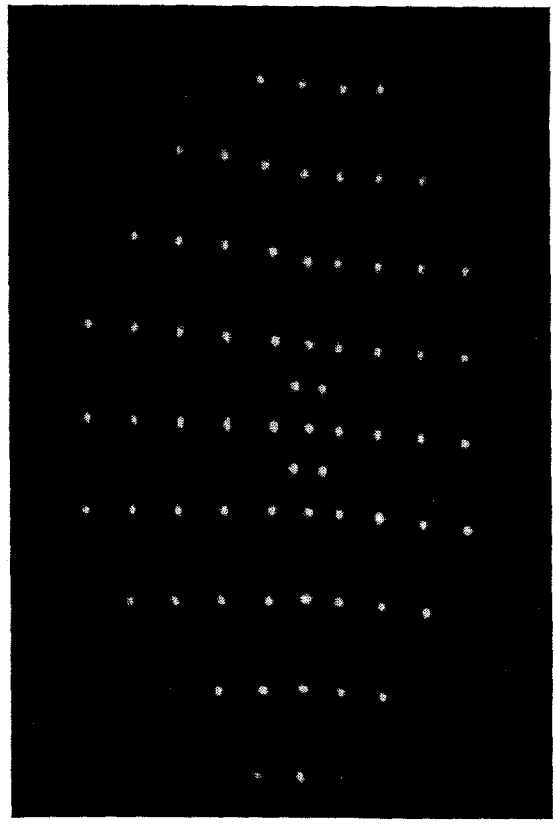
To accomplish the immense task of data handling involved in capturing high resolution images, identifying and measuring beamlet spot sizes and locations, and performing the data analysis described above I used a combination of a commercial image capture software and a modified version of a pepperpot data analysis code developed at LLNL.[28] The image capture software, *IPLab Spectrum*^{TM} by Signal Analytics Corporation, running on a Macintosh computer incorporates an extension allowing it to control the CCD camera and store images into the computer memory.[32] It also allows other operations to be done such as focusing the camera and determining the field of view. Certain image manipulations were also accomplished with this software such as measuring distances between known points for pixel size calibration, animating a series of images of the beam into a "beam movie", an electronic version of a flip book of sequential images, and correcting images prior to data analyses. The mechanics of acquiring and correcting data images is discussed in Chapter 5.

The pepperpot data analysis code, "PEPPERPOT" was written for analysis of pepperpot images generated by a pepperpot with holes arranged in concentric circles. It is written in Fortran-like commands of the *Interactive Data Language* (IDL) developed by Research Systems, Inc.[33] I modified the source code to optimize its use with the GBI on the Small Recirculator. Changes included the ability to do calculations based on a rectangular array of beamlets. The rectangular array is more appropriate

for diagnosing a beam with two orthogonal planes of motion produced by the FODO lattice. The original application was for solenoidal transport which produces radial symmetry, hence the circular pepperpot design. Other modifications I made included the ability to analyze different size data arrays to accommodate different CCD arrays, calculation of the x-y cross moments, and I added in the algorithm for weighting each beamlet as prescribed earlier in this chapter. This last item was done to bring the calculation in line with the method by which the data from the slit scanner diagnostics is analyzed so that direct comparisons are possible. I also added a module to the source code that fits a gaussian to a cross-section of a beamlet so fitted beamlet size data could be used for moment calculations if desired. To gather time-sequenced data as a narrow gate is stepped through successive beam shots, moment and emittance data is written to files for later display using other graphics routines as shown in Chapter 7. The Pepperpot code allows interactive analysis of the GBI images with graphical and tabular output. A copy of the PEPPERPOT IDL source code is contained in Appendix E.

4.6 Sample Output from the Data Analysis Code

The following figures are samples of the output of various analyses on an image.



722971us132.ras
integrated beam

(67 beamlets Smoothing 0)

Figure 4.9: GBI image used for analysis of beam moments. The four spots between rows surrounding the center spot allow easy identification of the center hole as a reference and are not used in the data analysis. Each spot is approximately 0.5 mm in diameter and the field-of-view is 2.07 by 3.11 cm.

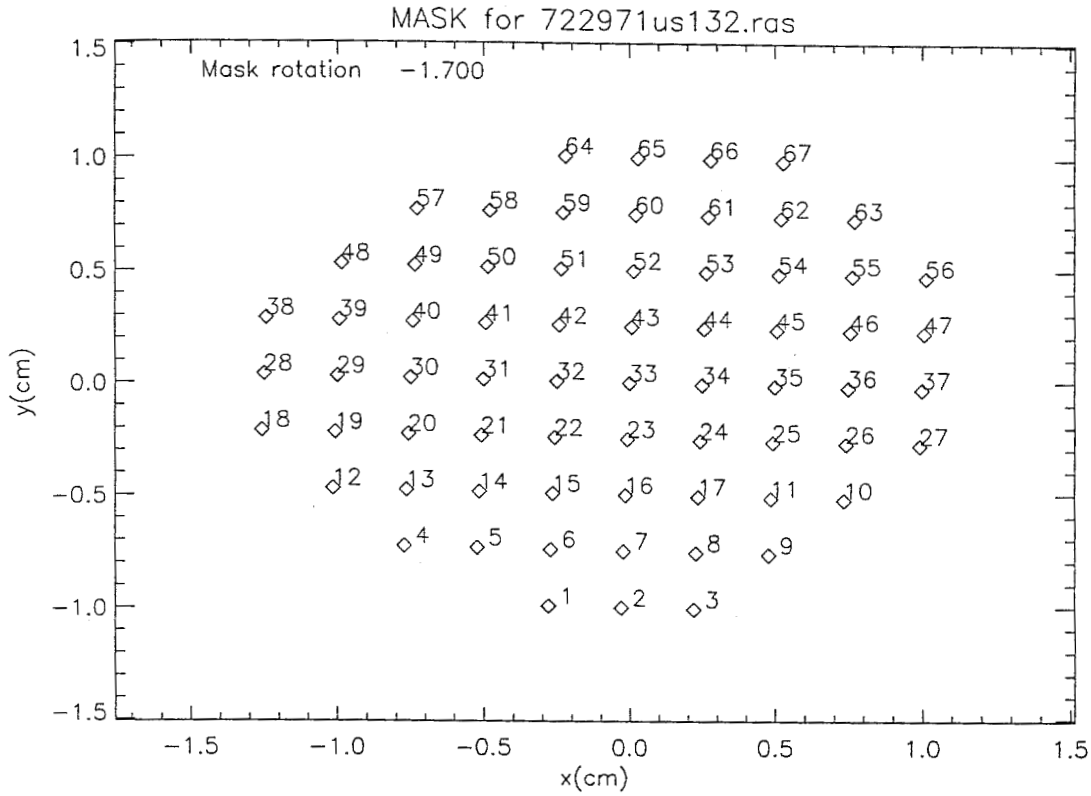


Figure 4.10: Identification of which holes the beamlets in Figure 4.9 originated from.

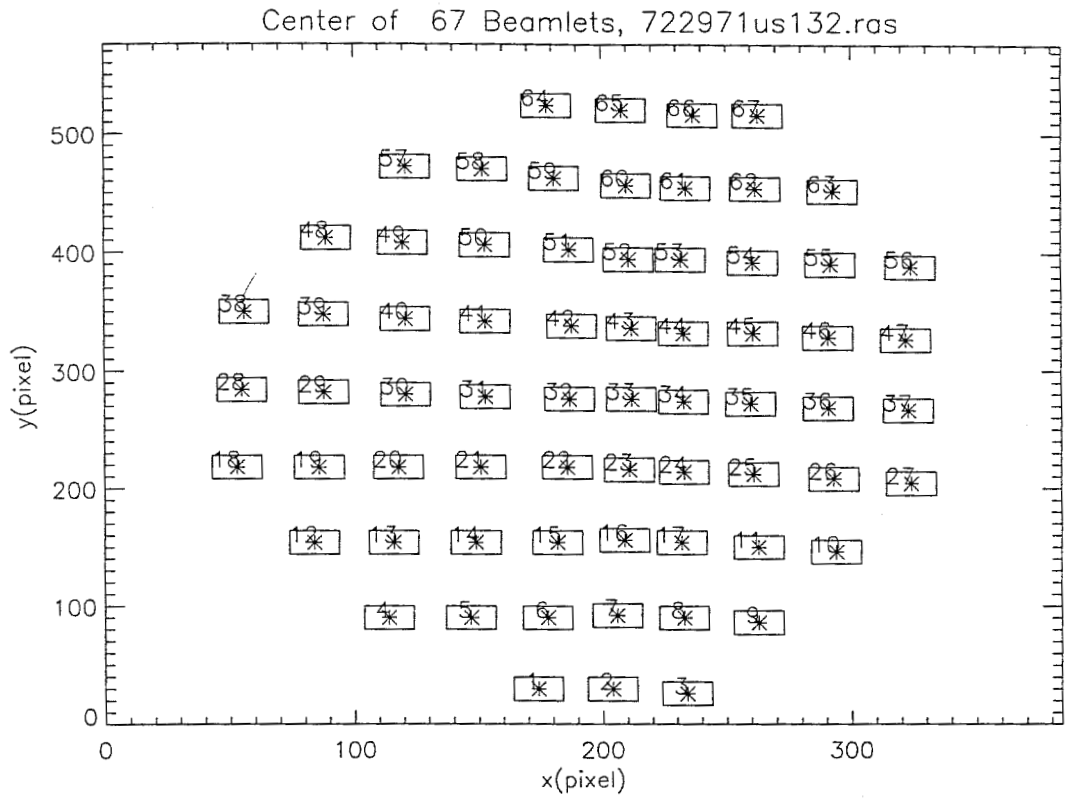


Figure 4.11: Plot of the data analysis windows surrounding the holes through which the beamlets passed.

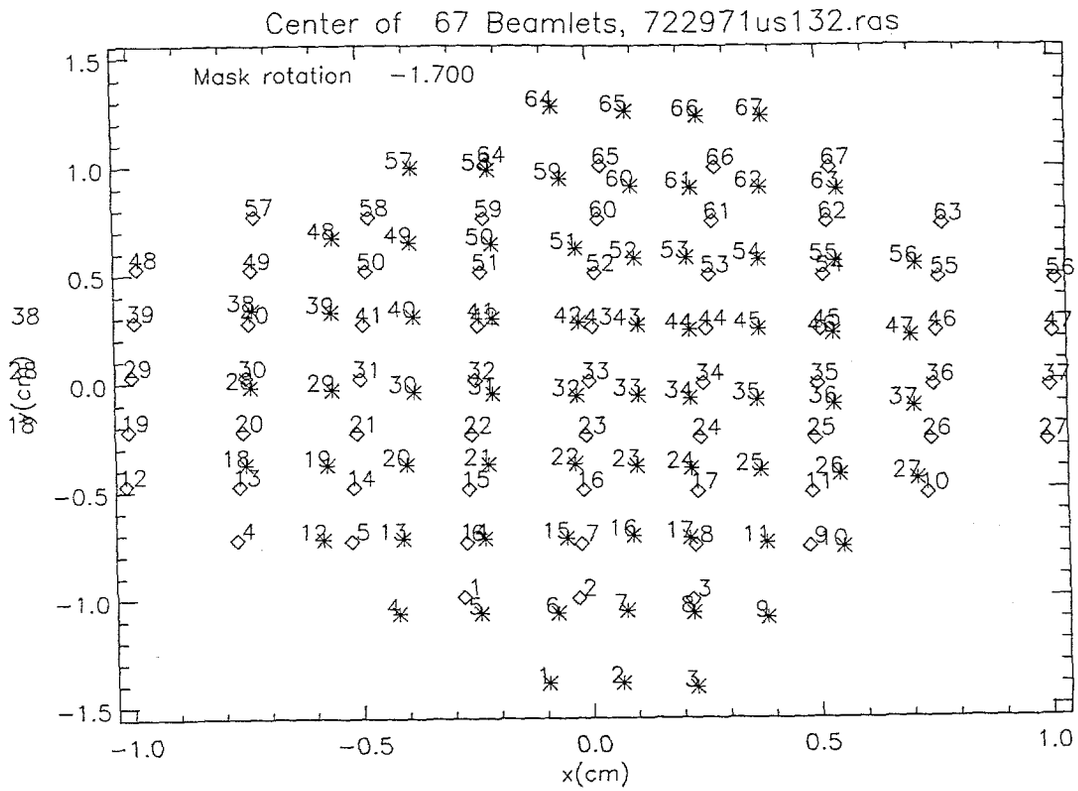


Figure 4.12: Plot superimposing the pepperpot holes (diamond symbols) and the centroids of the beamlet spots (asterisks) at the detector showing focusing in the x-plane and defocusing in the y-plane.

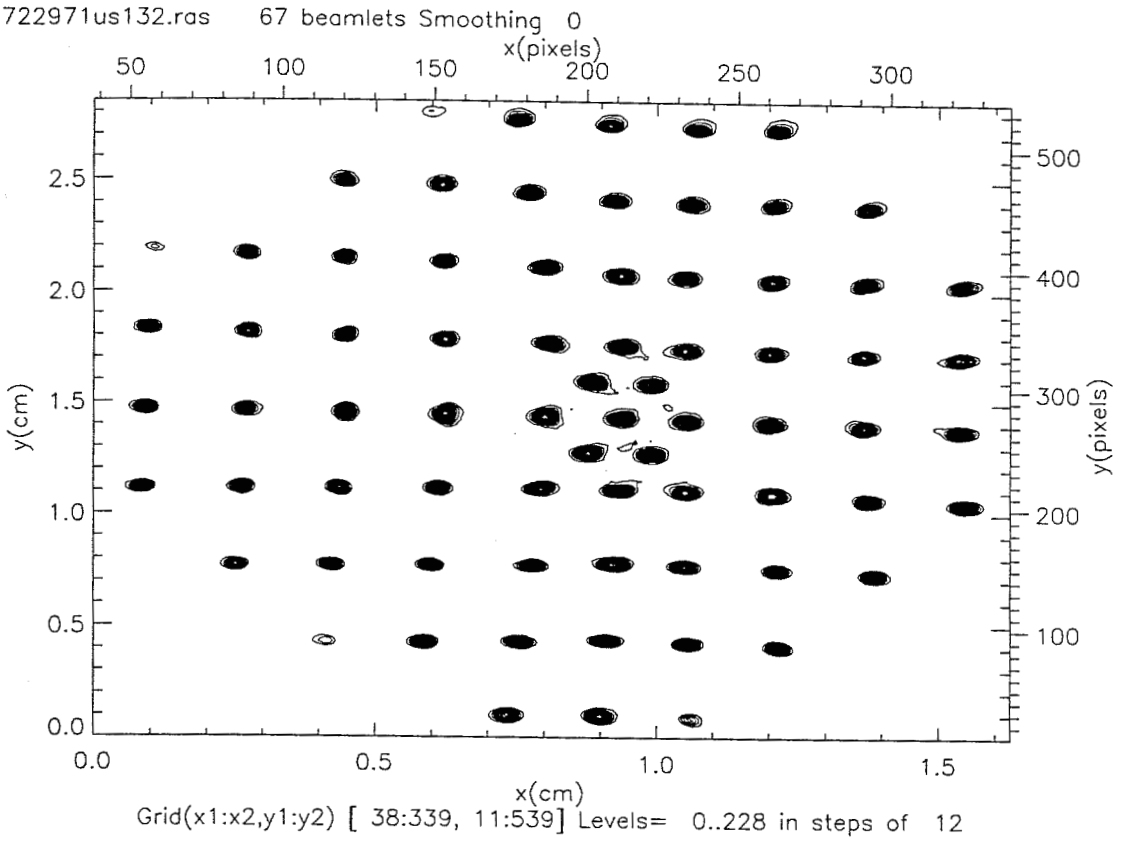


Figure 4.13: Contour plot of all data from the CCD camera field of view.

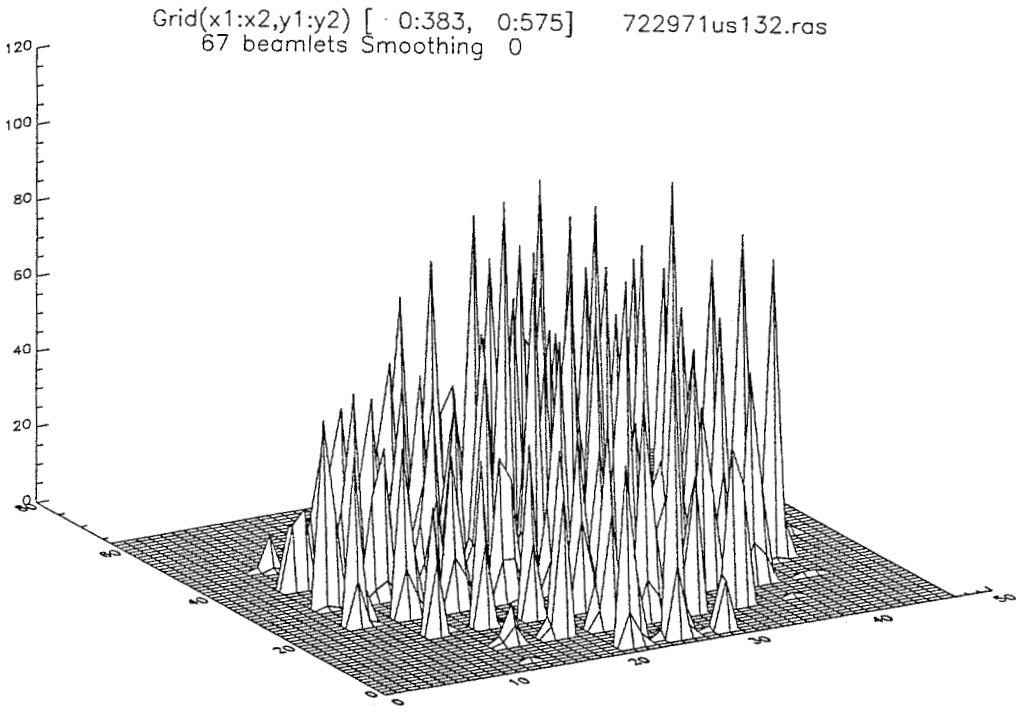


Figure 4.14: Surface plot of the intensity vs. position for the CCD camera field of view.

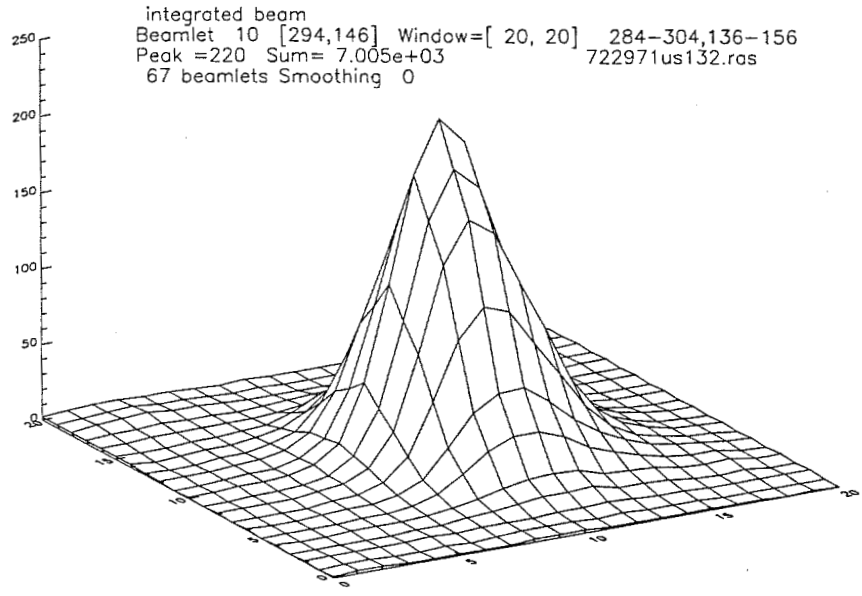


Figure 4.15: Surface plot of beamlet intensity vs. position limited to a 20 by 20 pixel data analysis window around one beamlet.

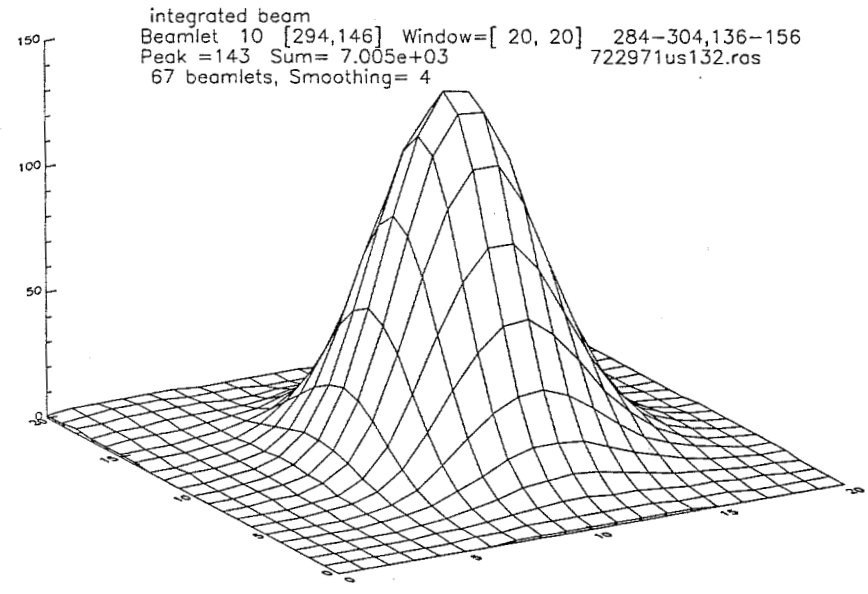
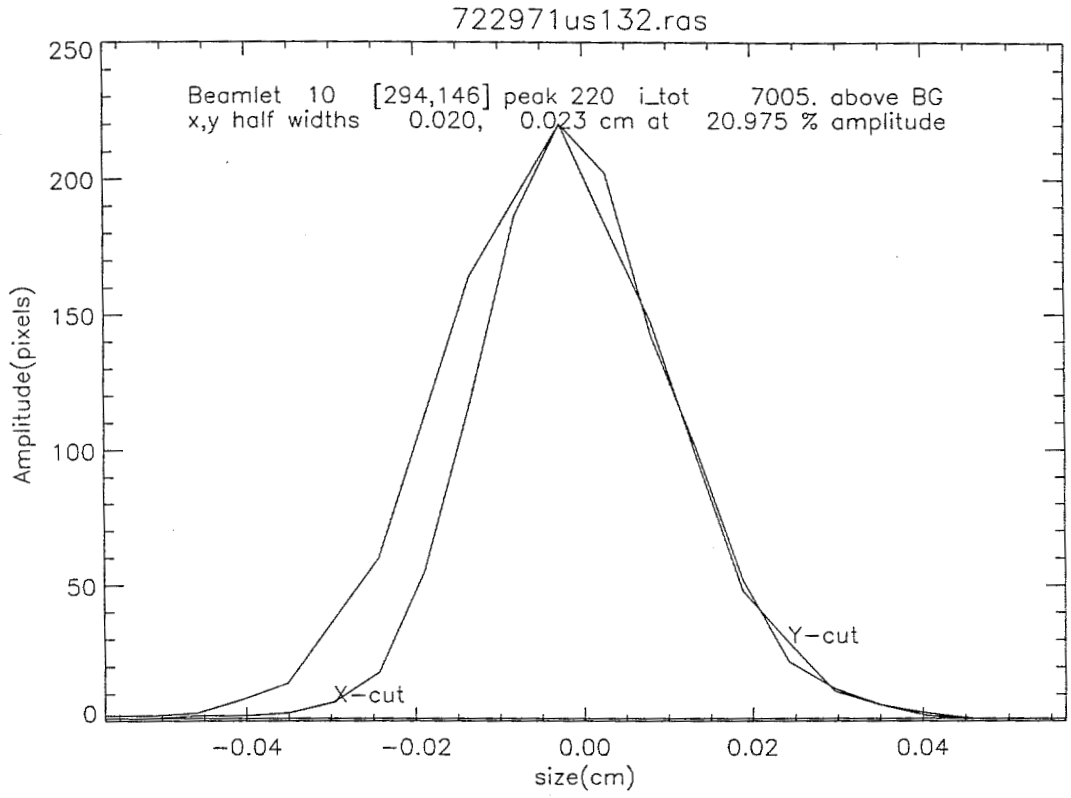


Figure 4.16: Surface plot of the beamlet of Figure 4.15 after a 4 by 4 pixel block averaging of the intensity values.



x,y half widths 0.008, 0.010 cm at 73.541 % amplitude
x,y half widths 0.024, 0.030 cm at 9.695 % amplitude
x,y half widths 0.020, 0.023 cm at 20.975 % amplitude

Figure 4.17: Beamlet cross-section plots of intensity vs. position. The width at different intensity levels can be determined by positioning a cursor.

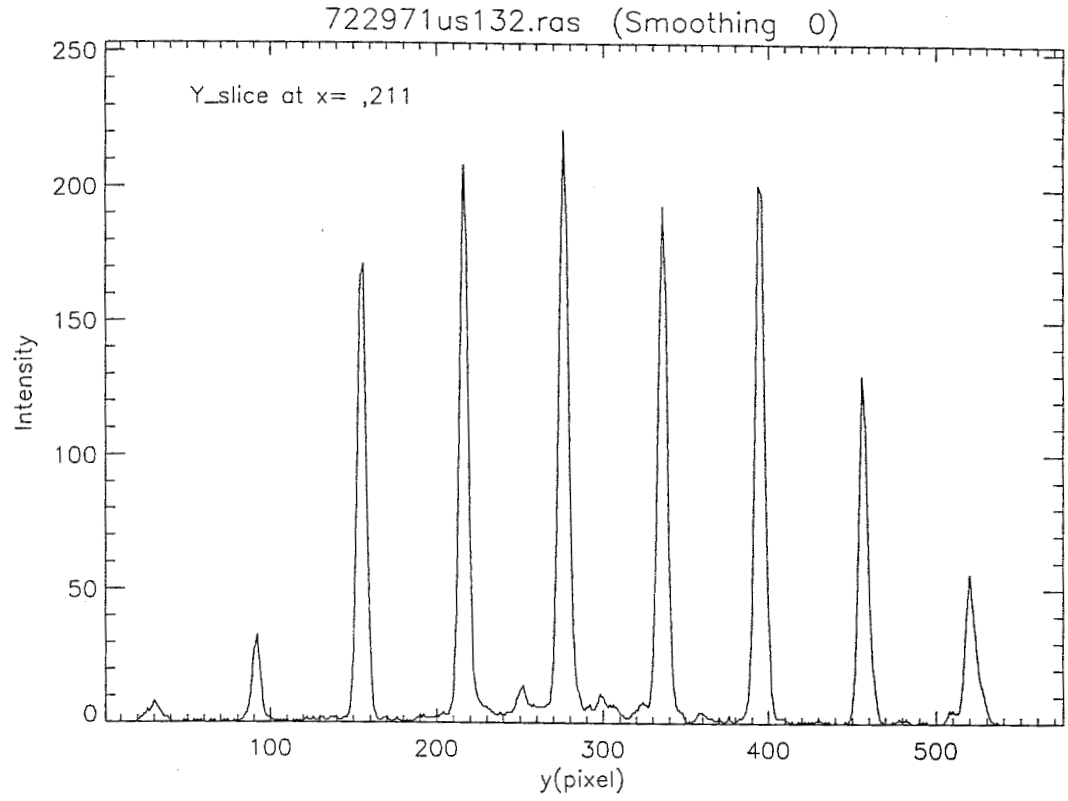


Figure 4.18: Plot of a cut through the image at a selected row or column. Here column 211 of Figure 4.9 is displayed.

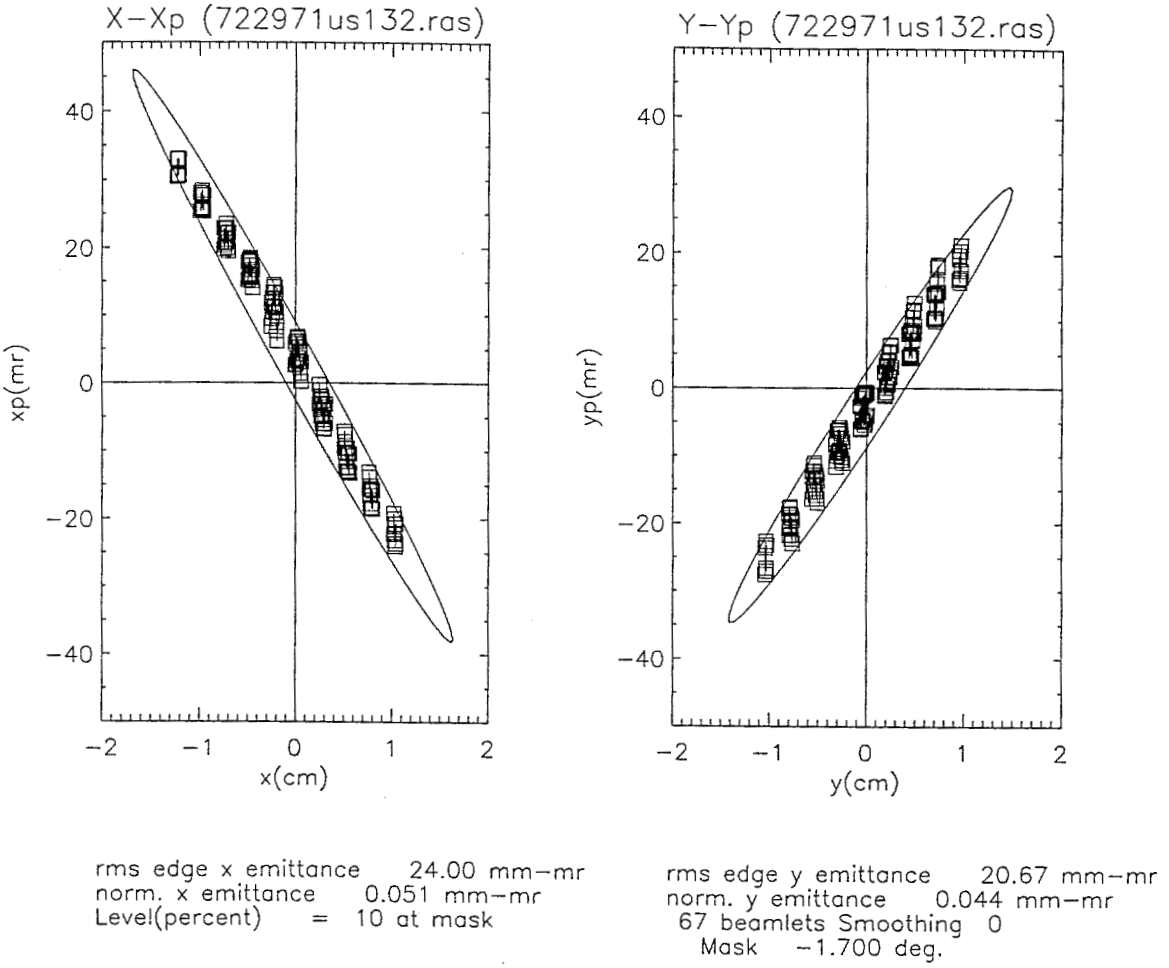


Figure 4.19: 2-D phase space plots with individual beamlet data fitted with an ellipse and emittance calculations displayed.

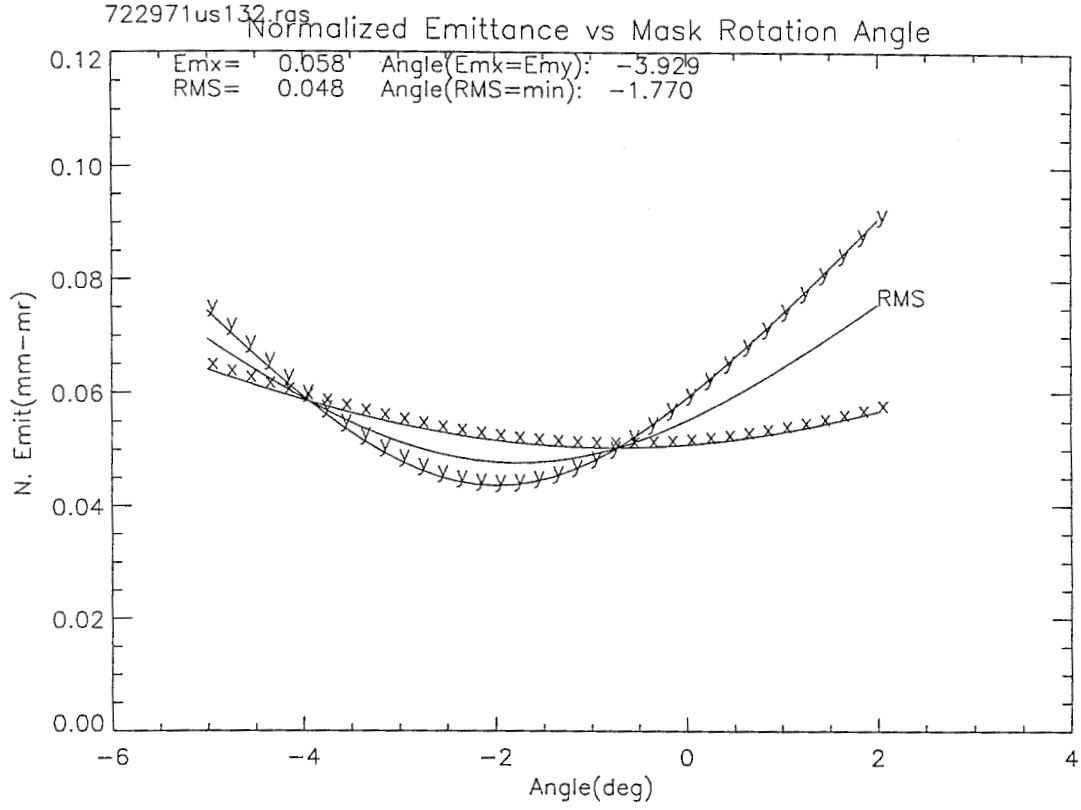


Figure 4.20: Plot of normalized emittance vs. pepperpot holeplate rotation angle.

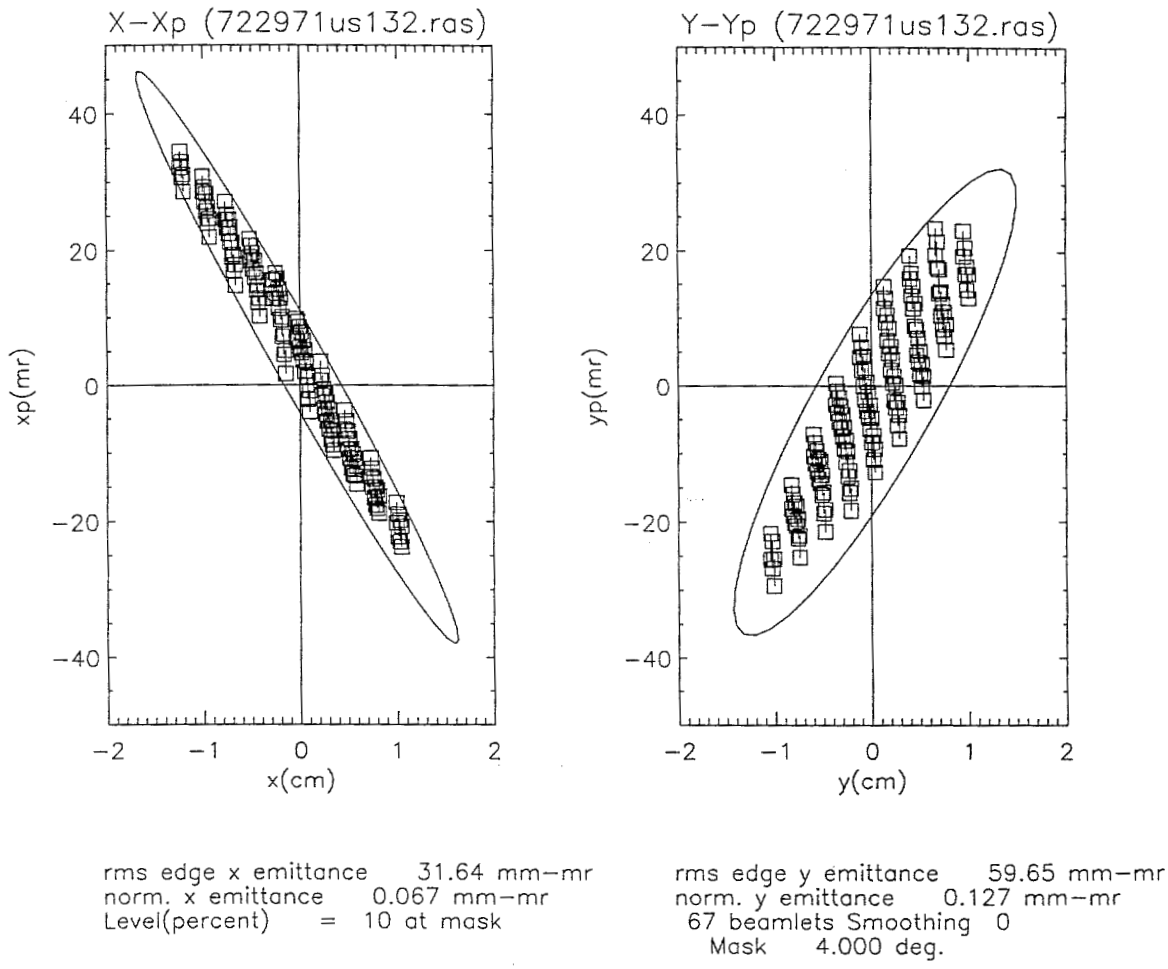


Figure 4.21: The appearance of 2-D phase space plots when an incorrect holeplate rotation angle is used in calculating x' and y'. Erroneous beamlet divergence angles are calculated resulting in a "vertical" separation of beamlet data and and a larger emittance value.

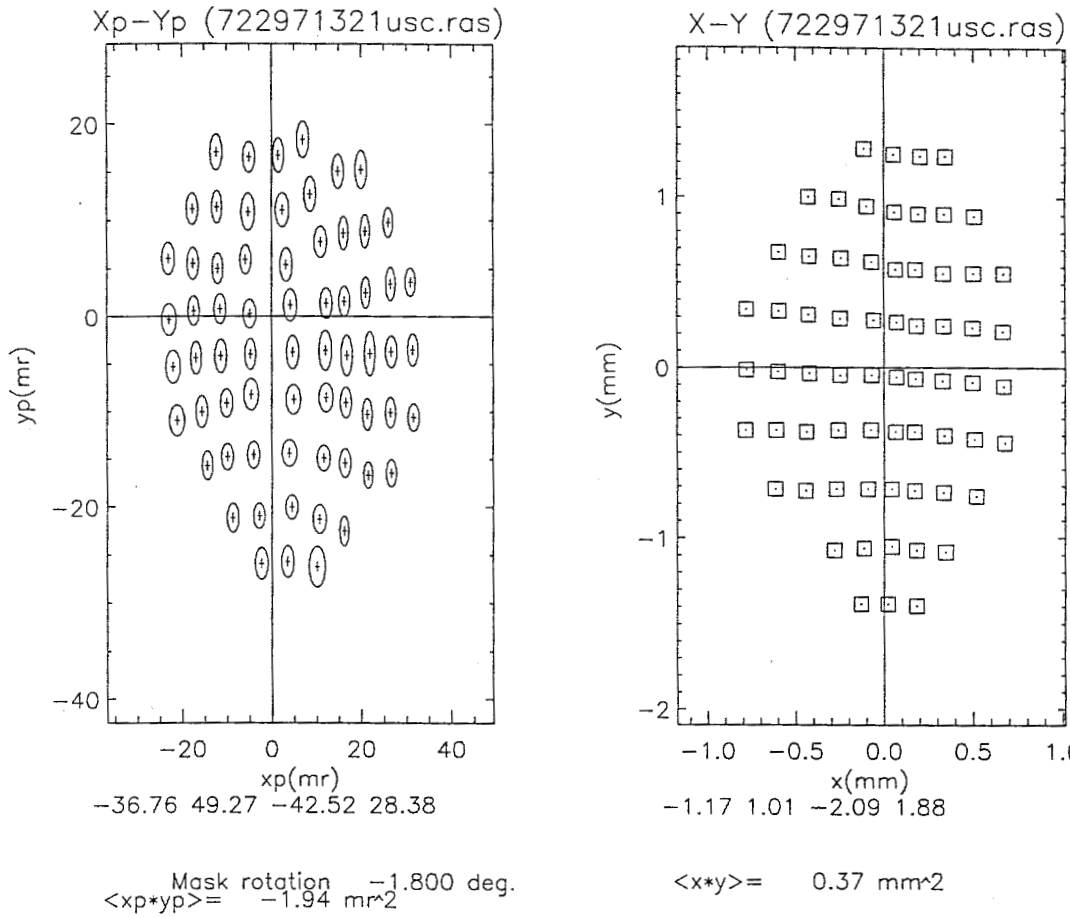


Figure 4.22: Plot of $x-y$ cross moments phase space with $<x'-y'>$ and $<x-y>$ shown. The magnitude of the axes of the $x'-y'$ ellipses are determined by the minimum and maximum divergence angles of each beamlet spot.

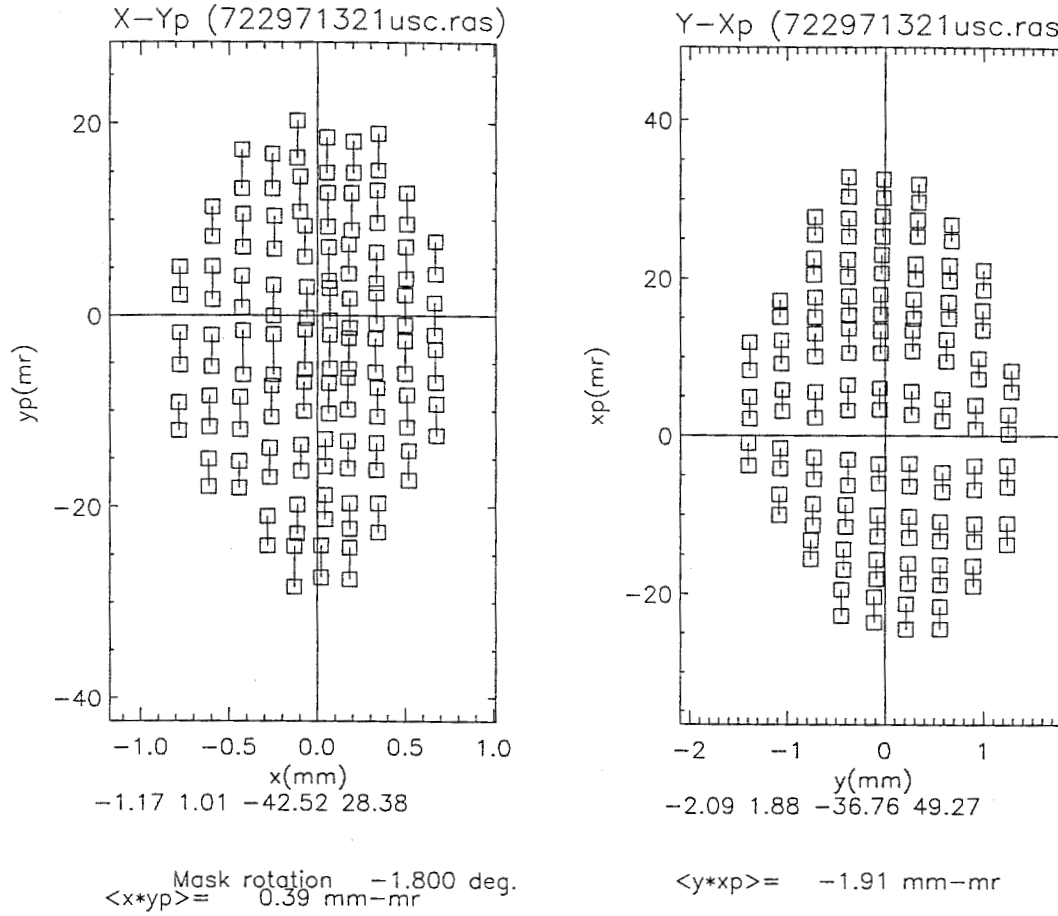
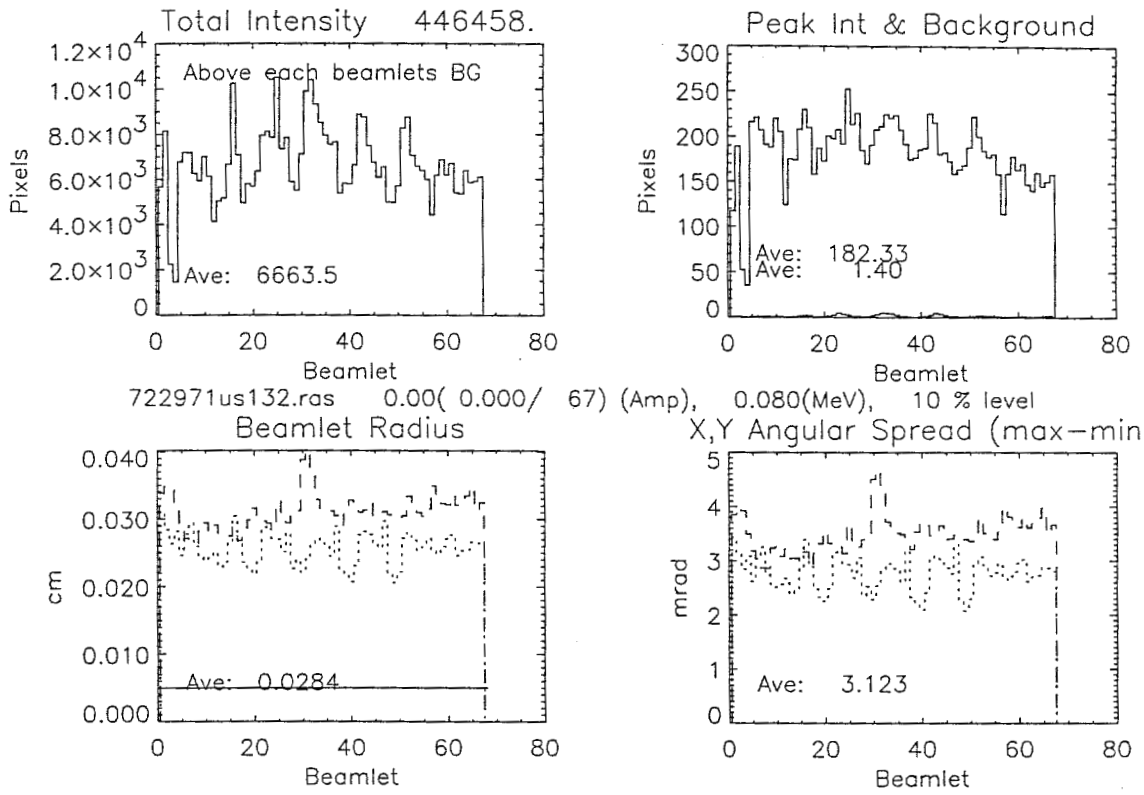


Figure 4.23: Plot of $x-y$ cross moments phase space with $<x'-y>$ and $<x-y'>$ shown.




```
0 "*.RAS" File:        =722971us132.ras (temp.dat)     VERSION(5/19/97)
1 Image(nx,ny)         =[384,576]  67 beamlets Smoothing  0
2 Imin,Imax,Backg      =[  0,253]   0
3 X,Y (cm/pixel)       =[0.00540,0.00540]
4 Energy(MeV),betag =[  0.0800,  0.0021]        36262.199
5 Window(pixels)       =[ 20, 20]
6 R_Hole,Spacing(cm)=[  0.0050,  0.2500,  0.0152] N
7 Drift(meters)        =  0.1500
8 Emittance(mm−mr)  =[   0.000,   0.000]   ave:   0.000   norm:   0.0000
9 Current,Brightness=[     0.00( 0.000/  67),  4.55e+08 +−2.01e+08]
10 Level(percent)      =  10
11 xr(cm),r12,rp(mr) =[ 0.8326,−0.9906,  21.07]      0.391 m
12 yr(cm),r12,rp(mr) =[ 0.7283, 0.9846,  16.22]     −0.442 m
13 J(x),(y),(xy),(av)=[  1.68e+23,  2.26e+23,  1.00e+12,  1.21e+11]
14 R(cm),RP(mr),Ipass=[   0.780,  1.562,  0.000]
19 Mask Rotation(deg)=  −1.700
```

Figure 4.24: Plots of beamlet peak and total intensity statistics.

Chapter 5

Detector Selection and Equipment Description

The selection of a specific material or device to serve as the beamlet detector in the GBI was done by considering the desired performance criteria. These criteria were:

1. Detector light output to be a linear function of incident beamlet intensity.
2. Shot-to-shot repeatability.
3. Fast detector response time to allow gating the camera.
4. Simplicity in construction.

5.1 Fast Plastic Scintillator as a Heavy Ion Detector

Due to the heavy ion's short range, its energy is deposited in a very small volume of material.[34] Phosphors are known to damage easily necessitating their frequent replacement. They also have generally long decay constants on the order of 300 μs, making them unsuitable for gating the camera at the desired interval of 100 ns and thus were not considered. The initial promising material considered was commercially available fast plastic scintillator. I conducted experimental studies on fast plastic scintillator, specifically BICRON Corporation BC-412. Its important characteristics are summarized in Table 5.1.

BC-412 was chosen based on its short decay constant, and good mechanical properties.

Table 5.1: BC-412 scintillator properties[52]

Light Output % Anthracene	60
Wavelength of Maximum Emission	434 nm
Decay Constant	3.3 ns
Bulk Light Attenuation	400 cm

It was unknown whether its light output would be sufficient for small, nanosecond gated signals or if its molecular structure responsible for scintillation would be rapidly damaged from intercepting the heavy ions. Initially its response to the 80 keV potassium ion beam was characterized by placing it in the Small Recirculator beam and measuring its light output signal with a photomultiplier tube. The output signal was observed to be proportional to, and have the same features as the ion source pulser voltage waveform and Faraday cup traces. The light output closely tracked the signal produced by the ion current into the Faraday cup.

Encouraged by the initial photomultiplier tube results, I then conducted experiments to photograph the image of the beam on the scintillator with increasing ion exposure. I exposed the scintillator to repeated 4 μs long, 80 keV pulses (368 shots) and periodically examined the image of the beam on the scintillator with a CCD camera. The images showed a rapid degradation in image quality. A plot of the sum of the CCD camera pixel intensity values in the area of the beam image versus exposure is shown in Figure 5.1. It can be seen that the light output in the area of the beam exhibits a sharp drop in level then a slower degradation. The images are shown in Figures 5.2(a)-(d). The previous photomultiplier tube measurements had not shown such a decrease in light output. It appeared from the full CCD images that there had been a lot of light reflected off the interior of the chamber that the photomultiplier tube recorded although the image in the area of interest on the scintillator degraded significantly. This reflected light is not shown in the sequence of Figure 5.2. I conducted several tests to determine whether the surface of the scintillator was charging up and deflecting ions from their trajectories by attaching an electrically grounded, fine mesh screen with an ion transmittance of 90% to the scintillator surface. No

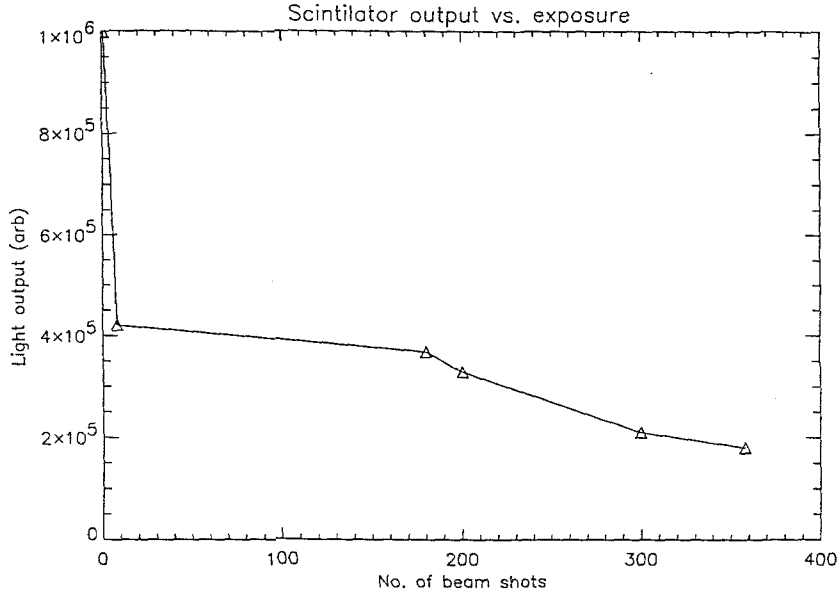


Figure 5.1: BC-412 scintillator light output vs. exposure.

significant effect on preventing image degradation was observed. Various techniques of scintillator surface preparation were also attempted to ensure a clean surface was exposed to the beam. Visual inspection of the scintillator after exposure showed no discernible changes such as clouding or yellowing of the material that would affect its light transmission. Scintillator once exposed also exhibited no recovery after not being exposed for up to a week afterwards and then tested again. Confirmation that damage was occurring in the localized area where beam ions interacted was tested by steering the beam to unexposed portions of the scintillator after at least 357 beam pulses. From Figure 5.2(d), it can be seen that the light output increases for that portion of the scintillator not previously exposed to the beam as the beam is steered to the right at the detector on shot 358. Various schemes of introducing unexposed scintillator into the beam for each shot were considered such as a rotating wheel of scintillator or "window shade" roller assembly. Although this was a complication in the GBI design it appeared to be feasible.

The remaining question was whether sufficient optical signal would be present

with a pepperpot introduced into the beam. This would cut the number of ions on the scintillator by roughly a factor of 40,000 and gating the CCD camera at a 100 ns interval would capture about 2.5% of that signal. I performed an experiment to image 4 μs long beamlets produced by a pepperpot with 400 micron diameter holes on unexposed scintillator. The images showed that not enough signal was produced to give confidence that accurate beamlet spot identification and analysis could be done. The ultimate pepperpot design was to have 100 micron diameter holes and to apply a 100 ns gate to the beam signal. As a result of the complications of scintillator damage and low light output in response to heavy ions, it was determined that fast plastic scintillator would be unsuitable as the detector in the GBI. Attention was now given to using a microchannel plate detector with the obvious sacrifice in simplicity and cost.

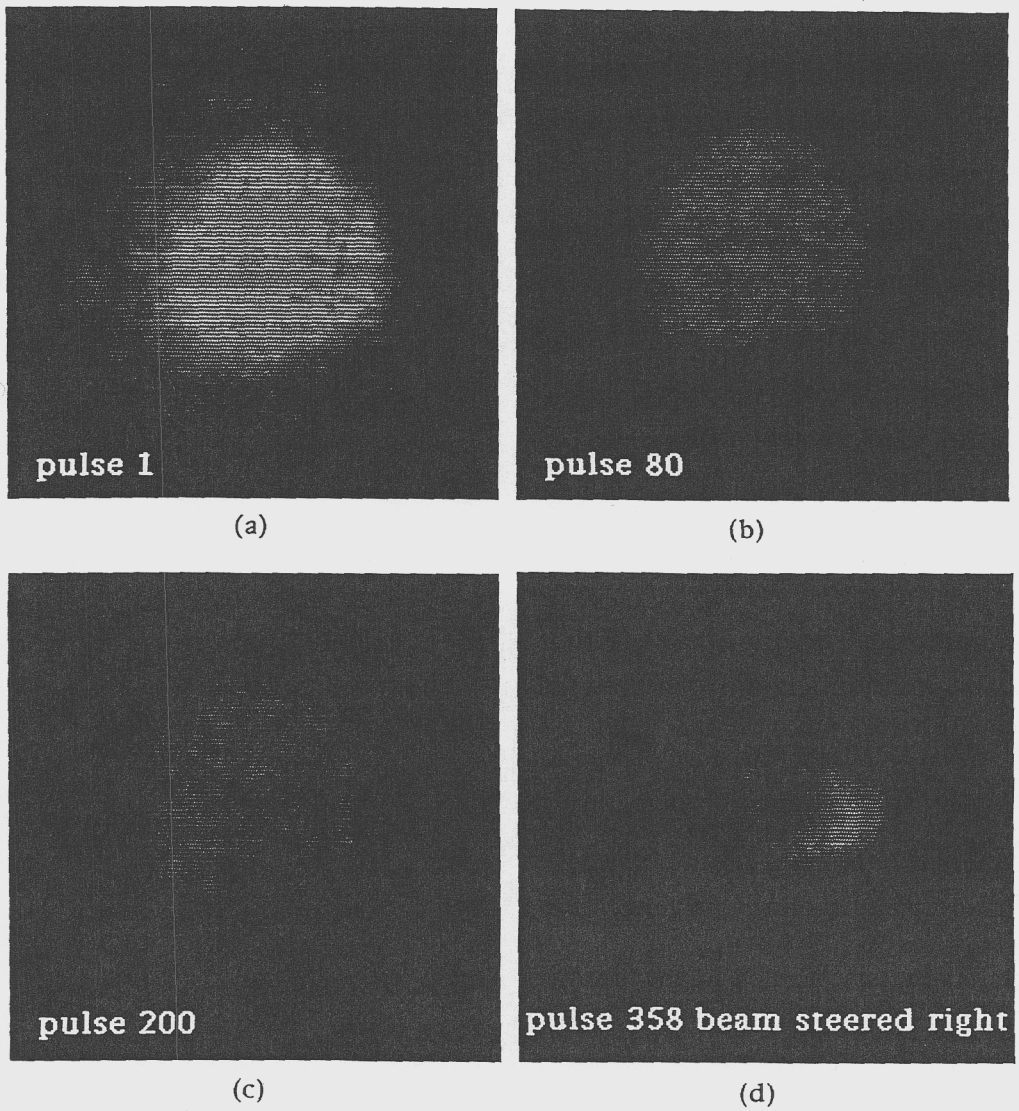


Figure 5.2: Sequence of whole beam images showing scintillator degradation with exposure. **(a)** First shot onto unexposed scintillator **(b-c)** Scintillator exposed to 80 and 200 4 μs shots **(d)** Beam steered slightly right onto unexposed scintillator after 357 shots.

5.2 Microchannel Plate Detector as a Heavy Ion Detector

The response of phosphors and scintillators to intense electron beams is known to be satisfactory from the standpoint of using them as reliable and accurate particle

detectors.[35] The stopping power is much less for electrons than for heavy ions resulting in less energy deposited per unit volume and more interactions giving rise to photons in the visible spectrum. A device that converts the heavy ions in the pepperpot beamlets to an electron signal while maintaining the spatial orientation of original beamlets is the microchannel plate (MCP) consisting of many tiny parallel electron multiplying channels spaced tens of microns apart. Initial consideration on the use of a MCP as a detector revealed some useful advantages and some disadvantages that needed to be tested to determine whether they would limit the usefulness of the detector. The advantages which brought MCP's under initial consideration were:

1. The heavy ions could be converted to electrons which then interact with a detector material, thus side-stepping the problems seen with heavy ion damage.
2. MCP's could provide signal amplification once the heavy ions were converted to electrons while maintaining spatial resolution of the beamlet image.

Concerns regarding the MCP were:

1. Unknown damage response of the MCP due to intense exposure to heavy ions.
2. How to obtain fast gated images as commercially available MCP's use relatively slow phosphors as output screens.

A survey of commercially available devices showed that a detector with desirable characteristics could be purchased from several companies. A device purchased from Galileo Electro-Optics Corporation and used in these experiments is shown in Figure 5.3(a). A schematic of the device is shown in Figure 5.3(b) and its important characteristics shown in Table 5.2. An attractive feature of the Galileo MCP detector was the stainless steel front anode with minimum thickness of 1500-2000 Å plated onto the input side of the MCP. Its primary purpose is to allow placing a bias voltage across the channels. It is also thick enough to stop all the incident potassium ions in the Small Recirculator beam at their incident energies of 80-320 keV, thus protecting the microchannels from any damage that would affect their performance. To model the ability of the front metal anode to stop the heavy ions, the computer code TRIM (Transport of Ions in Matter) was used to calculate the ion range and statistics of their interaction with the metal anode.[36] Table 5.3 shows the calculated statistics of

320 keV potassium ions stopping in the MCP front metal anode.

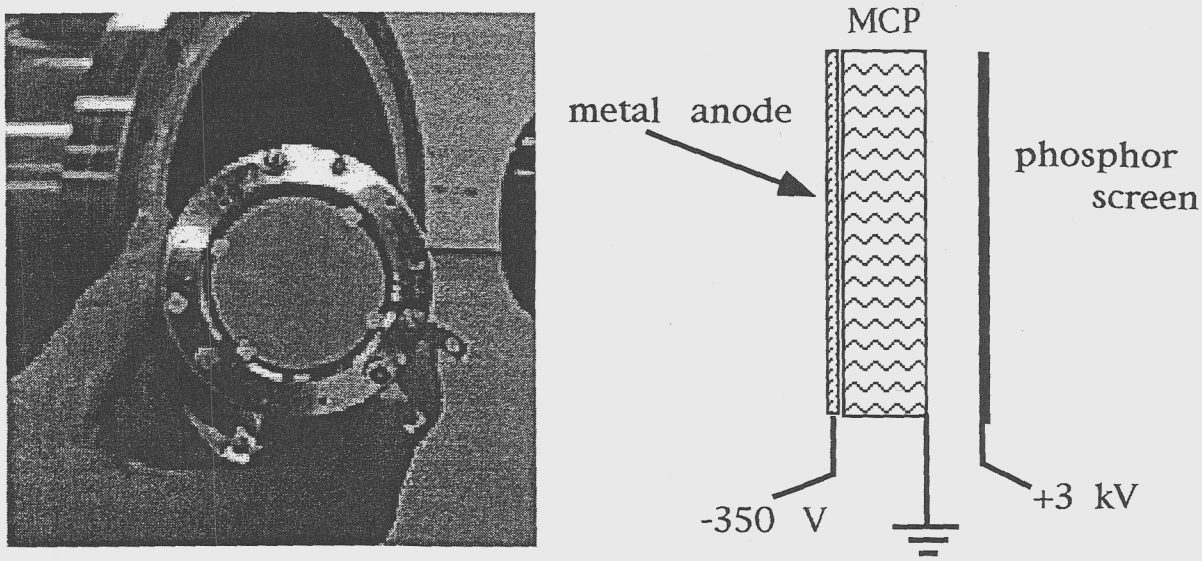


Figure 5.3: (a)Galileo Corp. microchannel plate and phosphor readout (b) MCP biasing scheme for detection of positive ions.

Table 5.2: Galileo microchannel plate characteristics[37]

Diameter	40 mm	Channel diameter	10 microns
Thickness	0.46 mm	Channel spacing	12 microns
Anode thickness	1500 Å	Channel bias angle	8°
Bias current	8-40 μamps	Channel coating	MgO
Phosphor screen	P20	Min. gain @1000V	5000

During the stopping of the ions in the metal anode, secondary electrons are produced. At 80 keV it is estimated that 3-5 secondary electrons with initial energies of a few eV are produced based on the empirical data in Kaminsky.[38] When the MCP is biased as shown in Figure 5.3(b) the electrons are accelerated into the MCP and amplified through interactions with the walls of the microchannels. A first order calculation of the number of secondary electrons produced in the anode from a 100 ns

long, 100 micron diameter beamlet formed from the Small Recirculator beam pulse shows that about 9.4×10^4 electrons would be produced. The MCP has a minimum amplification of 5000 at 1 kV bias so there is more than adequate electron signal from a single gated beamlet to produce an optical signal in the output phosphor. From the output of the MCP, the electrons are proximity focused across a thin gap by a strong positive bias of 3 kV onto a P20 phosphor screen which produces the image of the beamlet spot. The adjustable gain of the MCP allows amplification of very fast gated signals which proved problematic in the testing of fast plastic scintillator as a detector material.

Table 5.3: Transport of Ions in Matter (TRIM code) ion stopping statistics for the front metal anode of the MCP detector

	Range (Å)	Straggle (Å)
Longitudinal	363	170
Lateral	124	71
Radial	194	108
Vacancies/ion	750.7	
Energy loss (%)	Ions	Recoils
Ionization	22.67	18.39
Vacancies	0.14	1.73
Phonons	0.71	56.35

The microchannel axes of the MCP are manufactured at an 8° angle to the surface of the MCP. This improves the efficiency of the first interaction of incident electrons with the channel walls and prevents electrons from streaming through the channels without wall interactions. This 8° angle results in a 46 micron position offset of the output of the channels from the input, thus all the beamlet images are equally shifted by this amount. The high voltage used in the proximity focusing and the short distance from the channel output to the phosphor reduces to a minimum additional offset across the output gap. This shift in the overall position of the beamlet images was considered negligible in the moment calculations discussed in Chapter 4.

The spatial resolution of the MCP is determined from the size and spacing of its

microchannels. The Galileo MCP used in the GBI has channels 10 micron in diameter with a center-to-center spacing of 12 microns. A manufacturer's plot of resolution in microns versus channel spacing is shown in Figure 5.4. Given the channel spacing of 12 microns the uncertainty in detecting a "sharp" beamlet edge is 12 microns. This is an indication of the lower limit on the "graininess" of the beamlet images. However, it is not the limiting factor in system resolution as the pixel size on the camera CCD chip is 22.4 microns square, which is the limiting spatial factor in resolution. In fact, the optical resolution achieved in the GBI camera system during data runs was 54 microns per pixel, well within the MCP and CCD limit.

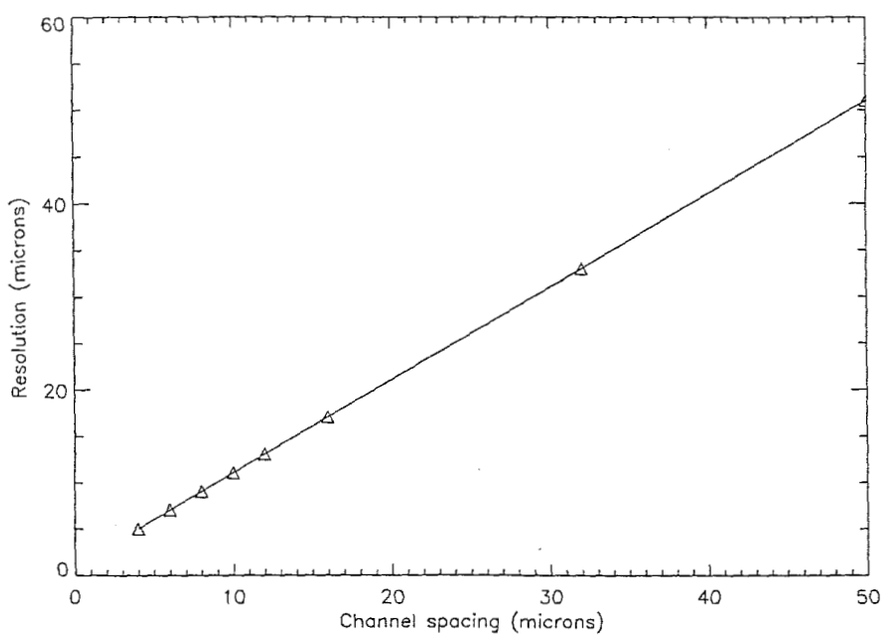


Figure 5.4: Microchannel plate resolution[37]

5.3 Obtaining Time-Resolved Beam Data by Gating the MCP

Using a MCP detector with a slow phosphor output precluded gating the CCD camera at a gate much shorter than the phosphor decay constant. To do so would include the image produced from ions arriving at the detector prior to the desired time interval. Faster gating required a different method than camera gating to be able to resolve the time history of the beam. It was necessary to gate the MCP on and off such that it was "opaque" to ions outside the desired time interval.

The attractive feature of the BC-412 fast plastic scintillator was its fast decay constant of 3.3 ns. This fast decay constant produced an image that contained the information from a 107.6 ns long slice of the beam when the camera was gated "ON" for 100 ns. The extra 7.6 ns of information results from light output corresponding to ions intercepted just prior to the camera gate. The fast decay constant easily allowed isolating narrow beam slices using the camera gate. On the other hand, the P20 phosphor of the MCP output has a decay constant of 200 μs, thus gating the camera for the desired 100 ns with the MCP operated in the DC mode "contaminates" the image obtained with the output of any previous beam impinging the detector prior to the gate interval. To overcome this difficulty a "Pinhole Neutron Experiment (PINEX)" MCP intensifier gate circuit and power supply (Figure 5.5) was obtained on loan from the Amador Valley Operations Office of Bechtel Nevada Corporation. The PINEX gate power supply is able to bias the MCP "ON" with a -385 V pulse for 100 ns duration. An oscilloscope trace of a 100 ns and a 1 μs gate signal is shown in Figure 5.6. The detector is effectively opaque to the secondary electrons generated by the heavy ions when not gated "ON", thus no optical output signal is produced. At these fast gates it was difficult to obtain a totally clean MCP bias signal with a rapid and smooth rise and fall. The trailing voltage spikes seen in the 100 ns trace of Figure 5.6(b) do not produce significant amplification of the beam signal due to the nonlinear response of the MCP to bias voltage, thus the gate signal is effectively 100 ns wide. Figure 5.7 is a schematic of the GBI equipment interconnections that were

necessary to obtain gated images of the beam.

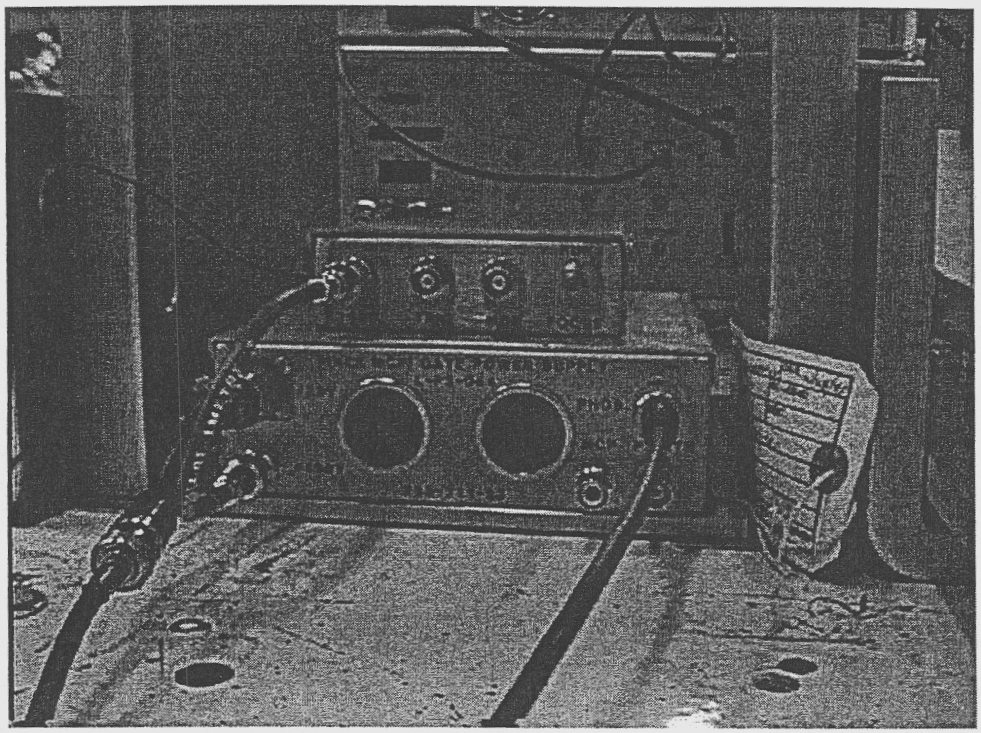


Figure 5.5: PINEX MCP gate circuit and power supply.

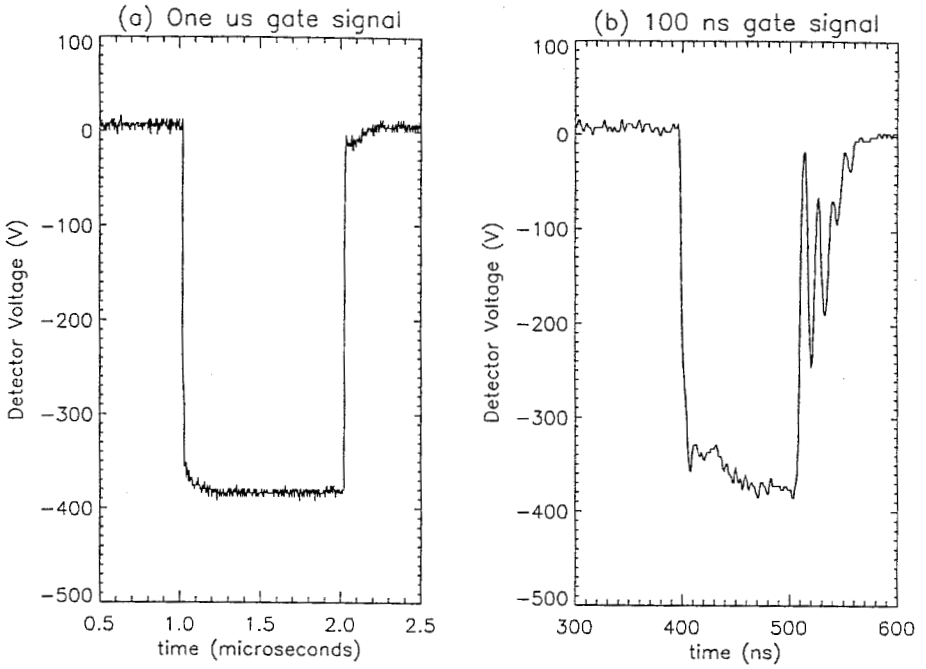


Figure 5.6: MCP detector gate signal **(a)** 1 μs gate used for image capture of a wide slice out of the beam flattop. **(b)** 100 ns gate used for the narrow slices to construct a time history of the beam.

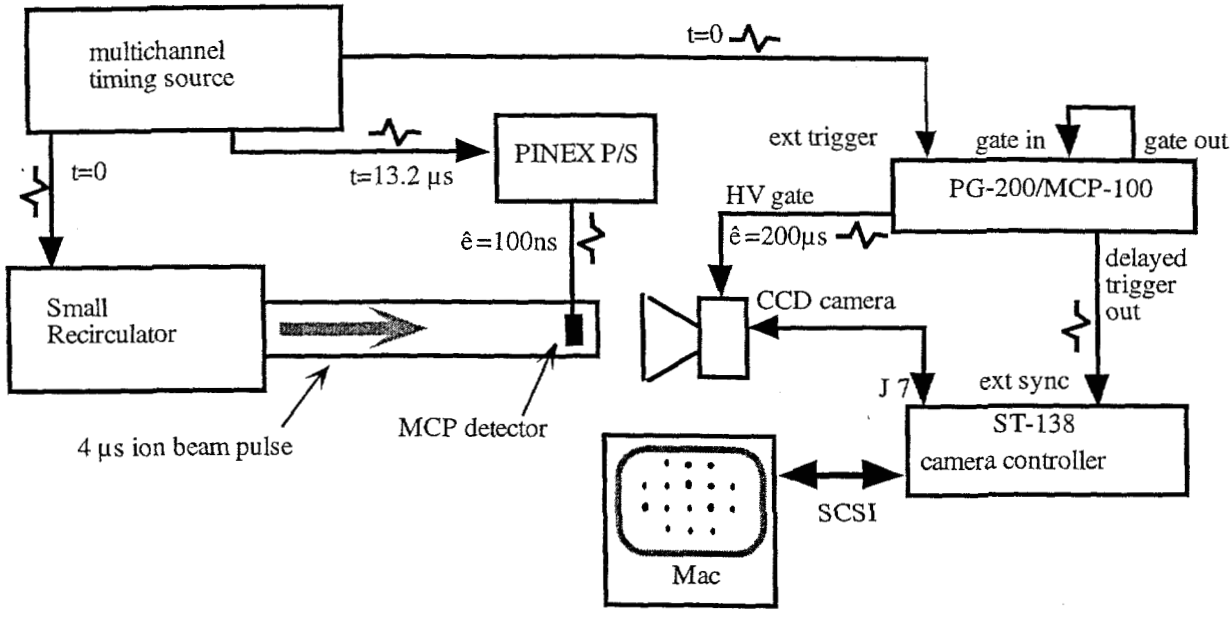


Figure 5.7: GBI equipment interconnections

5.4 Camera System Description

The Gated Beam Imager uses an intensified gated CCD camera to view the MCP detector output screen and capture the image of the beamlet on the detector. The camera is mounted outside the GBI diagnostic chamber in which the pepperpot and MCP detector are housed in the beamline as shown in Figure 5.8. The MCP phosphor screen is observed through a glass viewport either directly at the end of the beamline or with a mirror arrangement if placed in an intermediate diagnostic station. The camera system used was manufactured by Princeton Instruments, Inc. Inside the camera housing, a 384 by 576 pixel CCD of dimension 8.6 by 12.9 mm is fiber-optically coupled to an image intensifier allowing the observed image to be amplified and gated

to approximately 5 ns. To reduce dark charge buildup in the CCD during exposure and readout, it is seated on a cold finger which, in turn is seated on a thermoelectric (Peltier Effect) cooler. The cooler is mounted on a heat removal block cooled by air from cooling fans in the camera housing. To maximize the light transmission from the MCP phosphor screen to the camera CCD, I chose an f/1.2 Canon lens for its low f-stop and 85 mm focal length. In order to allow close-in focusing it was necessary to use a 32 mm lens extension between the camera and lens as shown in Figure 5.8. This allowed for a sharp focus to within 8 inches of the detector, and the appropriately sized field-of-view to maximize the spatial resolution of the data images.

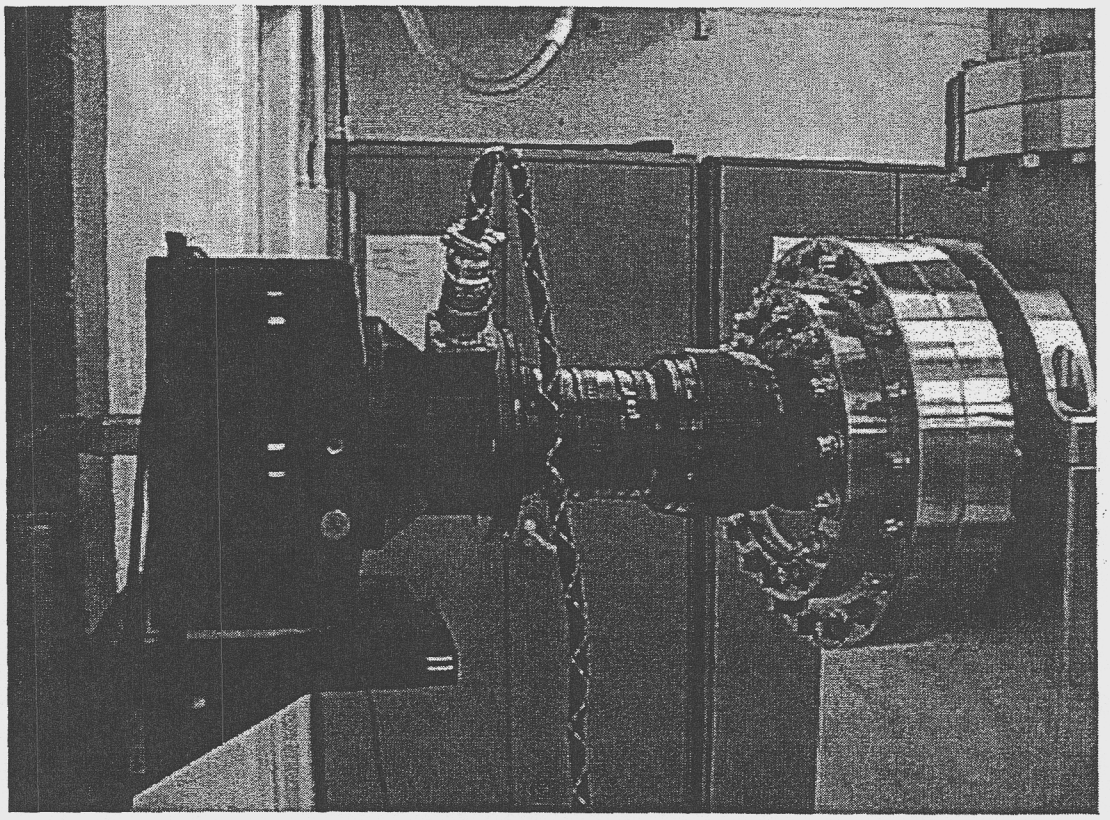


Figure 5.8: GBI CCD camera in position. Canon f/1.2 lens mounted on a 32 mm lens extension to provide close focusing to within 8 inches of the MCP detector.

A Princeton Instruments ST-138 CCD camera controller controls the readout of the CCD and the operation of the CCD cooler. The readout of the CCD after exposure is acquired into memory of a Macintosh computer via a serial interface card connected to the ST-138. Gating the camera intensifier on and off is accomplished with a Princeton Instruments PG-200 Programmable Pulse Generator. Both the ST-138 and PG-200 are shown in Figure 5.9. Voltage ranging from 500-1000 V can be applied to the camera intensifier to amplify light passed to the camera CCD. The PG-200 also controls the timing of the voltage to the intensifier in response to the experiment timing reference. A commercial software, IPLab *Spectrum*^{TM} with Princeton Instruments-supplied plug-in software modules was used to both control the camera exposure and display the CCD readout of the beamlet images immediately after acquisition.

5.5 Operating the Gated Beam Imager

To acquire a gated image of the beam, a time-of-flight calculation for a given beam energy is made to estimate when ions will arrive at the detector after being transported along the Small Recirculator. The head of the beam is then searched for by adjusting the actual gate start time while observing the images. The proximity focusing voltage of 3 kV from the microchannel output to the phosphor screen is applied continuously and the MCP is biased "ON" and "OFF" by the PINEX power supply. As mentioned before, the 100 ns gate on the MCP produces an image on the phosphor screen that has a 200 μs decay time. In order to increase the signal-to-noise ratio in the pixel voltages read out of the camera CCD it is desirable to gate the camera electronic shutter observing the MCP detector phosphor output. To accomplish this, the camera is gated on for 300 μs starting coincident with the MCP detector gate pulse. This gate proved to be optimum for excluding stray light in the GBI diagnostic chamber while allowing adequate signal for good beamlet spot resolution. A diagram of the timing sequence for acquiring a 100 ns gated image is shown in Figure 5.10. The long readout time (4.4 seconds) from the CCD to the

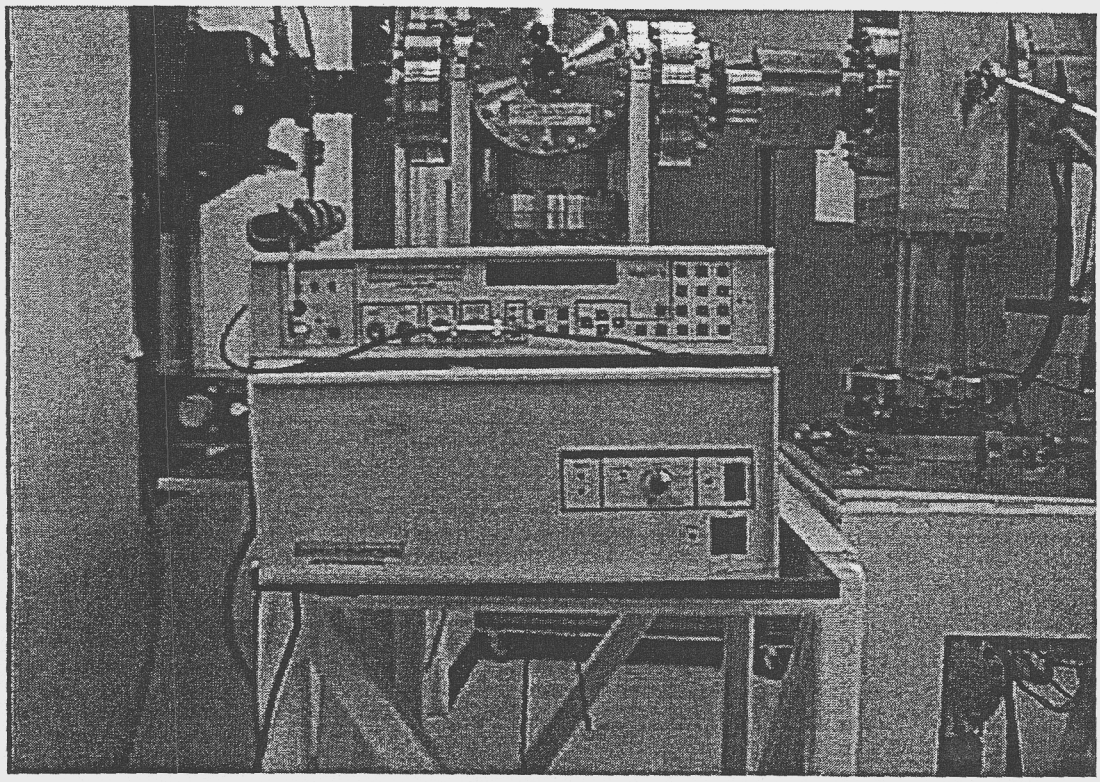


Figure 5.9: Princeton Instruments ST-138 CCD camera controller(lower) and PG-200 CCD intensifier power supply (upper). The wrapped coil on the left side of the PG-200 adds inductance to the trigger input cable to prevent prematurely triggering on the rapid electrical transients picked up from the ion source pulser spark gap switches.

computer is a consequence of the camera's "slow scan mode" clock frequency of 50 kHz. A slow readout of the camera CCD is important in limiting the readout noise in the image as readout noise is proportional to the readout clock frequency.[39] Reading out images when not collecting data, such as focusing the camera was done at a 1 MHz clock frequency resulting in a 0.22 second readout time allowing easier lens adjustment while viewing the image. After readout, the captured image is displayed and examined for quality prior to saving by the operator. For stepping through the beam, the trigger to the PINEX power supply is then delayed the appropriate interval (e.g., from 13.2 μs to 13.3 μs for 100 ns slices) and another image acquired on the next beam shot.

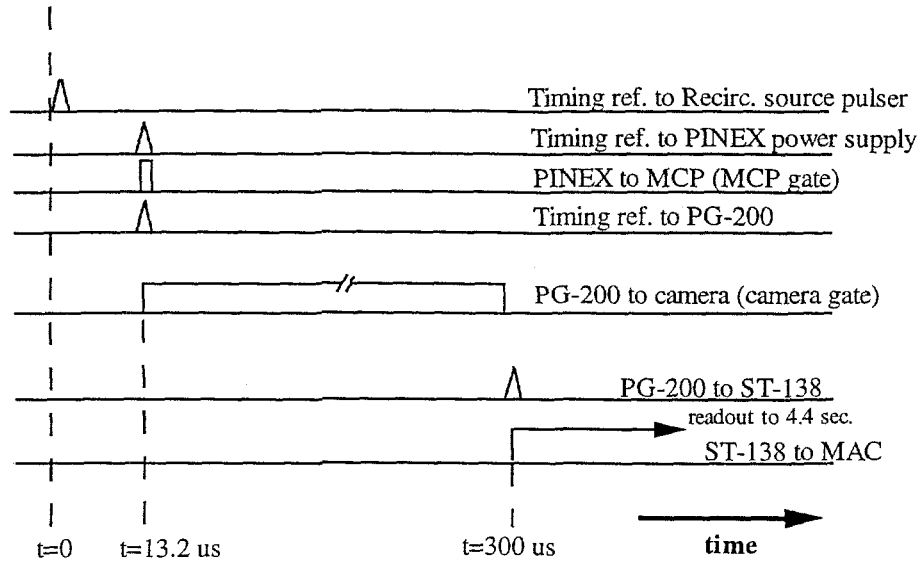


Figure 5.10: GBI image capture timing diagram.

5.6 Applying Corrections to the GBI Images

It was necessary to apply corrections to the data images in order to account for the response of the CCD to dark charge accumulation during exposure and readout and for differing pixel sensitivities that result from manufacturing tolerances from pixel-to-pixel in the same CCD. Corrections were also made to account for the uneven response across the MCP detector to an even intensity input.

5.6.1 CCD Dark Charge Correction

Dark charge accumulation is a function of exposure and readout time, and the temperature of the CCD.[40] Several dark charge measurements with the same exposure and readout times as data images were made during the data runs to account for any change due temperature drift of the camera cooling system, although changes are small due to the camera controller's ability to maintain CCD temperature within

± 0.05°C.[41] Figure 5.11 shows a typical dark charge result displayed by averaging the CCD pixel rows. The dark charge buildup is fully repeatable given a constant temperature and readout time between exposures and is corrected for by individual pixel value subtraction.

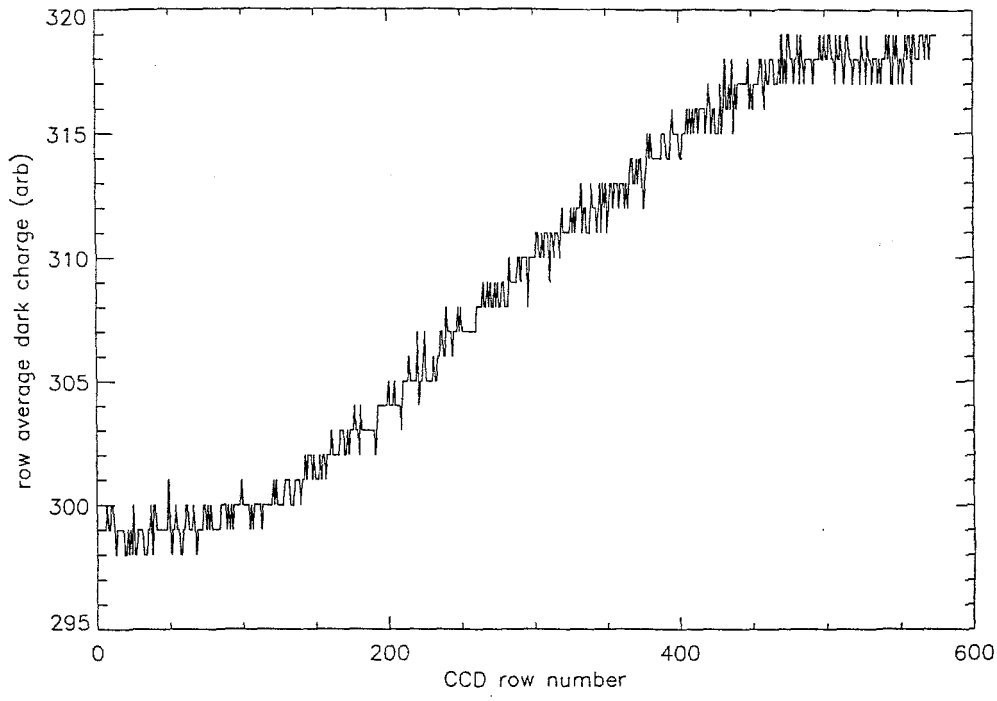


Figure 5.11: Image row average of camera CCD pixel values showing dark charge buildup during the finite time taken to read out the CCD by row from left to right.

5.6.2 CCD Flat Field Response Correction

Each CCD chip is unique in its pixel-to-pixel sensitivity due to slight variations in manufacturing.[40] As well as being unique, the response to the same light level is fully repeatable from exposure to exposure. The difference in pixel sensitivity was determined with a "flat field" image which can be used to multiplicativly correct a data image. The "flat field" image was obtained using the "integrating light sphere"

shown in Figure 5.12. The interior surface of the sphere is coated with a smooth white diffusing material which provides an evenly illuminated image from a light source in the side of the sphere. A flat field image was taken using a 300 μs exposure along with a nearly concurrent dark charge image to account for the CCD temperature at the exposure time. Along with these flat field and dark charge images, a data image of the GBI beamlet spots is manipulated within the IPLab *Spectrum*^{TM} software to correct for the dark charge buildup on the CCD during exposure and the uneven flat field response. The maximum dark charge level for a 300 μs exposure was about 1.2% of the peak spot intensity and the peak variation in the flat field measurement was $\pm$16%. Pixel by pixel the correction is made using the formula[32] :

$$Corrected\ Image = \frac{Mean(Flat - DarkF)(Image - DarkI)}{(Flat - DarkF)} \tag{5.1}$$

where "*Flat*" is a flat field image, "*DarkF*" is a dark charge image associated with the flat field image, "*Image*" is an image of the beamlet spots, and "*DarkI*" is a dark charge image associated with the beamlet spots image.

5.6.3 MCP Detector Flat Field Response Correction

The third correction made to all data images is a correction to account for the variation in response of the MCP microchannels to a flat field "illumination".[37] Ideally all microchannels would give the same output for like input as would the phosphor screen. Lacking the ability to generate a detector-size flat field of heavy ions, I tested the MCP using a flat field x-ray source at LLNL. The machine shown in Figure 5.13 with the GBI attached to it generates 7.47 keV x-rays from a nickel anode. Figure 5.14 shows the field-of-view of the CCD camera just prior to x-ray illumination and Figure 5.15 shows the flat field image obtained. The response of the MCP varied by as much as 70% from one edge to the other in a gradual and even manner. Extending this response to an experiment, equivalent beamlets would be weighted more or less in moment calculations depending on their position on the MCP.

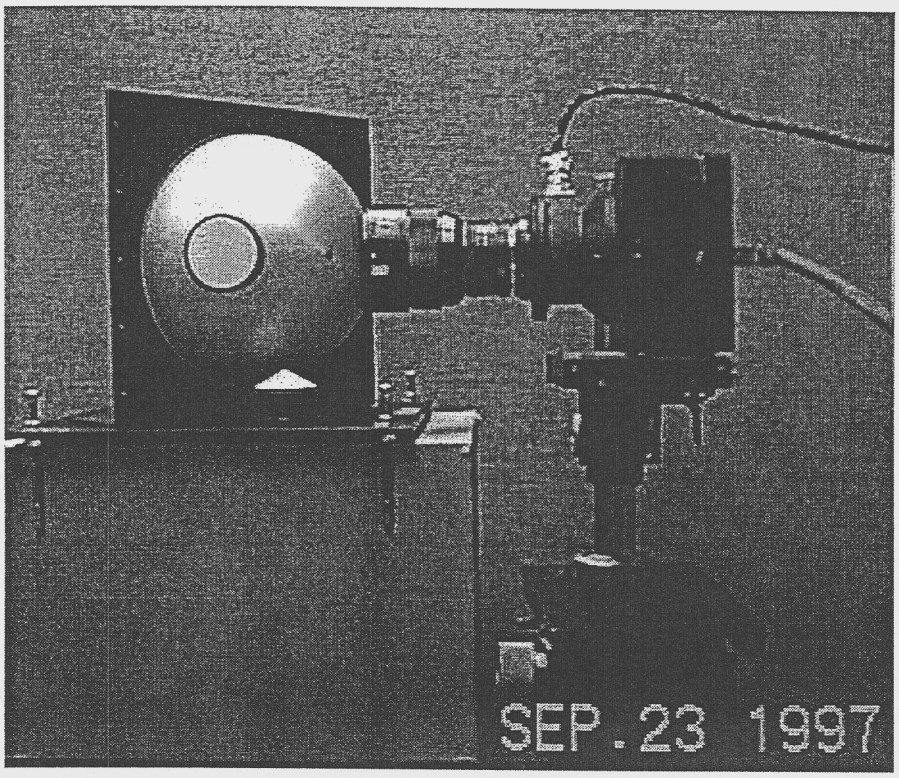


Figure 5.12: CCD camera and integrating light sphere setup for obtaining a flat field image.

Figure 5.13: GBI attached to a LLNL x-ray source to measure the spatial response of the MCP to a uniform signal.

Once an evenly illuminated MCP image is obtained, the data images can be corrected. As the field of view of the CCD camera differs from the experiment setup to the MCP flat field setup, it is impossible to match up the exact position of the two images. This is not a severe limitation as the change in MCP response is gradual across the detector. The four fiducial marks shown in Figure 5.14 around the periphery of the detector allow the orientation of the flat field and data images to be matched. The larger field of view of the flat field image is measured and the image is extrapolated to the same size as the data image using the IPLab imaging software. The data image is then corrected for the non-uniform response of the MCP using the algorithm of Equation 5.1.

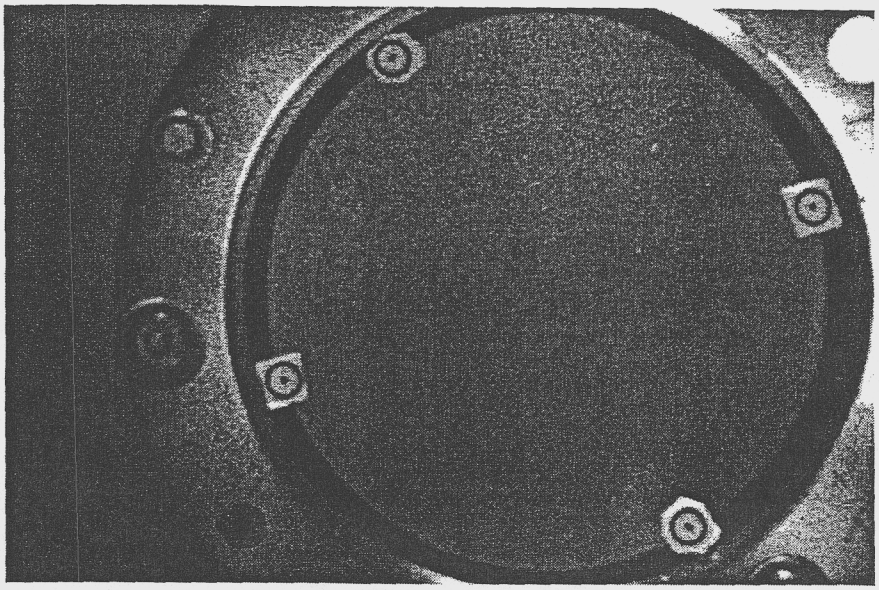


Figure 5.14: Field-of-view of the CCD camera observing the MCP detector for the MCP flat field x-ray measurements.

5.6.4 Preparing the Images for Data Analysis

Using a script of IPLab imaging software commands, the image corrections are applied in a matter of seconds. The corrected image in the form of a 2-D array (384 by 576) is then saved as a text file with each array address corresponding to a CCD pixel value representing the intensity of light which fell on the pixel. The array is then sent to the DEC Alpha workstation running the IDL pepperpot data analysis code. To conserve memory the array values are linearly scaled from 16 bit digitization (0 to 65535), to 8 bit digitization (0 to 256) and read into memory. Figure 5.16 shows a small part of a text array that includes a beamlet spot. Data analysis then can be carried out in accordance with the methods discussed in Chapter 4.

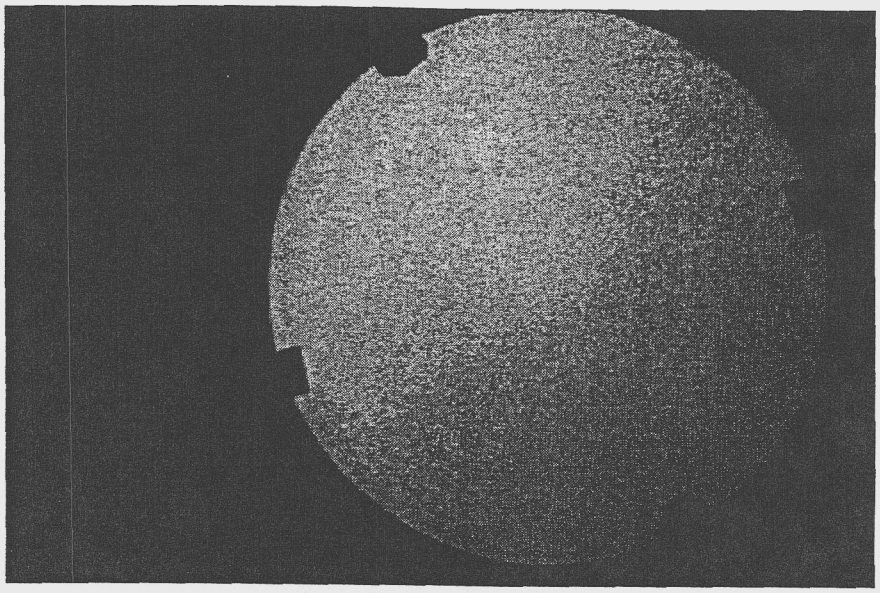


Figure 5.15: View of the non-uniform response of the MCP detector to a uniform field of 7.47 keV x-rays. The field of view that matches that of the experimental measurements is selected from this image and applied as a correction for the non-uniform response.

3	3	3	5	7	7	6	4	4	4	4	4	3	3	2	2	2	1
3	3	2	3	5	5	5	4	4	5	5	5	4	3	3	2	3	1
4	4	2	3	4	5	4	4	5	7	6	6	6	5	4	3	4	2
3	3	2	2	4	4	5	7	9	12	12	11	9	7	5	3	4	2
2	3	3	3	4	5	7	11	15	19	19	17	12	10	7	4	4	3
2	2	2	3	4	5	10	18	29	39	46	42	31	22	12	7	4	2
2	3	3	4	5	7	14	27	44	60	74	69	50	34	19	11	5	3
2	2	3	4	5	8	17	35	66	98	122	116	88	58	32	18	8	4
2	3	4	5	6	11	21	45	89	137	171	165	126	83	46	26	12	6
2	3	3	4	6	12	24	50	100	160	197	190	143	95	53	30	13	7
3	4	3	5	7	14	29	56	112	185	224	217	160	108	61	34	14	8
2	4	3	4	6	13	27	52	100	159	191	184	140	91	51	28	11	7
3	4	4	4	7	13	27	48	88	135	158	152	120	75	43	22	9	6
2	3	3	3	5	10	21	39	66	97	111	106	83	53	31	17	7	5
3	2	3	3	4	9	17	30	45	61	64	60	47	33	21	12	6	4
2	2	2	2	4	7	13	22	30	41	43	40	30	22	15	10	5	3
3	3	3	3	4	7	11	15	17	22	22	20	15	13	11	8	5	3
3	3	2	3	4	6	9	12	13	16	14	13	10	8	7	6	3	2
4	4	3	4	4	6	8	10	10	10	7	7	6	5	5	4	3	2
3	3	2	3	4	5	7	8	8	7	4	5	5	4	3	3	2	2
2	2	3	3	4	4	6	7	7	5	3	4	4	4	3	2	2	2

Figure 5.16 A small sample of a beamlet spot data array file in text format encompassing one beamlet spot.

Chapter 6

A Beam Correction System

The utility of measurements possible with the GBI in the design, construction, and operation of heavy ion accelerators can be realized in several ways. Experiments such as the LLNL Small Recirculator are meant to push the state of the art in induction accelerator design and in intense heavy ion beam control. More advanced measurements of more beam parameters are essential to evaluating its performance. The Small Recirculator is intended to produce data that will benchmark a numerical simulation code such as WARP.[15] One area that is ideally suited to being diagnosed by the GBI is the effect on the beam of rotationally misaligned quadrupole focusing magnets. The amount of misalignment that can be tolerated is an essential factor in quadrupole magnet design and accelerator construction costs. Measuring the effect of misalignments and removing the effect prior to final focus could increase the quadrupole magnet alignment tolerances and thus reduce accelerator construction and maintenance costs.

6.1 The Effect on the Beam of Rotationally Misaligned Quadrupoles

A magnetic quadrupole rotated about the beam axis will not cause a displacement of the beam centroid off-axis if it is initially on-axis.[29] However, a rotated quadrupole will create a coupling between the previously uncoupled transverse x and y motions

of beam ions. The beam control elements such as the quadrupoles, steering dipoles, insertion/extraction magnets, and beam final focus magnets are all assumed to be aligned to the laboratory frame. Barnard showed that in the presence of rotational misalignments, the beam 2-D transverse emittance is not conserved even under linear equations of motion, and the beam acquires a finite angular motion.[29] The detrimental effect of this on the final focus element performance is indeed a real effect as any final focus lens will be designed to be aligned for an assumed orientation of the beam (most likely, upright in the laboratory frame). As the misaligned quadrupole still exerts a linear force on the beam it is possible to "correct" its effect using a suitable number of correction elements generating the proper combination of fields. Linear forces on the beam do not result in hopelessly entrained unoccupied phase space in the beam distribution as in Figure 4.1. A conceptual beam correction system is proposed in the following discussion (see Figure 6.6) that would use as input, the moment data obtained from the GBI.

Barnard derived beam first and second moment evolution equations for a beam transported and accelerated through an alternating gradient focusing, or FODO lattice.[29] An outline of the derivation appears in Appendix B. He assumed that:

1. The force on the ions comes from two sources; the external focusing (defocusing) from the quadrupolar field, and the internal defocusing from space charge of the beam.

2. The space charge is distributed in a uniform ellipse as in a K-V distribution.

Stating the results obtained in Appendix B the two closed sets of coupled moment evolution equations for the first and second moments of the beam distribution are:

<u>First Moments</u>

$$\frac{d<x>}{dz} = <x'> \qquad \frac{d<x'>}{dz} = K_{qxx} < x > + K_{qxy} < y > - (dln\beta_z/dz) < x' >$$

$$\frac{d<y>}{dz} = <y'> \qquad \frac{d<y'>}{dz} = K_{qyy} < y > + K_{qyx} < x > - (dln\beta_z/dz) < y' >$$

Second Moments

$$\frac{d\Delta x^2}{dz} = 2\Delta xx'$$

$$\frac{d\Delta xx'}{dz} = \Delta x'^2 + K_{xx}\Delta x^2 + K_{xy}\Delta xy - (dln\beta_z/dz)\Delta xx'$$

$$\frac{d\Delta x'^2}{dz} = 2K_{xx}\Delta xx' + 2K_{xy}\Delta x'y - 2(dln\beta_z/dz)\Delta x'^2$$

$$\frac{d\Delta y^2}{dz} = 2\Delta yy'$$

$$\frac{d\Delta yy'}{dz} = \Delta y'^2 + K_{yy}\Delta y^2 + K_{yx}\Delta xy - (dln\beta_z/dz)\Delta yy'$$

$$\frac{d\Delta y'^2}{dz} = 2K_{yy}\Delta yy' + 2K_{xy}\Delta xy' - 2(dln\beta_z/dz)\Delta y'^2$$

$$\frac{d\Delta xy}{dz} = \Delta x'y + \Delta xy'$$

$$\frac{d\Delta x'y}{dz} = \Delta x'y' + K_{xx}\Delta xy + K_{yx}\Delta y^2 - (dln\beta_z/dz)\Delta x'y$$

$$\frac{d\Delta xy'}{dz} = \Delta x'y' + K_{yy}\Delta xy + K_{xy}\Delta x^2 - (dln\beta_z/dz)\Delta xy'$$

$$\frac{d\Delta x'y'}{dz} = K_{xx}\Delta xy' + K_{xy}\Delta yy' + K_{yy}\Delta x'y + K_{yx}\Delta xx' - 2(dln\beta_z/dz)\Delta x'y'$$

where the notation $\Delta ab \equiv (< a^2 > - < a >^2)(< b^2 > - < b >^2)$ stands for a second moment with the contribution from a non-zero first moment removed. The coefficients in the evolution equations in terms of the external quadrupolar and internal space-charge forces and angles defined in Figure 6.1 are:

$$K_{qxx} = -K_{qyy} = K_{qx0}\cos 2\theta \qquad K_{qxy} = K_{qyx} = K_{qx0}\sin 2\theta$$

$$K_{sxx} = K_{sxb}\cos 2\alpha + K_{syb}\sin 2\alpha \qquad K_{syy} = K_{syb}\cos 2\alpha + K_{sxb}\sin 2\alpha$$

$$K_{sxy} = K_{syx} = (K_{sxb} - K_{syb})\sin\alpha\cos\alpha \qquad K_{qx0} = -K_{qy0} = \frac{B'}{B\rho}$$

$$K_{sxb} = \frac{\kappa}{2(\Delta x_b^2 + (\Delta x_b^2 \Delta y_b^2)^{1/2})} \qquad K_{syb} = \frac{K}{2(\Delta y_b^2 + (\Delta x_b^2 \Delta y_b^2)^{1/2})}$$

Barnard also found two independent constants of the motion of a beam in a FODO lattice with rotated quadrupoles inducing coupled motion between x and y.[29] As the

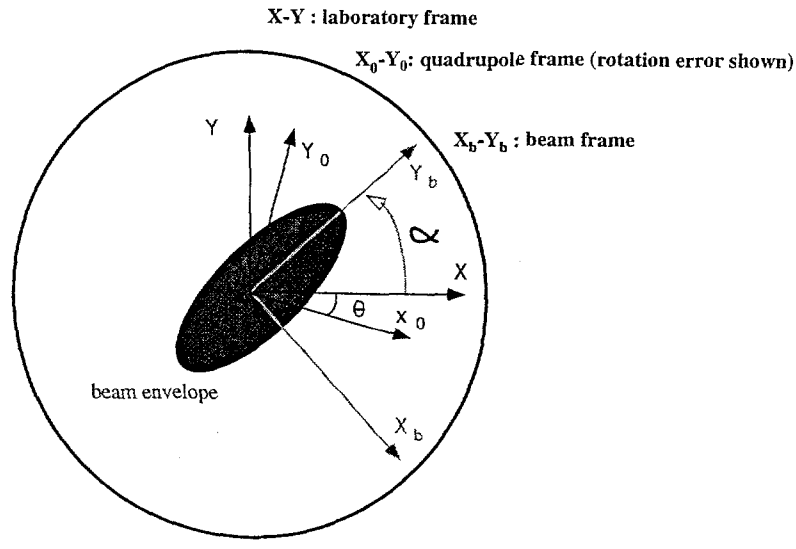


Figure 6.1: Beam and laboratory coordinate system.

emittance is a constant of the motion in an aligned FODO lattice, Barnard called these constants "generalized emittances". Their definitions are:

$$\epsilon_{ng} = \left[\frac{1}{2}\epsilon_{nx}^2 + \frac{1}{2}\epsilon_{ny}^2 + (4\beta)^2(\Delta xy\Delta x'y' - \Delta xy'\Delta x'y)\right]^{1/2} \tag{6.1}$$

$$\epsilon_{nh} = \left[(\epsilon_{nx}^2\epsilon_{ny}^2 + (4\beta)^4([\Delta x'y']^2[\Delta xy]^2 - 2\Delta x'y\Delta x'y'\Delta xy\Delta xy' + [\Delta x'y]^2\right]^{1/2} \tag{6.2}$$

where the terms ϵ_{nx} and ϵ_{ny} are the normalized transverse emittances defined in terms of the moments as:

$$\epsilon_{nx} = 4\beta\gamma\left[\Delta x^2\Delta x'^2 - (\Delta xx')^2\right]^{1/2} \tag{6.3}$$

$$\epsilon_{ny} = 4\beta\gamma\left[\Delta y^2\Delta y'^2 - (\Delta yy')^2\right]^{1/2} \tag{6.4}$$

Any other combinations of second moments that yield constants can be reduced to these transverse and generalized emittances and in fact, these quantities reduce to linear combinations of the standard transverse emittances in an aligned FODO lattice

which can be easily verified by setting the cross-moment terms to zero.[42] To verify that the generalized emittances were indeed constants of the motion and to study the emittance growth caused by random quadrupole rotations Barnard wrote a computer code, "MOMENT" which numerically integrates the moment equations through a FODO lattice representing the C-design of a recirculator driver.[29]

6.2 Approximate Correction for a Single Misaligned Quadrupole

I modified the MOMENT code into several versions to support calculations to show that with a set of four adjustable magnetic quadrupoles, (variable rotation angle and field strength) and the information from the Gated Beam Imager (full set of beam moments), any effect of rotation errors could be removed from the beam and the transverse emittance growth caused by these errors removed from the beam.

The first modification (FULLAG in Appendix B) of the MOMENT code was to tailor its model parameters to match those of the Small Recirculator and to allow for the placing of known rotations on specific quadrupoles or to create random rotations on all quadrupoles. Figures 6.2 and 6.3 show the effect on the beam moments of a single misaligned quadrupole rotated 5° counter-clockwise. A large error of 5° was chosen to support experimental verification of the effect of rotation errors on the Small Recirculator. The evolution of the moments from the simulation is plotted with and without the error for comparison. There is only a slight effect on the x and y moments but the $x-y$ cross moments become non-zero indicating the the motion of the ions in the two orthogonal directions becomes coupled after the misalignment. The beam acquires a rapidly changing angular momentum as it passes through subsequent quadrupoles and the 2-D transverse emittance rises above its original value. An attempt to focus the beam to a "spot" at a point past the misalignment would not be as successful as had the misalignment not been present. Note that the generalized emittances in Figure 6.3 are unaffected by the rotation error as expected.

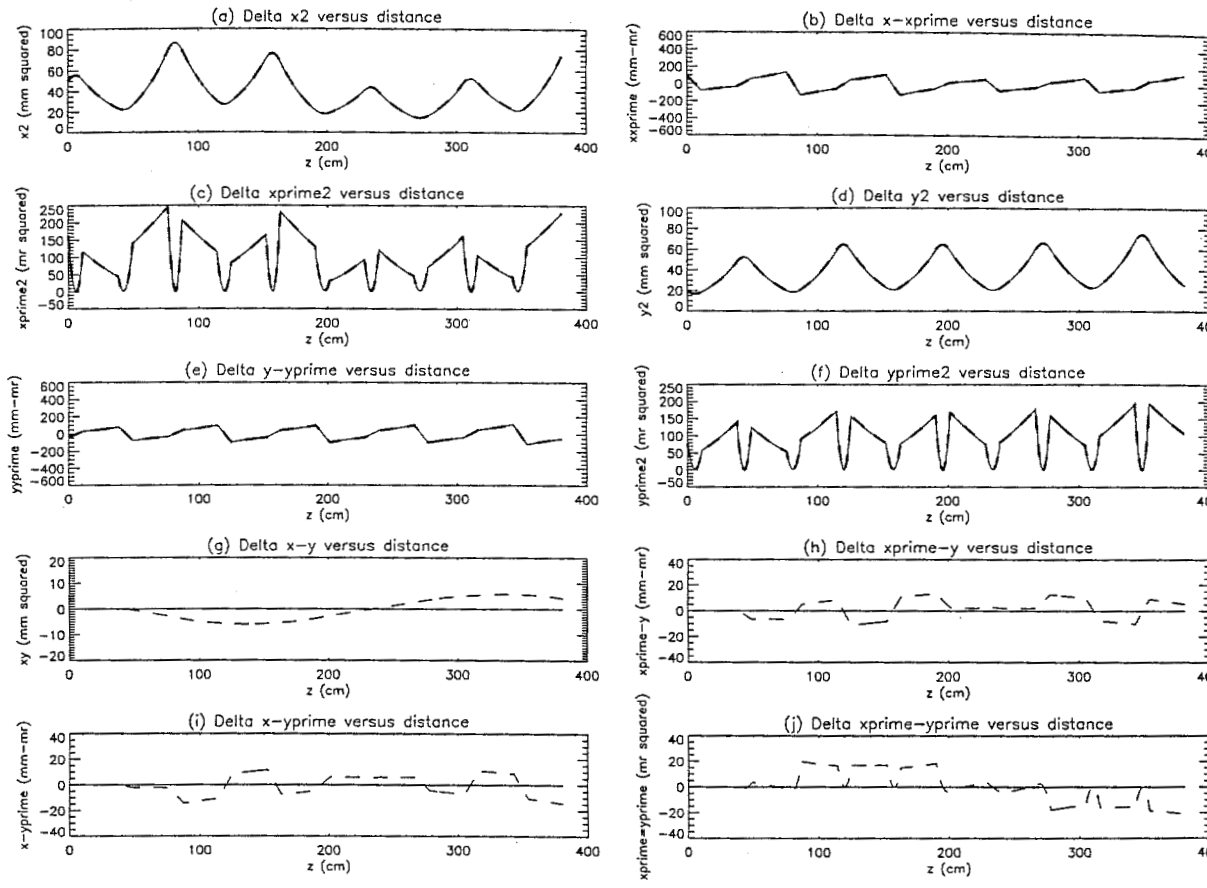


Figure 6.2: Effect of a misaligned quadrupole on the beam moments as a function of distance along the accelerator axis. Dashed lines denote the moments with a single error, quadrupole 2 at z=38.1 cm rotated 5° counter-clockwise. Solid lines are without a rotation error. Output from FULLAG.

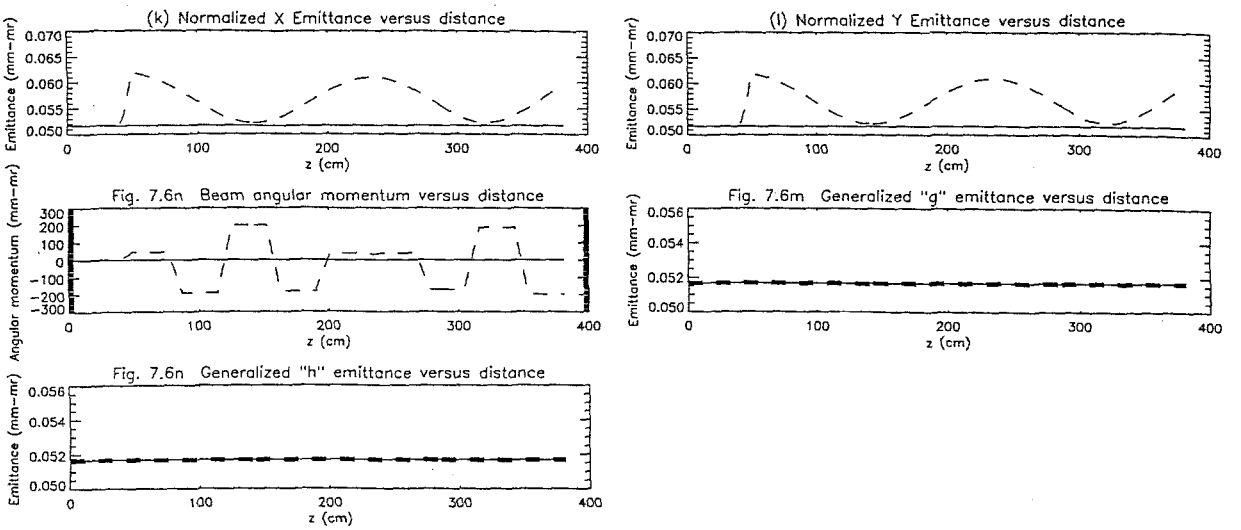


Figure 6.3: Effect of a misaligned quadrupole on the beam emittances and angular momentum. Dashed lines denote the quantities with a single error, quadrupole 2 at z=38.1 cm rotated 5° counter-clockwise. Solid lines are without a rotation error. Output of FULLAG.

A special case of an approximate beam correction method was noted while running the code with multiple known quadrupole rotations. With a known rotation error and beam phase advance (here 72°) that divides an integral number of times into 360°, the effect on the beam can be approximately removed by counter-rotating a quadrupole 5 half lattice periods subsequent to the misaligned quadrupole. This provided a rudimentary method of testing beam correction on the Small Recirculator. These tests are discussed in Chapter 7. Figures 6.4 and 6.5 show the results of a simulation run with quadrupole 2 rotated 5° counter-clockwise, as above, and quadrupole 7 rotated 5° clockwise. The motion between the x and y directions is mostly decoupled as shown in Figure 6.5 as the $x - y$ cross moments return to near zero. The 2-D transverse emittances return to their original values leading one to conclude that focusing the beam downstream of the quadrupoles would attain the almost the same same results as if the quadrupoles were perfectly aligned. Determining how to correct for any combination of upstream rotation errors would allow quadrupole alignment tolerances to increase, presumably simplifying the design and construction process of

a driver size machine while maintaining the performance of an aligned lattice.

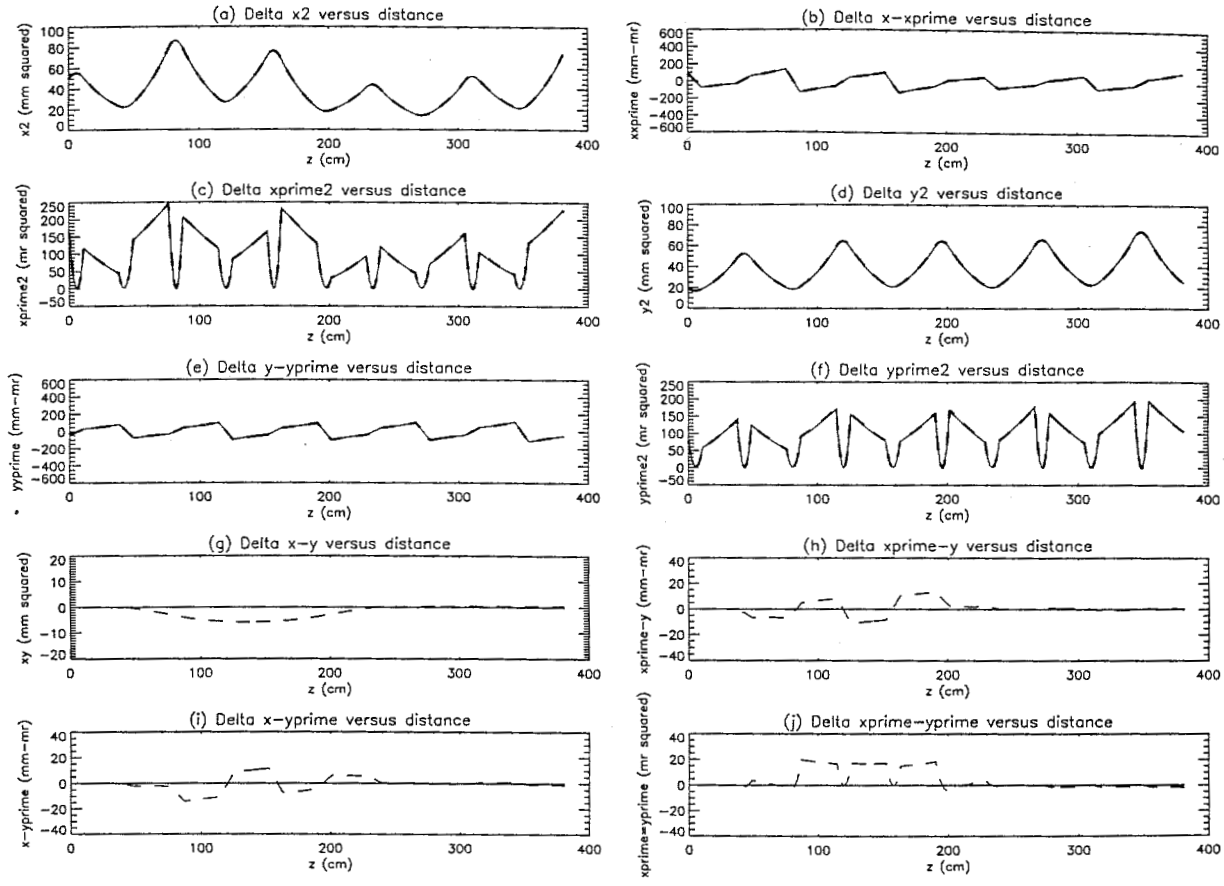


Figure 6.4: Correction of a known quadrupole rotation error. Quadrupole 2 at 38.1 cm rotated 5° counter-clockwise and quadrupole 7 at 228.6 cm rotated 5° clockwise. Dashed lines denote the moments with rotated quadrupoles. Solid lines are without quadrupole rotations. Output from FULLAG.

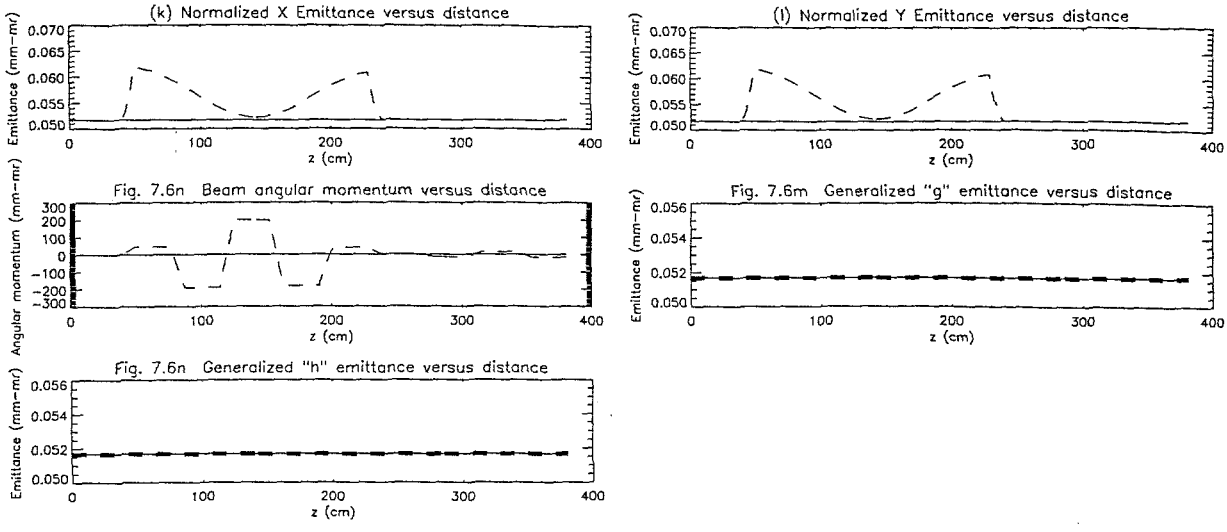


Figure 6.5: Correction of a known quadrupole rotation error. Quadrupole 2 at 38.1 cm rotated 5° counter-clockwise and quadrupole 7 at 228.6 cm rotated 5° clockwise. Dashed lines denote the quantities with rotated quadrupoles. Solid lines are without quadrupole rotations. Output from FULLAG.

6.3 Correction for Random Quadrupole Rotation Misalignments

For the more general case and one that exists in a real machine, the location and magnitude of quadrupole rotation errors is not known, only the moments of the beam can be measured at points along the beamline where a suitable diagnostic such as the GBI can be placed. Non-zero cross-moments would indicate the presence of quadrupole rotation errors causing coupling between x and y motions. Examining the second moment evolution equations one sees that they consist of a closed set of ten equations with ten independent parameters (the moments). A matrix representation of the equations at constant energy is:

$$\frac{d}{dz}\begin{pmatrix} \Delta x^2 \\ \Delta xx' \\ \Delta x'^2 \\ \Delta y^2 \\ \Delta yy' \\ \Delta y'^2 \\ \Delta xy \\ \Delta x'y \\ \Delta xy' \\ \Delta x'y' \end{pmatrix} = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{xx} & 0 & 1 & 0 & 0 & 0 & K_{xy} & 0 & 0 & 0 \\ 0 & 2K_{xx} & 0 & 0 & 0 & 0 & 0 & 2K_{xy} & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{yy} & 0 & 1 & K_{yx} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2K_{yy} & 0 & 0 & 0 & 2K_{xy} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & K_{yx} & 0 & 0 & K_{xx} & 0 & 0 & 1 \\ K_{xy} & 0 & 0 & 0 & 0 & 0 & K_{yy} & 0 & 0 & 1 \\ 0 & K_{yx} & 0 & 0 & K_{xy} & 0 & 0 & K_{yy} & K_{xx} & 0 \end{pmatrix} \begin{pmatrix} \Delta x^2 \\ \Delta xx' \\ \Delta x'^2 \\ \Delta y^2 \\ \Delta yy' \\ \Delta y'^2 \\ \Delta xy \\ \Delta x'y \\ \Delta xy' \\ \Delta x'y' \end{pmatrix} \tag{6.5}$$

The beam can be manipulated to adjust these moments to the desired values. To simultaneously adjust the ten moments to desired values requires ten adjustable parameters.

A set of five quadrupole magnets with variable rotation angle and variable field strength would supply these manipulations. The generalized emittances provide relief in the engineering and cost of a beam correction system in that they provide constraints on the system of moment equations allowing the elimination of two parameters by solving explicitly for them in terms of the other parameters. Applying the two constraints and eliminating Δx^2 and Δy^2 by substituting in their explicit

solutions yields:

$$\frac{d}{dz}\begin{pmatrix} \Delta xx' \\ \Delta x'^2 \\ \Delta yy' \\ \Delta y'^2 \\ \Delta xy \\ \Delta x'y \\ \Delta xy' \\ \Delta x'y' \end{pmatrix} = \begin{pmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} & K_{17} & K_{18} \\ 2K_{xx} & 0 & 0 & 0 & 0 & 2K_{xy} & 0 & 0 \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} & K_{37} & K_{38} \\ 0 & 0 & 2K_{yy} & 0 & 0 & 0 & 2K_{xy} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66} & K_{67} & K_{68} \\ K_{71} & K_{72} & K_{73} & K_{74} & K_{75} & K_{76} & K_{77} & K_{78} \\ K_{yx} & 0 & K_{xy} & 0 & 0 & K_{yy} & K_{xx} & 0 \end{pmatrix} \begin{pmatrix} \Delta xx' \\ \Delta x'^2 \\ \Delta yy' \\ \Delta y'^2 \\ \Delta xy \\ \Delta x'y \\ \Delta xy' \\ \Delta x'y' \end{pmatrix} \tag{6.6}$$

where the numbered coefficients are complicated functions of the second moments but are nevertheless calculable in a code, if not manually on paper. The resultant system of eight coupled equations and eight parameters allows beam correction to be accomplished by a system of four adjustable quadrupole magnets. A schematic of such a system is shown in Figure 6.6. Four adjustable magnetic quadrupoles replace four standard quadrupoles at some point in the beamline. A natural placement may be just before the final focus lens to ensure the beam enters in the best condition possible (upright, with zero angular momentum) for effective focusing. Other positions within the recirculator may be more effective in reducing the cumulative effect of quadrupole rotation errors. The difference between a regular quadrupole and these are in their ability to rotate the orientation of their magnetic fields and may have a greater degree of field strength adjustment.[43]

6.3.1 Applying GBI Data to a Beam Correction System

A GBI-type instrument would measure the moments of the beam at the input side of the correction system providing information on the condition of the beam as in Figure 6.6. A simulation code run would provide the desired (and attainable) moments at the output of the correction system. A benchmarked and calibrated code such as WARP would provide moments that included the unavoidable effects of the

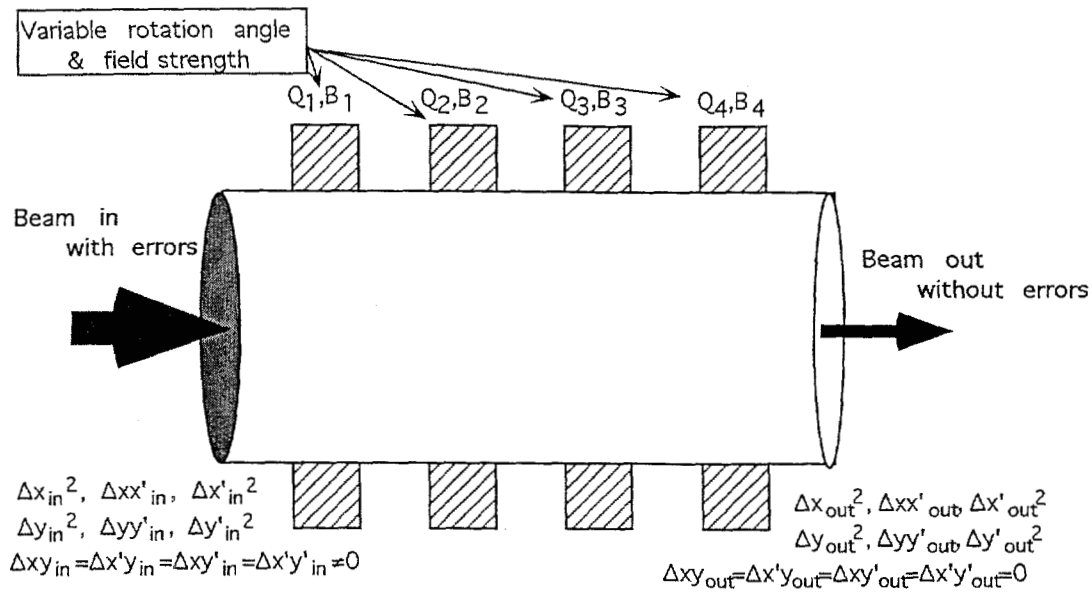


Figure 6.6: Conceptual beam correction system.

actual fields and resonances that result in nonlinear emittance growth even with a perfectly aligned model FODO lattice. This nonlinear effect on the moments cannot be removed through the application of linear fields so the desired moments calculated for the output of the correction system must be obtained from a sophisticated simulation that includes nonlinear effects. The simplest useful correction would be to correct the cross moments to zero which would deliver an upright and non-rotating beam (x and y motions de-coupled) to the final focus. This may not be the best that can be done though which is why a sophisticated code which estimates nonlinear growth in an aligned lattice is desirable. Once the input moments are measured and the output moments are calculated, an optimization of the adjustable parameters in the correction quadrupoles is calculated and set until a decrease in beam transport element performance warrants another GBI measurement. A GBI measurement at the output of the correction system would be useful in directly verifying its performance.

I modified Barnard's MOMENT code into a third version to serve as an optimizer

to calculate the field strengths and rotation angles of a correction system to remove the effect of quadrupole rotation errors. A copy of this code, "OPTIMIZE" is in Appendix C. Using the generalized emittance equations to explicitly solve for two moments proved quickly to be too difficult to do by hand. Using the simultaneous equation solving capabilities of *Mathematica*^{TM}, a symbolic mathematics software system, allowed me to do this task and to find the simplest solution of the ten possibilities.[49] Solving explicitly for Δx^2 and Δy^2 resulted in the 8 simplest moment evolution equations with 8 adjustable parameters. Even though these solutions were the simplest obtainable, the results consist of algebraic equations of over 100 terms each. The complete explicit solutions appear in Appendix B. While doing optimization calculations it was noticed that a large number of the terms in the solutions for Δx^2 and Δy^2 were much smaller than the rest (by 10 orders of magnitude) when using nominal Small Recirculator quantities. These terms can therefore be dropped without loss of accuracy and the simplified explicit expressions are shown here:

$$\begin{aligned}
\Delta x^2 = {} & (4\beta^4(\Delta x'y')^2(\Delta xx')^2 - 8\beta^4(\Delta x'y')^3\Delta xy + 8\beta^4\Delta x'y(\Delta x'y')^2\Delta xy' && (6.7)\\
& + 8\beta^4\Delta x'y'\Delta x'^2\Delta xx'\Delta xy' - 4\beta^4(\Delta x'^2)^2(\Delta xy')^2 - 8\beta^4\Delta x'^2(\Delta xx')^2\Delta y'^2 && (6.8)\\
& + 8\beta^4\Delta x'y'\Delta x'^2\Delta xy\Delta y'^2 - 8\beta^4\Delta x'y\Delta x'^2\Delta xy'\Delta y'^2 + 4\beta^4(\Delta x'y)^2(\Delta y'^2)^2 && (6.9)\\
& - 8\beta^4\Delta x'y\Delta x'y'\Delta y'^2\Delta yy' + 4\beta^4(\Delta x'y')^2(\Delta yy'^2) + 0.5\beta^2(\Delta x'y')^2\epsilon_{ng}^2 && (6.10)\\
& - 0.5\beta^2\Delta x'^2\Delta y'^2\epsilon_{ng}^2)/(\beta^4\Delta x'^2(8(\Delta x'y')^2 - 8\Delta x'^2\Delta y'^2)) && (6.11)
\end{aligned}$$

$$\begin{aligned}
\Delta y^2 = {} & 0.5(-\beta^2(256\beta^2(\Delta x'y')^2(\Delta xx')^2 - 512\beta^2(\Delta x'y')^3\Delta xy + 512\beta^2\Delta x'y(\Delta x'y')^2\Delta xy' && (6.12)\\
& - 512\beta^2\Delta x'y'\Delta x'^2\Delta xx'\Delta xy' + 256\beta^2(\Delta x'^2)^2(\Delta xy')^2 + 512\beta^2\Delta x'y'\Delta x'^2\Delta xy\Delta y'^2 && (6.13)\\
& - 512\beta^2\Delta x'y\Delta x'^2\Delta xy'\Delta y'^2 - 256\beta^2(\Delta x'y)^2(\Delta y'^2)^2 + 512\beta^2\Delta x'y\Delta x'y'\Delta y'^2\Delta yy' && (6.14)\\
& + 256\beta^2(\Delta x'y')^2(\Delta yy'^2) - 512\beta^2\Delta x'^2\Delta y'^2(\Delta yy'^2) + 32(\Delta x'y')^2\epsilon_{ng}^2 && (6.15)\\
& - 32\Delta x'^2\Delta y'^2\epsilon_{ng}^2))/(\beta^4\Delta y'^2(-256(\Delta x'y')^2 + 256\Delta x'^2\Delta y'^2)) && (6.16)
\end{aligned}$$

To verify the accuracy of these expressions the reduced set of eight moment evolution equations were used to simulate the Small Recirculator beam drifting through a

lattice of perfectly aligned magnetic quadrupoles. Two calculations were performed. The first calculation integrated the full set of ten moment equations. The second calculation integrated the reduced set of eight moment with the simplified forms of Δx^2 and Δy^2 shown above. Figure 6.7 shows the difference between these calculations and that the maximum error is <0.1%.

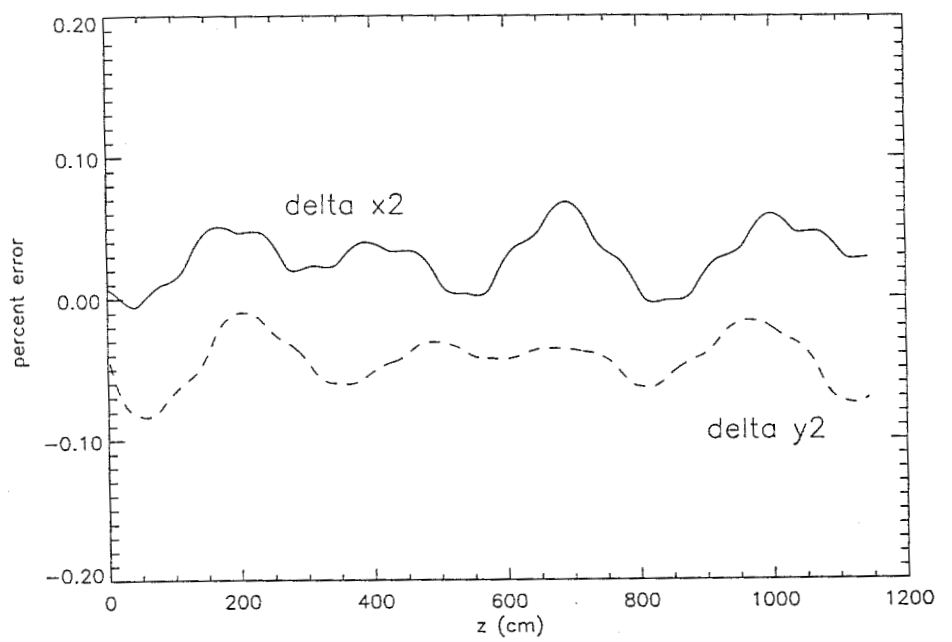


Figure 6.7: Verification of simplified Δx^2 and Δy^2 solutions. Each line is the percent difference between the calculated moments with the full set of 10 moment evolution equations and with the reduced set of 8 equations incorporating the simplified explicit solutions for Δx^2 and Δy^2.

6.3.2 Determining the Optimum Correction System Parameters

Modifying Barnard's MOMENT code to calculate the optimized values of the correction quadrupole rotation angle and field strength presented a challenge due to the moment evolution equations' nonlinear dependence on the two quantities (angle

and field strength) to be optimized. An elegant and efficient solution was found by adapting Spall's "Simultaneous Perturbation Stochastic Approximation" (SPSA) algorithm to the problem.[44]

The SPSA algorithm has been applied successfully to many problems where the equations governing the system are unknown and therefore obtaining the gradient of the objective function with respect to the parameters being optimized is not possible.[45,46] These so-called model-free applications are treatable with SPSA as it is based on a highly efficient gradient approximation requiring only two objective function measurements regardless of the dimensionality of the gradient vector. In the beam correction system the parameters to be optimized are the rotation angle and magnetic field strengths of the four quadrupole magnets with the aim of adjusting the ten second moments to remove the effect of previous rotation errors. The objective function I have defined is the root sum square of the difference between the moments of the beam emerging from the correction system and the desired moments:

$$
\begin{aligned}
&L(\underline{M}) = \\
&\quad ((\Delta x_c^2 - \Delta x_d^2)^2 + (\Delta x x'_c - \Delta x x'_d)^2 + (\Delta x'^2_c - \Delta x'^2_d)^2 \\
&\quad + (\Delta y_c^2 - \Delta y_d^2)^2 + (\Delta y y'_c - \Delta y y'_d)^2 + (\Delta y'^2_c - \Delta y'^2_d)^2 \\
&\quad + (\Delta x y_c - \Delta x y_d)^2 + (\Delta x' y_c - \Delta x' y_d)^2 + (\Delta x y'_c - \Delta x y'_d)^2 + (\Delta x' y'_c - \Delta x' y'_d)^2)^{\frac{1}{2}}
\end{aligned}
$$

$$
\underline{M} = \left[\Delta x^2, \Delta x x', \Delta x'^2, \Delta y^2, \Delta y y', \Delta y'^2, \Delta x y, \Delta x' y, \Delta x y', \Delta x' y'\right] \tag{6.17}
$$

where, for example, Δx_c^2 is the calculated moment through the beam correction system using the moment evolution equations, and Δx_d^2 is the desired moment at the correction system output based on a calculation with a perfectly aligned transport lattice. Although in the set of moment evolution equations, the gradients of the objective function, $L(\underline{M})$ with respect to rotation angle and field strength could be found and applied to a standard finite difference scheme, the nonlinearity of the equations makes this difficult. The SPSA algorithm allows one to sidestep this issue with what has appeared to be no loss of accuracy and a definite improvement in computational efficiency. SPSA requires only two measurements of the objective function rather

than the 16 of the standard finite differencing scheme.[44] This is quite a saving as each measurement requires integrating the moment evolution equations through the beam correction system which is quite demanding using the reduced set of equations required for a four magnet system. The goal is to minimize the objective function resulting in the best least squares fit of the desired moment vector $\underline{M}$. The SPSA algorithm works by iterating from an initial guess of the optimal $\underline{M}$, where the iteration process depends on the "simultaneous perturbation" approximation to the gradient

$$g(\underline{M}) = \frac{\partial L(\underline{M})}{\partial \underline{\theta}} \tag{6.18}$$

where $\underline{\theta}$ is the 8-dimensional vector of quadrupole rotation angles and field strengths. It is assumed that $L(\underline{M})$ is a differentiable function of $\underline{\theta}$ and the minimum point $\underline{\theta}^*$ corresponds to the zero point of the gradient, i.e., $\underline{\theta}^*$ is such that

$$g(\underline{\theta}^*) = \frac{\partial L(\underline{M})}{\partial \underline{\theta}} \mid_{\underline{\theta}=\underline{\theta}^*} = 0. \tag{6.19}$$

As a consequence of the basic recursive form of the algorithm there is generally not a risk of converging to a maximum or a saddle point. Nevertheless, one is able to check for convergence to a local minimum by examining the behavior of the moments in the lattice past the correction system, particularly whether the $x - y$ cross moments remain near zero.

6.3.3 Implementing the Optimizing Algorithm

The SPSA algorithm iterates through the following steps:

1. Pick an initial guess for $\underline{\theta}$ which for lack of any better information and the desire for the least adjustment for engineering considerations is zero quadrupole rotation angle and nominal magnetic field strength.

2. Set the counter index, $k = 0$ and compute the SPSA gain sequences

$$a_k = a(A + k + 1)^{-\alpha} \qquad c_k = c(k + 1)^{-\gamma} \tag{6.20}$$

Spall states that $\alpha = 0.602$ and $\gamma = 0.101$ are effective and theoretically valid and also gives guidelines for the selection of a, A, and c [footnote].

3. Generate by Monte Carlo method a p-dimensional random vector, Δ_k (p being the number of adjustable parameters, here, 8) where each component is independently generated from a zero-mean probability distribution satisfying the conditions of Spall.[44] The Bernoulli ± 1 distribution is a valid choice with probability $\frac{1}{2}$ for each ± 1.

4. Obtain two measurements of the loss function, $L(\underline{M})$ based on a simultaneous perturbation around the current $\underline{\theta}_k$, $y(\underline{\theta}_k + c_k\Delta_k)$ and $y(\underline{\theta}_k - c_k\Delta_k)$. These measurements are computed by first adjusting the quadrupole parameters by $+c_k\Delta_k$ and integrating the input moments through the beam correction system and computing the root sum square of the difference between the output moments and the desired moments. This process is repeated with the beginning quadrupole values adjusted by $-c_k\Delta_k$. This is where the advantage of needing only two measurements rather 16 is realized as integration of the reduced set of moment evolution equations is computationally demanding.

5. Using the two measurements of the objective function obtained in the last step generate the simultaneous perturbation approximation to the 8-dimensional gradient,

$$g_k(\underline{\theta}_k) = \begin{bmatrix} \frac{L(\underline{\theta}_k + c_k\underline{\Delta}_k) - L(\underline{\theta}_k - c_k\underline{\Delta}_k)}{2c_k\Delta_{k1}} \\ \vdots \\ \frac{L(\underline{\theta}_k + c_k\underline{\Delta}_k) - L(\underline{\theta}_k - c_k\underline{\Delta}_k)}{2c_k\Delta_{k8}} \end{bmatrix} \tag{6.21}$$

where Δ_{ki} is the ith component, (± 1), of the $\underline{\Delta}_k$ vector. Note the denominators change (stochastically) while the numerators are all the same, making possible only the two measurements to produce gradient approximations.

6. Update the adjustable quadrupole parameters using the standard stochastic approximation form

$$\underline{\theta}_{k+1} = \underline{\theta}_k - a_k g_k(\underline{\theta}_k) \tag{6.22}$$

7. The algorithm is terminated if there is little change in several successive iterations. If this condition is not met, the iteration counter is updated by one and the iteration is repeated from step 2. The last iterate, $\underline{\theta}_k$ is the estimate of the optimum $\underline{\theta}^*$ which corrects the beam for previous quadrupole rotation errors such that the beam is upright with zero angular momentum.

I modified Barnard's Moment code to include optimization iteration loops implementing the SPSA algorithm. This code CORRECTOR (see Appendix C) models the beam correction system previously mentioned and shown in Figure 6.6. The results of an optimization run are shown in Figures 6.8 and Table 6.1. The input moments to the beam correction system resulted from a calculation of a recirculator transport through 562 half lattice periods with maximum rotation angle error of 0.5°. The desired output moments were generated by the REDUCEDAG code, a modification of the "MOMENT" code modeling a perfectly aligned lattice which can be seen by the zero values of the $x - y$ cross moments in Table 6.1. Although the SPSA algorithm remained noisy, the desirable settings were those that produced the smallest difference between the calculated and desired beam moments. From Figure 6.8 one can see that the best parameters were not always obtained on the last iteration.

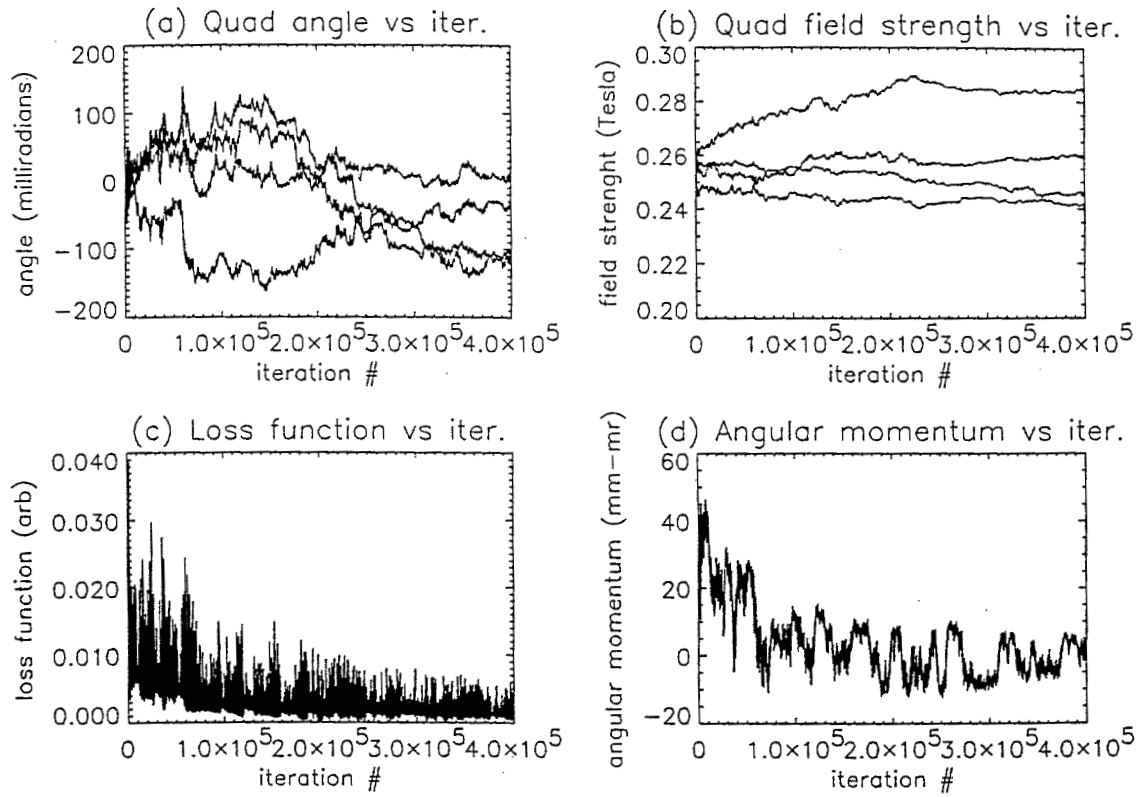


Figure 6.8: Beam correction system optimization code results plots. The best parameters occurred on iteration number 245,753 of 400,000.

Table 6.1: Results of the SPSA optimization run to determine beam correction system settings

Least Loss Function Calculated		$1.629\text{x}10^{-4}$	
Initial Loss Function		0.1562	
Angular Momentum		$3.740\text{x}10^{-4}$	
Initial Angular Momentum		0.126	
Nominal Quad Field (T)	0.25258	Nominal Quad Angle (°)	0
Quad Field 1 (T)	0.25692	Quad Angle 1 (°)	-3.956
Quad Field 2 (T)	0.24212	Quad Angle 2 (°)	0.621
Quad Field 3 (T)	0.25033	Quad Angle 3 (°)	-2.112
Quad Field 4 (T)	0.28685	Quad Angle 4 (°)	-3.651
Calculated Moments		Desired Moments	
Δx^2 (mm^2)	54.54	Δx^2 (mm^2)	54.54
$\Delta xx'$ (mm-mr)	102.28	$\Delta xx'$ (mm-mr)	101.66
$\Delta x'^2$ (mr^2)	179.35	$\Delta x'^2$ (mr^2)	177.19
Δy^2 (mm^2)	15.96	Δy^2 (mm^2)	15.96
$\Delta yy'$ (mm-mr)	-31.92	$\Delta yy'$ (mm-mr)	-30.51
$\Delta y'^2$ (mr^2)	66.27	$\Delta y'^2$ (mr^2)	60.78
Δxy (mm^2)	$-4.13\text{x}10^{-3}$	Δxy (mm^2)	0
$\Delta x'y$ (mm-mr)	$4.84\text{x}10^{-4}$	$\Delta x'y$ (mm-mr)	0
$\Delta xy'$ (mm-mr)	$1.92\text{x}10^{-2}$	$\Delta xy'$ (mm-mr)	0
$\Delta x'y'$ (mr^2)	$2.02\text{x}10^{-2}$	$\Delta x'y'$ (mr^2)	0
SPSA Parameters			
Maximum iterations	$4\text{x}10^5$	Gain coefficient, A	$2\text{x}10^4$
Gain coefficient, a	0.06	Gain coefficient, b	0.0033
Gain coefficient, c	$5\text{x}10^{-6}$	Gain coefficient, d	$5\text{x}10^{-6}$

The optimized values of the four rotation angles and field strengths are used in the simulation shown in Figures 6.9– 6.10. A system of 4 quadrupoles modeling the beam correction system accepts a beam whose input moments are obtained from a lattice of 562 quadrupoles with random errors up to ±0.5°. With the optimized settings of Table 6.1 the four quadrupoles deliver a beam that is "upright" and not rotating and whose other moments are nearly returned to the desired values.

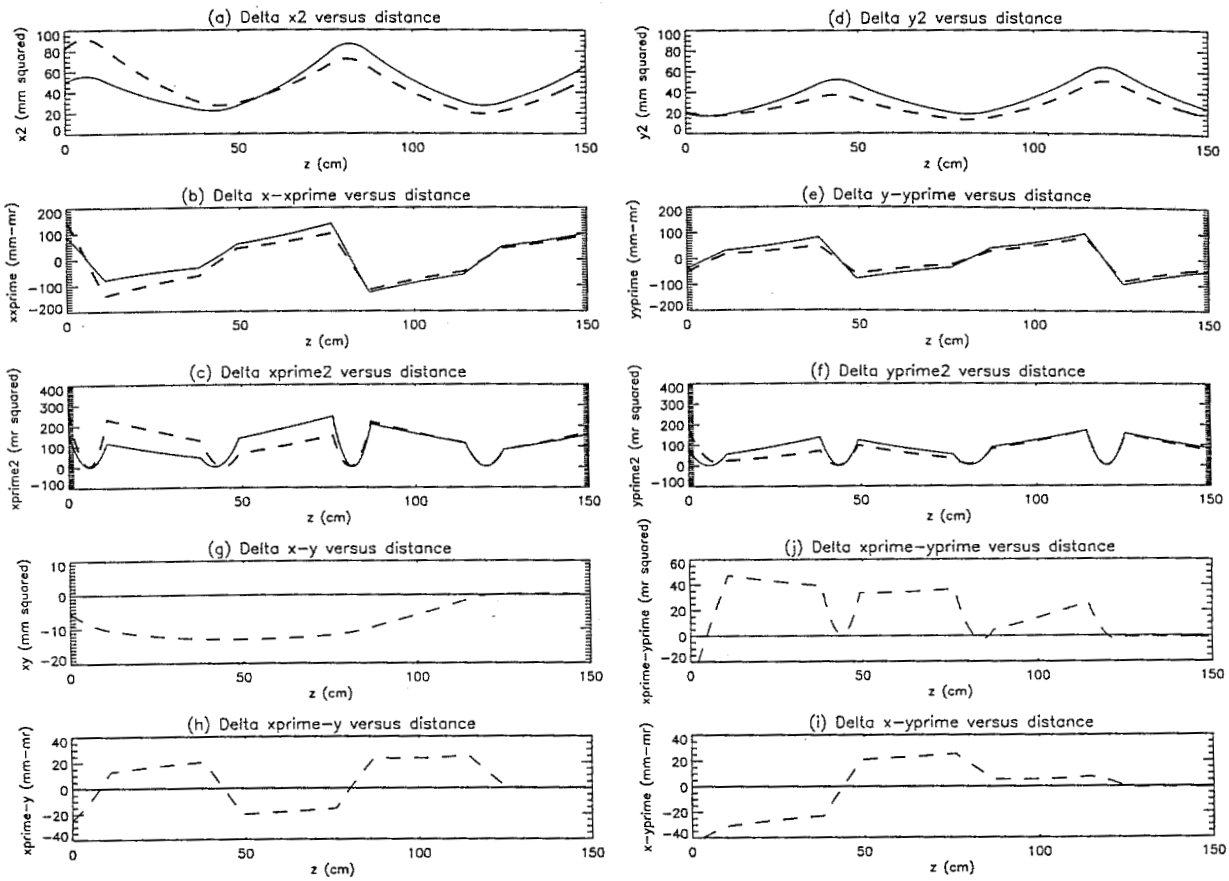


Figure 6.9: Correction of random quadrupole rotation errors of up to $\pm 0.5°$ from 562 half lattice periods of a Small Recirculator lattice. The beam correction system is placed between z=0 and 152 cm. The dashed lines show the moments as the beam is in the correction system being "corrected" to the desired values. The solid lines are the moments from an aligned lattice and are shown to depict the convergence of the misaligned beam to the desired moment values by the end of the correction system at z=152 cm.

Thus, looking at these results it can be concluded that such a correction system

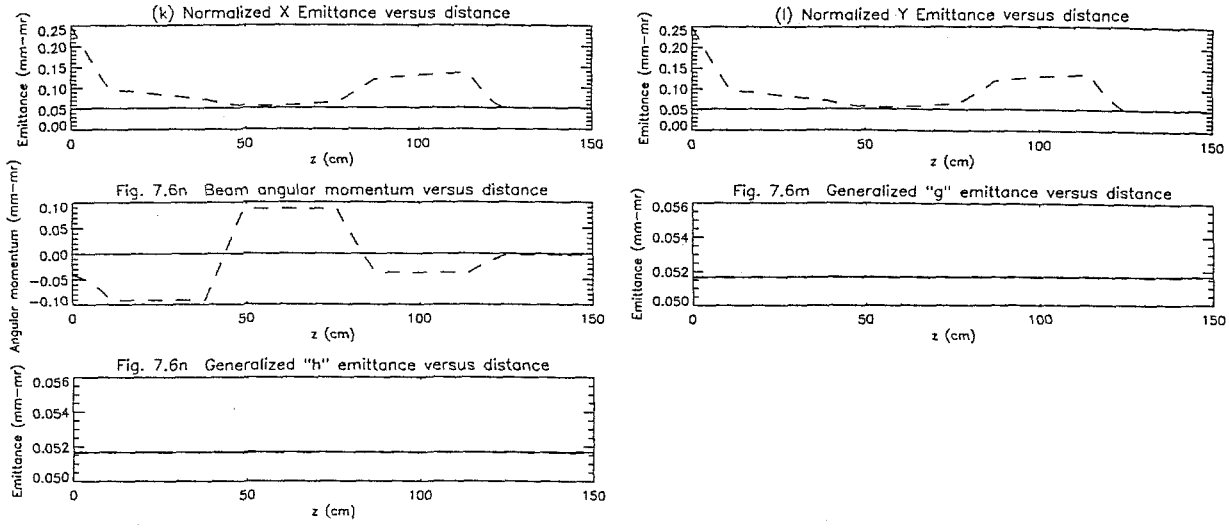


Figure 6.10: Correction of random quadrupole rotation errors of up to $\pm 0.5°$ from 562 half lattice periods of a Small Recirculator lattice. The beam correction system is placed between z=0 and 152 cm. The dashed lines show the moments as the beam is in the correction system being "corrected" to the desired values. The solid lines are the moments from an aligned lattice and are shown to depict the convergence of the misaligned beam to the desired moment values by the end of the correction system at z=152 cm.

placed prior to the final focusing element can have a positive impact on lessening the rotational alignment tolerance on a driver's quadrupole focusing elements. Whether or not the degree of precision required for the field strength and rotation angle of the four correction quadrupoles can be attained is not determined here. Nevertheless, for any kind of system that will correct for rotation errors, the $x - y$ cross moments will have to be measured. The GBI is well suited to do this. A test of rudimentary beam correction was carried out on the Small Recirculator using single quadrupoles and the results of these tests as measured by the GBI are discussed in Chapter 7.

Chapter 7

Experimental Results

A complete set of measurements were made on the Small Recirculator beam at the end of the 45° bend section with the Gated Beam Imager on July 22, 1997. Good results were obtained and show the effectiveness of the GBI as a diagnostic tool when combined with appropriate image analysis software. During the measurements, the beam energy was 80 keV with an initial 4 μs long pulse length which lengthened as the beam traveled through the accelerator since no axial confinement was possible without the modulators installed. The data run consisted of gathering 100 ns long gated images stepped through successive beam shots to time-resolve the beam into approximately 40 slices. Additionally, 1 μs long gated images were obtained during the flattop of the beam. Images with upstream magnetic quadrupole rotation errors were taken to study the effect on the beam emittance as discussed in Chapter 6. The rudimentary beam correction system of counter-rotating one quadrupole at the appropriate distance from a known quadrupole rotation error was also studied to determine the effect on the beam. Appendix D lists the important equipment and software settings used to obtain the data images and information about each image.

As the images were obtained during the initial testing phase of the Small Recirculator 45° bend section it was necessary to determine the optimum voltage settings for the bending dipole plates in each half lattice period for a drifting 80 keV beam such that the beam centroid would be on centerline when intercepted by the GBI. This was

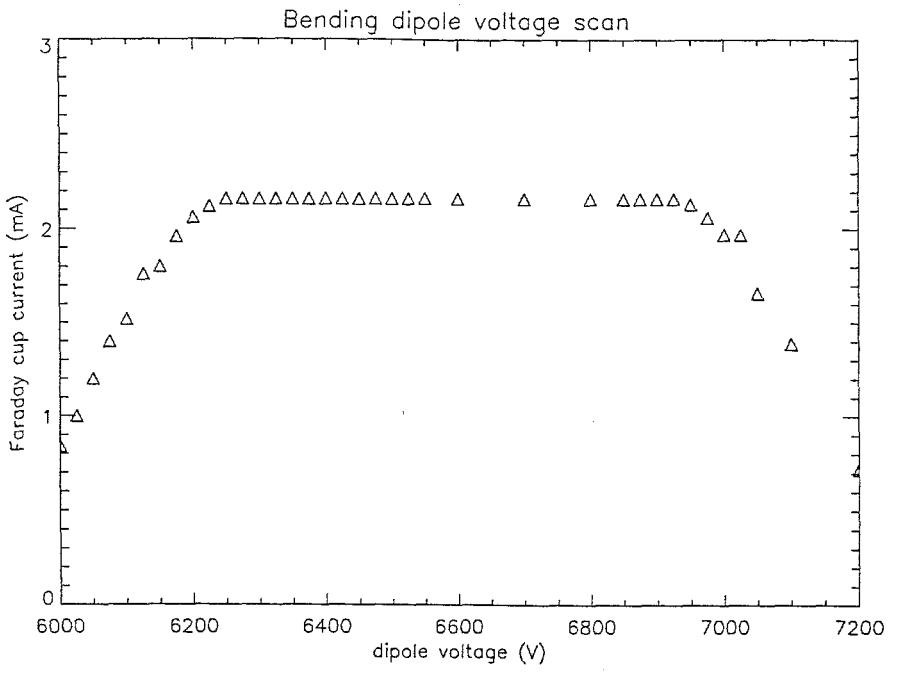


Figure 7.1: Faraday cup current vs. bending dipole voltage scan for determining the optimum dipole voltage to transport the 80 keV around the 45° bend.

done prior to installing the GBI by inserting a Faraday cup in the multi-diagnostic chamber such that it was centered on the beamline at the outlet of the bend. The beam current into the cup was measured while changing the bending dipole plate voltage in the half lattice periods and then plotted in Figure 7.1. The voltage values of the flattop corresponded to a beam current of 2.06 mA indicating full beam transport around the 45° bend. The 4 cm diameter of the Faraday cup is larger than the nominal diameter (2.6 cm) of the ion beam resulting in the wide plateau in the curve indicating that the cup intercepted all of the beam over a wide dipole voltage range. A first order estimate of the voltage that put the the beam on the centerline was made by taking the midpoint between the "shoulders" of the curve. This voltage of 6570 V was applied to the bending dipoles in the subsequent GBI measurements and verified as the proper bending voltage by the images obtained.

7.1 GBI Setup and Calibration

After determining the bending dipole voltage setting, the Faraday cup was withdrawn and the pepperpot and MCP detector of the GBI inserted in their centering mount within the multi-diagnostic chamber and the electrical connections supplying the detector intensifier and proximity focusing voltages made up. The camera was then positioned and focused to give a sharp image of the fiducial marks and scale on the face of the detector. Figure 7.2 shows the camera in position observing the detector through a window in the diagnostic chamber. A calibration image (Figure 7.3) of a scale mounted on the detector was then taken with the camera in place. Analysis of the calibration image with the measurement tools of the IPLAB imaging software determined the spatial resolution of the GBI to be 54 microns per CCD pixel and the field of view to be 2.07 by 3.11 cm. This resolution was used as an input to the IDL Pepperpot analysis code to determine beamlet spot sizes in the subsequent data images and as an input for estimating error bars on the analysis.

7.2 Observing the Effect of Axial Beam Expansion

Measurements at the end of the 45° bend captured a total of 56 images ranging from 4 μs long images to gated 100 ns long images. A time-of-flight calculation predicted the beam head would arrive at the detector at 11.9 μs after the source pulser was triggered. The beam head was measured arriving at the detector after 11.7 μs with beam tail arriving after 16.4 μs. Thus, the initial beam length of 4μs had expanded to 4.7μs long. This axial expansion of the beam was not unexpected as no axial confinement fields were applied. In fact, the time-resolved beam images can be viewed as a qualitative measure of the difference in ion energy axially along the beam as seen in the 3 images of Figure 7.4. These images also confirm that the timing signals gating the MCP "ON" and "OFF"are correct and discrete beam slices are being imaged by the GBI. The detector intercepts the head of the beam as it swings wide around the bend as the head is accelerated by space-charge forces within

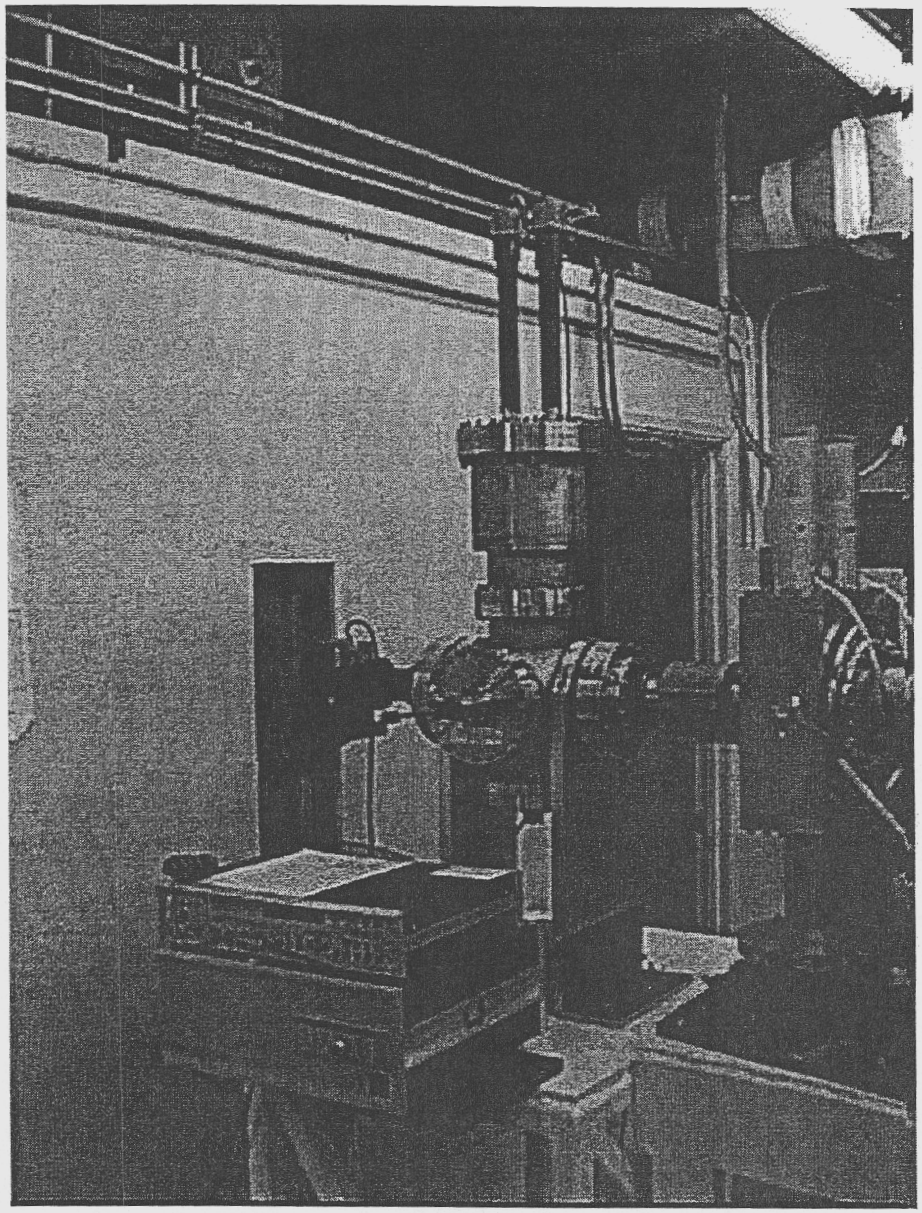


Figure 7.2: Gated Beam Imager in position for data collection at the end of the 45° bend section

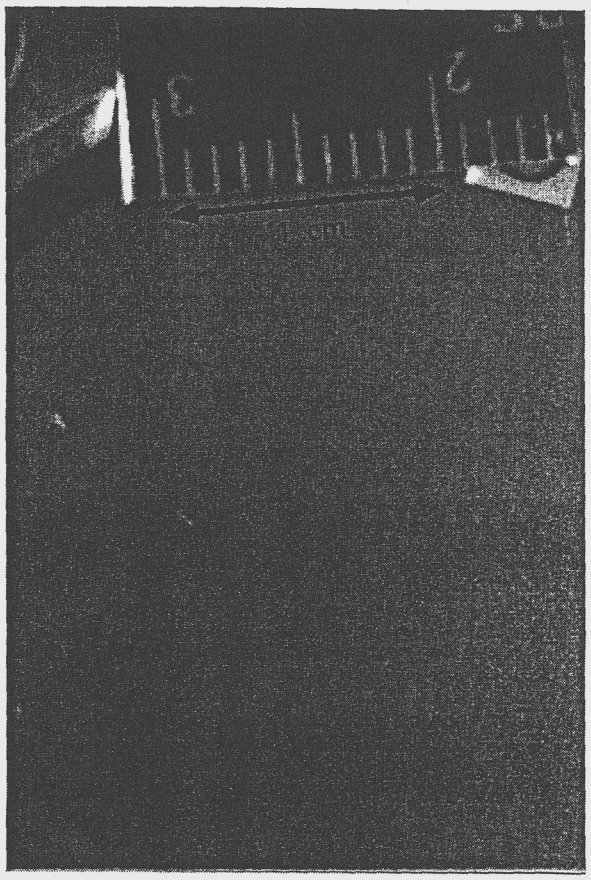
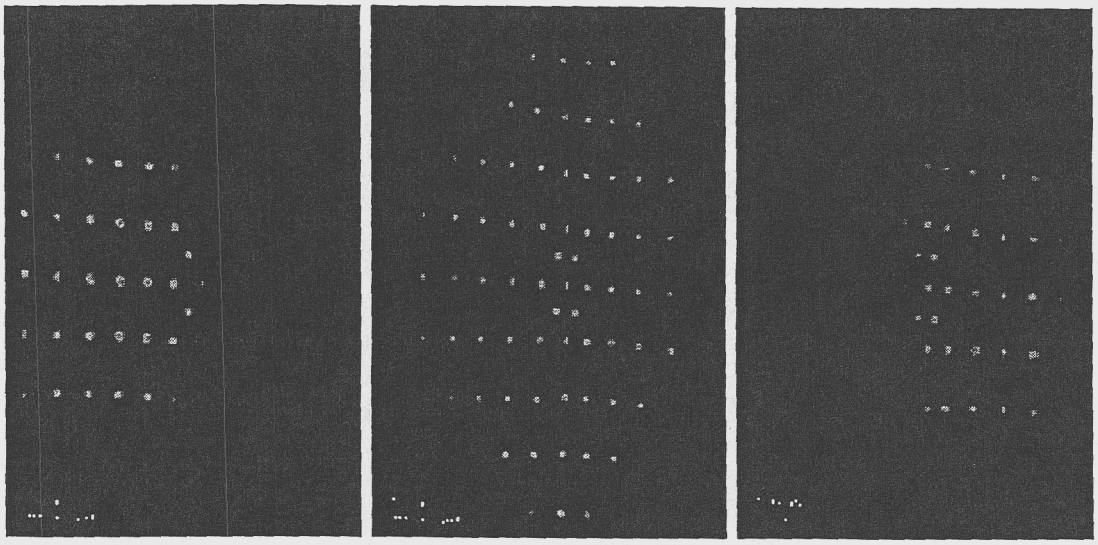


Figure 7.3: Calibration image taken to determine the CCD camera pixel size

the beam (Figure 7.4a). The center image (Figure 7.4b), at 14.0 μs representing the beam flattop is intercepted at the center of the beamline as it experiences little net longitudinal space-charge force. In the third image (Figure 7.4c), at 15.6 μs, the ions have been retarded and the beam swings to the inside in the bending dipole field. A "beam movie" was made of the series of 100 ns slices that graphically illustrates the acceleration of the beam head, retardation of the beam tail and the stability of the flattop region of the beam.



(a) (b) (c)

Figure 7.4: 100 ns images at the end of 45° without beam axial confinement showing: **a.** the beam head swinging wide due to space-charge acceleration, **b.** the beam flattop on center, **c.** the beam tail swinging inside the bend due to space-charge retardation.

7.3 Data Analysis and Results at the 45° Bend Section

A total of 48, 100 ns long gated images were obtained each delayed by 100 ns from the previous image giving a time history of the beam. Additionally, three 1 μs long gated images from the flattop of the beam were captured. These images were taken after times-of-flight of 12.2, 13.2 and 14.2 μs to characterize the center of the beam shown as the flattop in the oscilloscope trace of the beam into the Faraday cup (Figure 7.5). Each data image was modified prior to analysis as described in Chapter 5 to account for dark charge accumulation and non-uniform response of the camera CCD and the MCP detector.

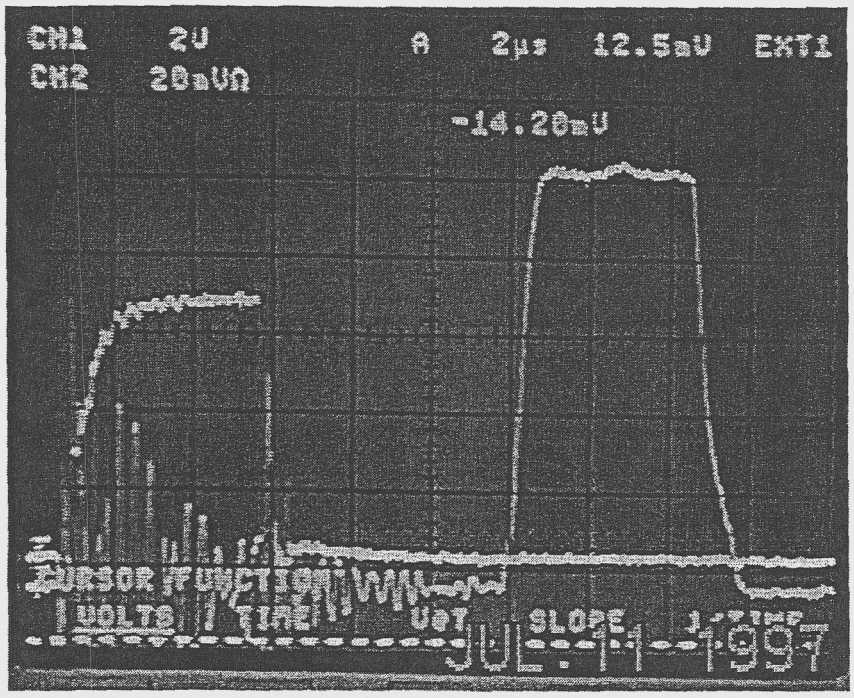


Figure 7.5: Oscilloscope traces of the source pulser (left) and the Faraday cup signal of the beam at the end of the 45° bend section. The time scale is 2 μs per division.

7.3.1 Comparison of GBI and Slit Scanner Measurements

Using the IDL Pepperpot analysis code, each image of the beam was analyzed. The emittance and second moments of the beam during each time slice was calculated in order to obtain a time history of the beam. The calculation was carried out as described in Chapter 4. The normalized emittance versus time along the beam with the time origin referenced to the beam head is shown in Figure 7.6. The value of the emittance in the y-direction obtained with a slit scanner is also indicated on Figure 7.6 with the triangle symbol. There is essential agreement between the two methods. Figure 7.7 shows the time history of the ten second moments of the beam. These figures represent a graphical distillation of the 4-D and time phase space map of the beam. The flattop numerical values characterizing the beam are shown in Table 7.1, where the symbol "Δ" is the second moment minus the contribution of a non-zero first moment, following the convention of Chapter 4.

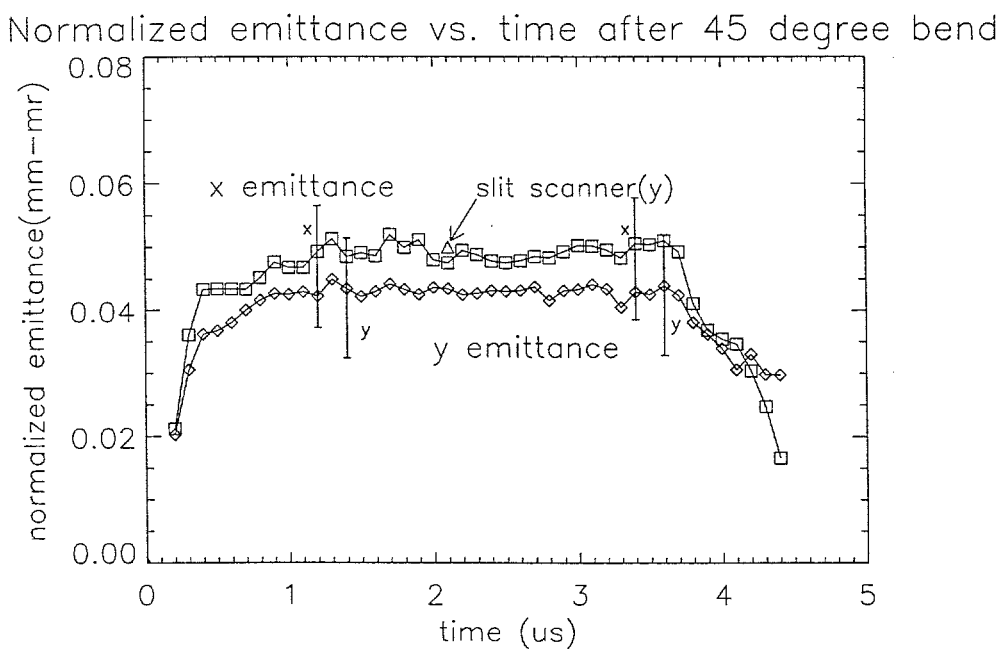


Figure 7.6: Normalized emittance vs. time from the GBI. A single data point representing the slit scanner flattop emittance measurement in the y-direction is shown with the triangle symbol.

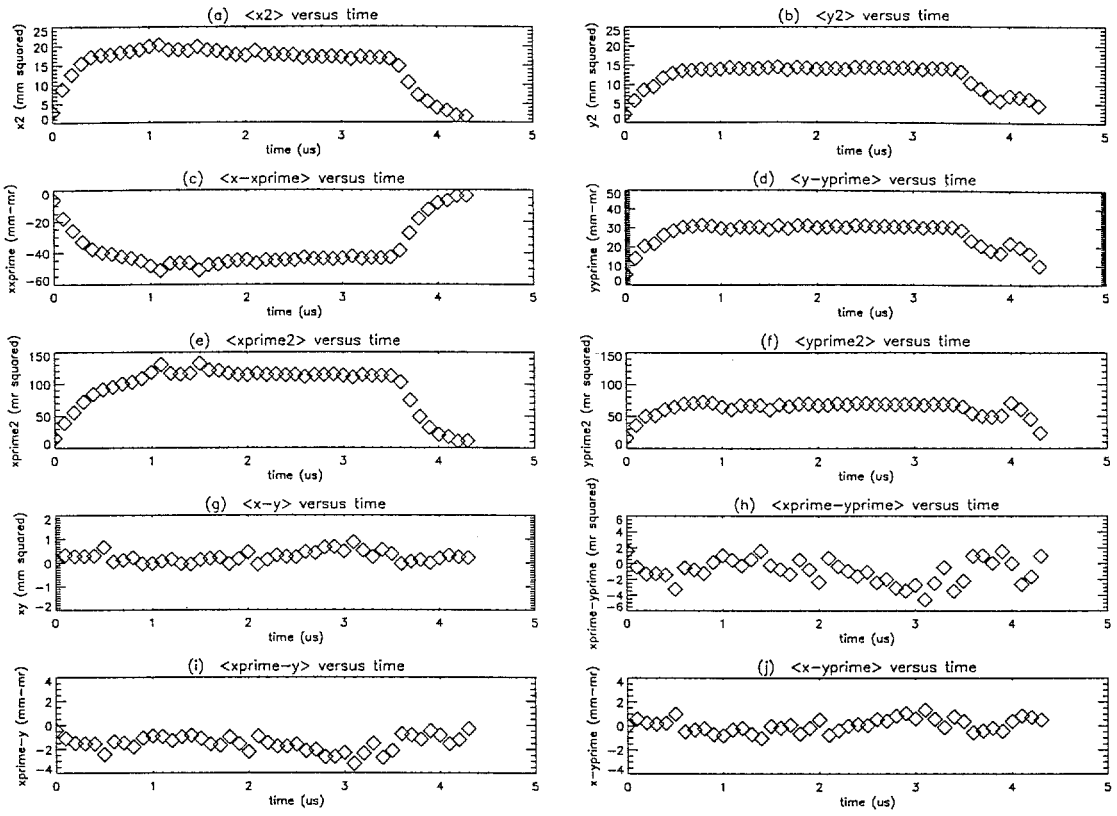


Figure 7.7: Beam ion distribution second moments vs. time

Table 7.1: Characterization of the Small Recirculator beam after the 45° bend section using the GBI.

$< x >$ (mm)	0.67	$< y >$ (mm)	0.39
$< x' >$ (mr)	1.06	$< y' >$ (mr)	-3.35
Δx^2 (mm^2)	34.6	Δy^2 (mm^2)	27.4
$\Delta xx'$ (mm-mr)	-87.3	$\Delta yy'$ (mm-mr)	60.0
$\Delta x'^2$ (mr^2)	223.8	$\Delta y'^2$ (mr^2)	135.3
Δxy (mm^2)	0.736	$\Delta x'y'$ (mr^2)	-3.881
$\Delta x'y$ (mm-mr)	-3.814	$\Delta xy'$ (mm-mr)	0.773
ϵ_x (rms edge) (mm-mr)	22.73	ϵ_{nx} (mm-mr)	0.0484
ϵ_y (rms edge) (mm-mr)	20.06	ϵ_{ny} (mm-mr)	0.0427
x envelope (mm)	11.8	y envelope (mm)	10.5
x' envelope (mr)	-29.7	y' envelope (mr)	22.9
angular momentum (mm-mr)	4.587		

7.3.2 Error Analysis

The error bars on the emittance data points in Figure 7.6 were calculated based on known or estimated tolerances in the physical size of GBI dimensions and measurement errors that have a substantial effect on the moment calculations. Each worst case tolerance value was input into the Pepperpot analysis code. The effect on the calculation was assumed independent of the effect of the other tolerances. Like sign effects were summed to generate the high and low limits of the error bars. The tolerances that were accounted for and their contributions to the emittance are summarized in Table 7.2. Additionally the Peppperpot analysis code was run with all of the like effect tolerances applied together to calculate the result when they are allowed to interact.

Proper interpretation of the error bars is important to drawing a conclusion as to the relative magnitudes of the x and y emittances. Those errors that tend to cause the emittance to be greater or smaller than the calculated value (where the

data point falls) track each other in both the x and y directions. The conclusion drawn from this is that indeed the x emittance which is "in-plane" with the bend will remain larger than the y emittance and both values will track together up or down their respective error bars. As can be seen in Table 7.2 the calculation of the emittance is most sensitive to errors in the alignment about the z-axis of the grid of holes in the pepperpot. A misalignment of the pepperpot results in an error in the calculation of the angle from the position of the hole to the position of the beamlet spot. Additionally this error is not constant but increases with the distance from the beam axis. Empirically, it was seen with the images obtained at the 45° bend, the error introduced did not contribute equally to the error bar in both the x and y directions. An error resulting from the misalignment of the pepperpot is however applied equally to all calculations from image to image within a series where the GBI has not been moved.

Table 7.2: Pepperpot data analysis sensitivity to measurement errors, and instrument manufacturing and alignment tolerances. (Emittance units are mm-mr)

Maximum Error↓	ϵ_{nx}	$\Delta\epsilon_{nx}$	ϵ_{ny}	$\Delta\epsilon_{ny}$
Nominal Emittances	0.04838	0	0.04269	0
Hole Size + 5%	0.0409	-0.00029	0.04237	-0.00032
Hole Size - 5%	0.0487	0.00031	0.04301	0.00032
mask rot/cam. angle + 0.5°	0.04768	-0.00070	0.04461	0.00195
mask rot/cam. angle - 0.5°	0.04952	0.00114	0.04312	0.00043
w/o intensity weighting	0.05042	0.00204	0.04474	0.00206
+ 5% pixel calibration error	0.05089	0.00251	0.04496	0.00227
- 5% pixel calibration error	0.04587	-0.00251	0.04042	-0.00227
spot radius meas. error (+27μ)	0.05160	0.00322	0.04619	0.00350
spot radius meas. error (-27μ)	0.04538	-0.00300	0.03928	-0.00341
change beam energy + 1%	0.04799	-0.00039	0.04234	-0.00035
change beam energy - 1%	0.04751	-0.00087	0.04192	-0.00077
space charge + 8.22% spot size	0.04363	-0.00475	0.03838	-0.00431
+ Error bar x emittance			0.00719	
- Error bar x emittance			-0.01212	
+ Error bar y emittance			0.00802	
- Error bar y emittance			-0.01107	

7.3.3 Comparison of GBI Data with WARP Numerical Simulation Code Results

Members of the LLNL HIF project staff have been developing methods to apply the numerical simulation code WARP as a tool both the theorist, and experimentalist can use to resolve questions regarding the performance of the Small Recirculator. Although this effort is in its infancy and the experimental results are meant to verify the correct values of variable initial parameters in the WARP code, it is nevertheless useful as a confidence building measure in the utility of the GBI to compare its measurements to the code results using reasonable values of the input parameters. The 6-D phase space distribution of the beam initially loaded into the WARP code is a constant density particle distribution in physical space (x,y,z) with a semi-gaussian spread in the velocities of the ions in the transverse (v_x,v_y) and axial (v_z) directions. It is the width of the semi-gaussian velocity spread in the axial direction that most influences the growth in transverse emittance caused by the transition from the straight transport section to the bend section. In the code results, the emittance growth occurs most in the "in-plane" direction (x) of the bend and to a lesser extent in the y direction until the perturbation thermalizes spreading the emittance across both directions. Figure 7.8 shows a graphical output from the WARP code with the data points from the GBI placed at their measured values. It can be seen from the sensitivity in the emittance growth that determining the proper initial value of the axial velocity spread is important. The data from the GBI and slit scanners indicates that a narrower spread is appropriate as the emittance in both the x and y directions does not grow as much as predicted using the initial velocity spreads (0.250.11 The envelope of the beam and its slope at the pepperpot location can be easily calculated from the moments measured by the GBI. A module of the WARP code (WARPenv) calculates the beam envelope parameters. Figure 7.9 shows data points calculated from GBI moments for the envelope and envelope slope along with code simulation

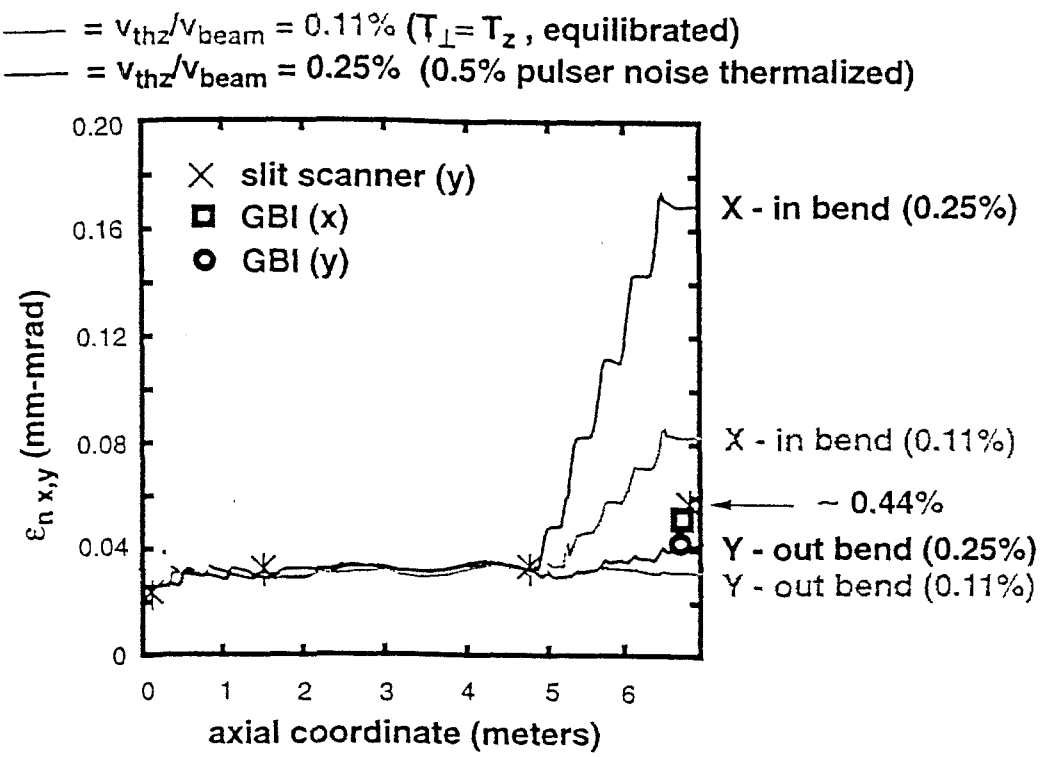


Figure 7.8: Experimental emittance data compared to "WARPxy" simulation.[47]

output. This shows a favorable comparison between the code envelope calculations and the experimental measurements. The envelope dimensions (a, b), and envelope slopes (a', b') were calculated from the second moments using:

$$a = 2\sqrt{<(x-<x>)^2>} \qquad b = 2\sqrt{<(y-<y>)^2>} \tag{7.1}$$

and

$$a' = 2\,\frac{(x-<x>)(x'-<x'>)}{\sqrt{<(x-<x>)^2>}} \qquad b' = 2\,\frac{(y-<y>)(y'-<y'>)}{\sqrt{<(y-<y>)^2>}} \tag{7.2}$$

The slope equations having been obtained by taking the derivative with respect to z of the envelope equations.

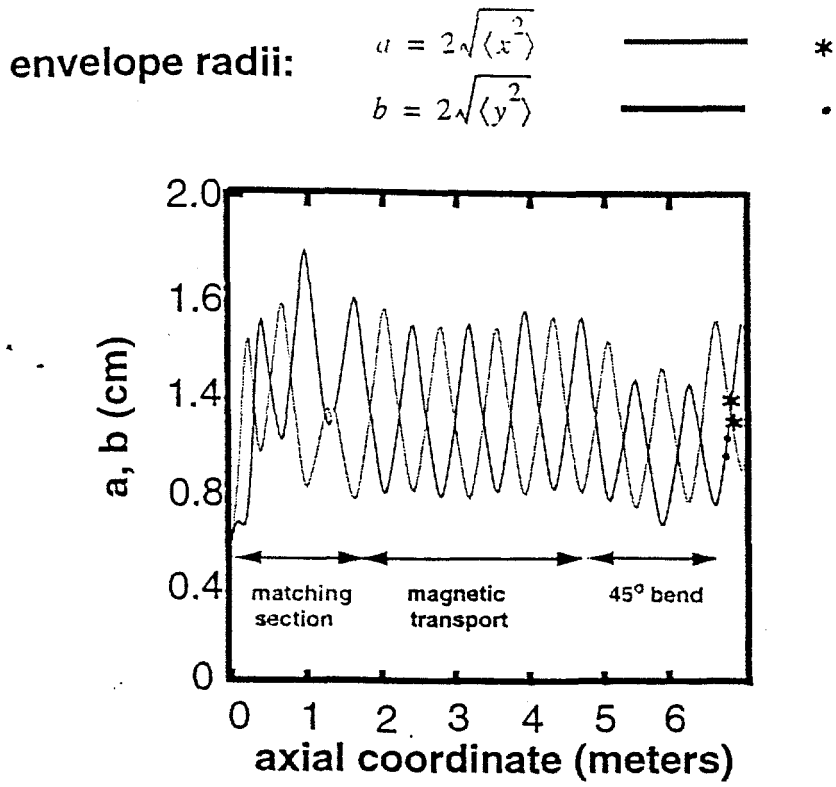


Figure 7.9: Experimental beam envelope data compared with "WARPxy" numerical simulation simulation. The two asterisks show the envelope detected by the GBI at the pepperpot and the detector in the x-direction. Similarly the two dots are the data in the y-direction.[47]

7.4 Observing and Measuring the Effect of Magnetic Quadrupole Rotation Errors and Applying Approximate Corrections

The effect on the beam of magnetic quadrupole rotation errors discussed in Chapter 6 was tested by imaging the beam with known quadrupole rotation errors applied to the Small Recirculator. The rudimentary correction of the beam by counter-rotation of a single quadrupole of the same field strength at the appropriate position was also tested. The proposed beam correction system of four adjustable quadrupoles

was not tested on the Small Recirculator and the construction of such a system is not yet included in project plans. Nevertheless, the results of the rudimentary tests conducted point to the utility of the more complicated system.

From Chapter 6, empirical simulation studies showed that the effects of a known single rotation error can be largely removed by counter-rotating a single, downstream quadrupole. The proper number of half-lattice periods between the adjusted quadrupoles is given by $360/\sigma_0$ when this quotient is an integer. As the GBI was positioned at the output of magnetic quadrupole 13, testing was accomplished with quadrupole 8 and 13 with 5° rotations. The limitations to the tests conducted included the fact that the actual phase advance of the Small Recirculator is 78° when the effect of the bending dipoles is added to the 72° phase advance of the magnetic quadrupoles alone. Because 360° divided by the actual phase advance is not an integer, but is 4.61, rotating the 8th and 13th quadrupole is a close approximation. The other limitation was the existence of unknown random quadrupole rotation errors that exist in any machine. These unknown errors are most likely a contributor to the non-zero $x - y$ cross moments obtained in the images without any intentional rotations. Nevertheless, inducing known quadrupole errors was instructive in confirming their effect on the beam which exploring the concepts of rotation error tolerances and beam correction. Figure 7.10 shows a series of images taken and analyzed for beam moment characterization. They are all 1 μs long gated images of the flattop of the beam starting at a time-of-flight of 13.2 μs, about 1 μs back from the beam head.

Figure 7.10(a) shows an image of the GBI beamlet spots with only the small existing random quadrupole rotation errors causing coupling between the x and y directions. The calculations of the second moments results in non-zero $x - y$ cross moments as listed in Table 7.3. Looking at the image, what appears like a tilt in the rows and columns of the spots, is more likely a combination of the random rotation errors and a 1.7° rotation of the pepperpot from the "laboratory frame". The tilt in the pattern can also be identified as not solely due to the pepperpot rotation as the angle of the rows with the x-axis is greater than the angle of the columns with the y-

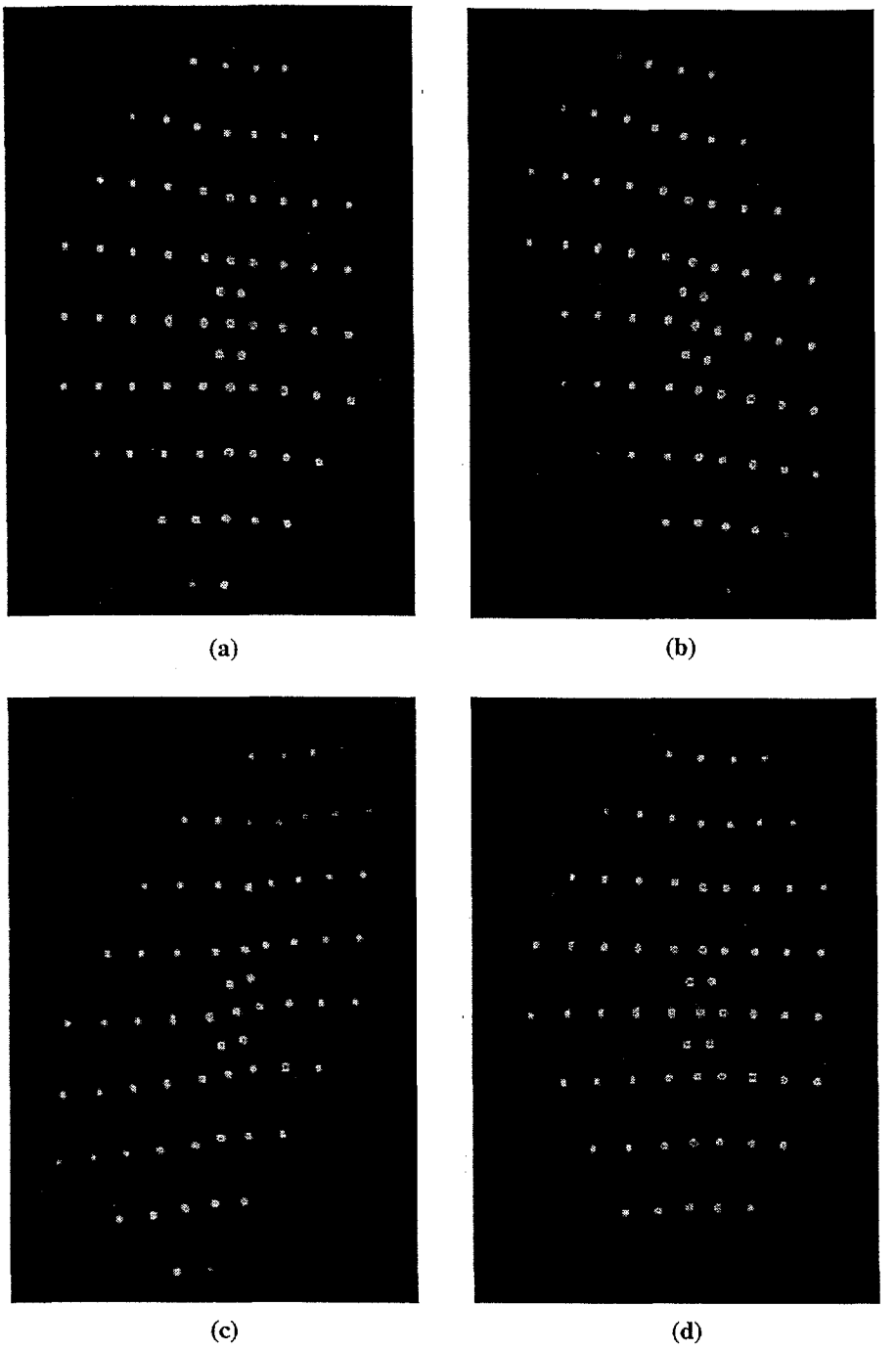


Figure 7.10: GBI images with quadrupole rotation errors. All images are 1 μs long from the beam flattop at 13.2 μs. **a.** Existing experimental rotation errors and other effects **b.** Quad 8 rotated 5° counter-clockwise only **c.** Quad 13 rotated 5° clockwise only **d.** "Corrected" beam with quad 8 at 5° CCW and quad 13 at 5° CW.

axis. With the pepperpot rotation angle removed during the calculation of moments, $x - y$ cross moments remained non-zero, albeit much smaller than when coupling is intentionally induced. Figures 7.11, and 7.12 show the emittance ellipses and cross moment data plots for this case.

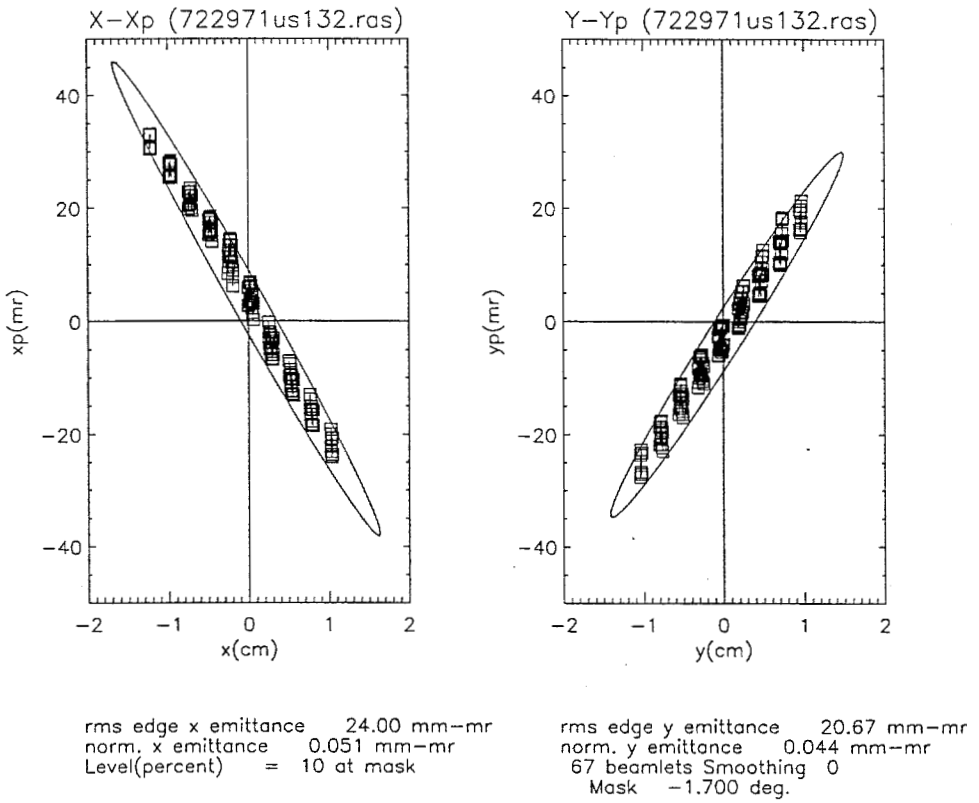


Figure 7.11: Phase space emittance plots with "aligned" quadrupoles.

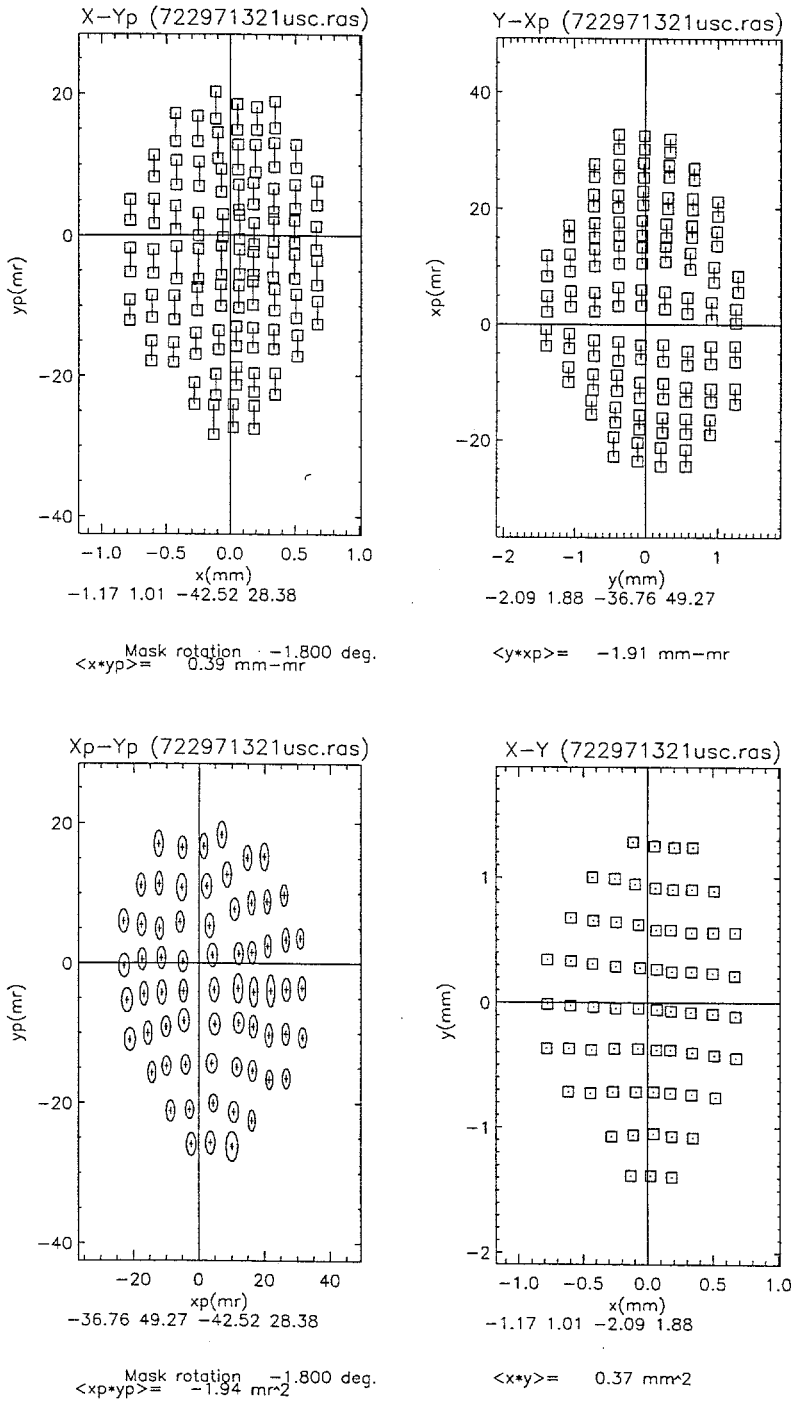


Figure 7.12: Cross moment plots with "aligned" quadrupoles.

Figure 7.10(b) shows the beamlet spot images when quadrupole 8 was rotated 5° counterclockwise. Its moment values are listed in Table 7.2 and emittance ellipses and cross moments data plots are shown in Figures 7.13, and 7.14. Not unexpectantly, the emittance in the x and y directions increases indicating the decrease in focusability of the beam due to their (more) non-zero angular momentum. This image along with Figure 7.10(c) illustrate the ease with which one can observe using the GBI that quadrupole rotation errors have caused an increase in the emittance. A slit scanner will detect the same emittance increase from Figure 7.10(a) to 7.10(b) but with the data available from the measurement the analysis would be hard-pressed to definitively indicate that the cause. As the slit scanner averages the beam information parallel to its slits, detection of the slight envelope changes even for a gross rotation error as I induced here is not likely.

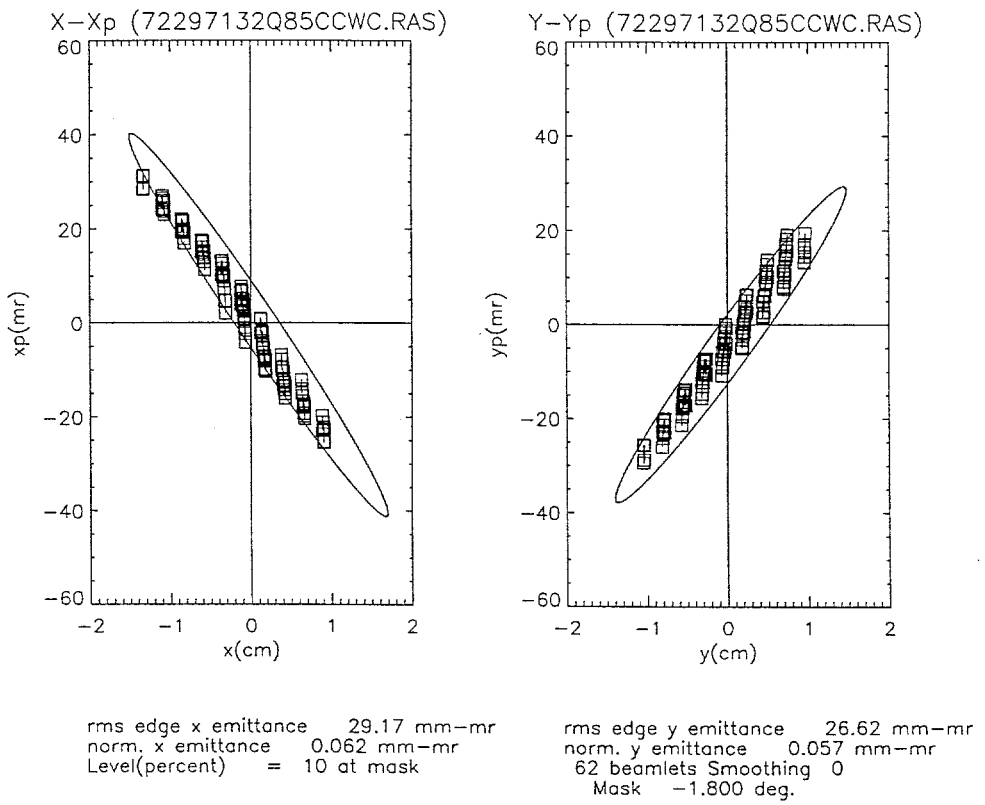


Figure 7.13: Phase space emittance plots with quadrupole 8 rotated 5° counterclockwise.

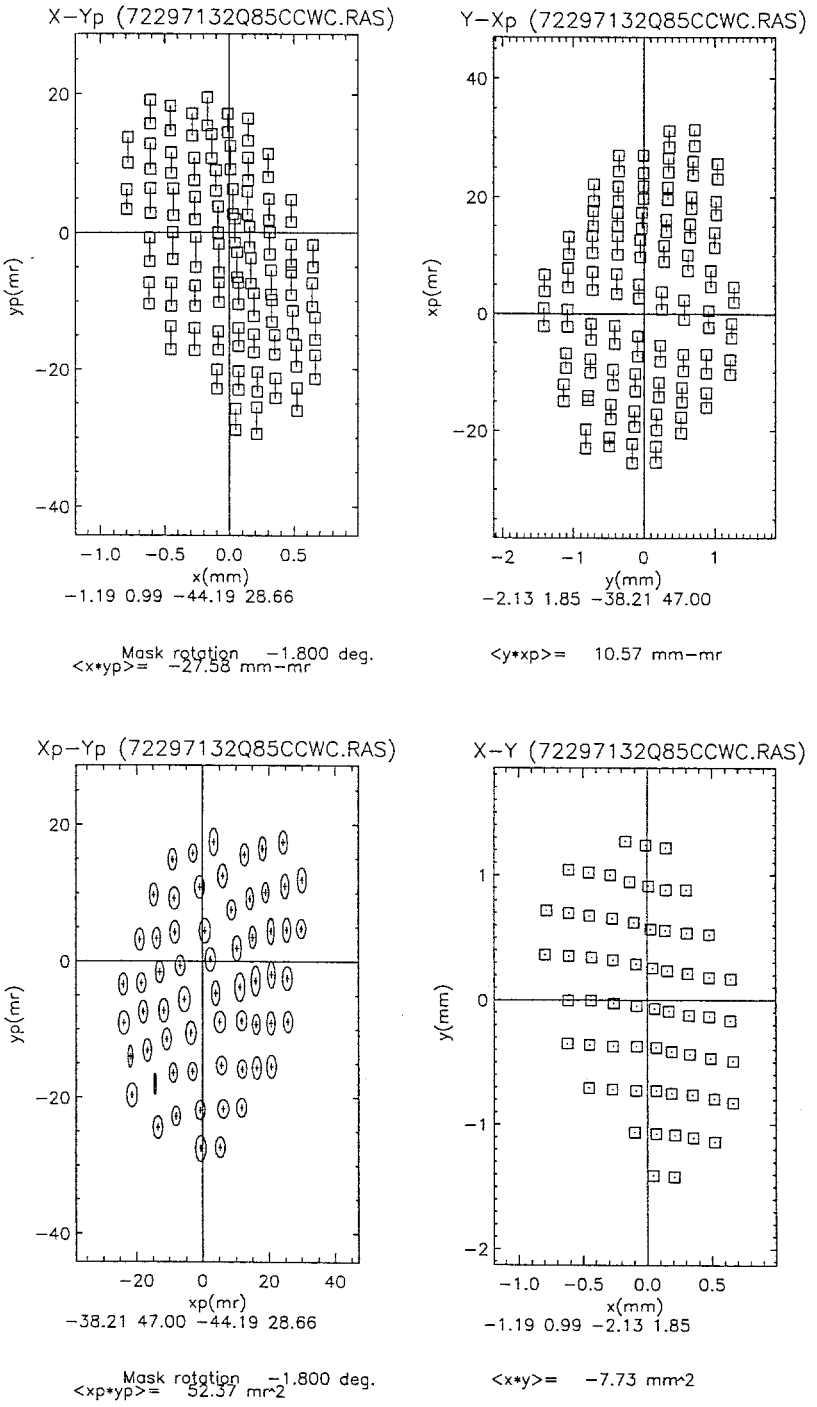


Figure 7.14: Cross moment plots with quadrupole 8 rotated 5° counter-clockwise.

Figure 7.10(c) shows the image obtained when quadrupole 8 is returned to alignment and a 5° clockwise rotation is placed on quadrupole 13. Similar to the rotation of quadrupole 8, the beam's x and y emittances increase along with the cross moments because of the increased coupling. Figures 7.15, 7.4, and 7.16 show the emittance ellipse and cross moment data plots.

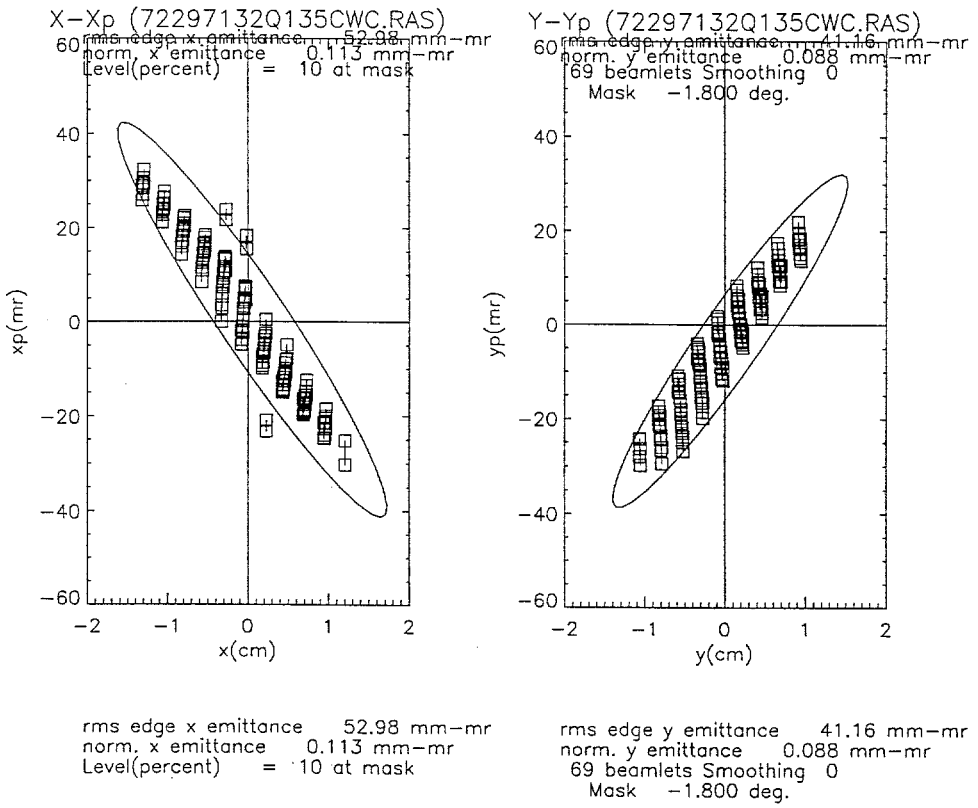


Figure 7.15: Phase space emittance plots with quadrupole 13 rotated 5° clockwise.

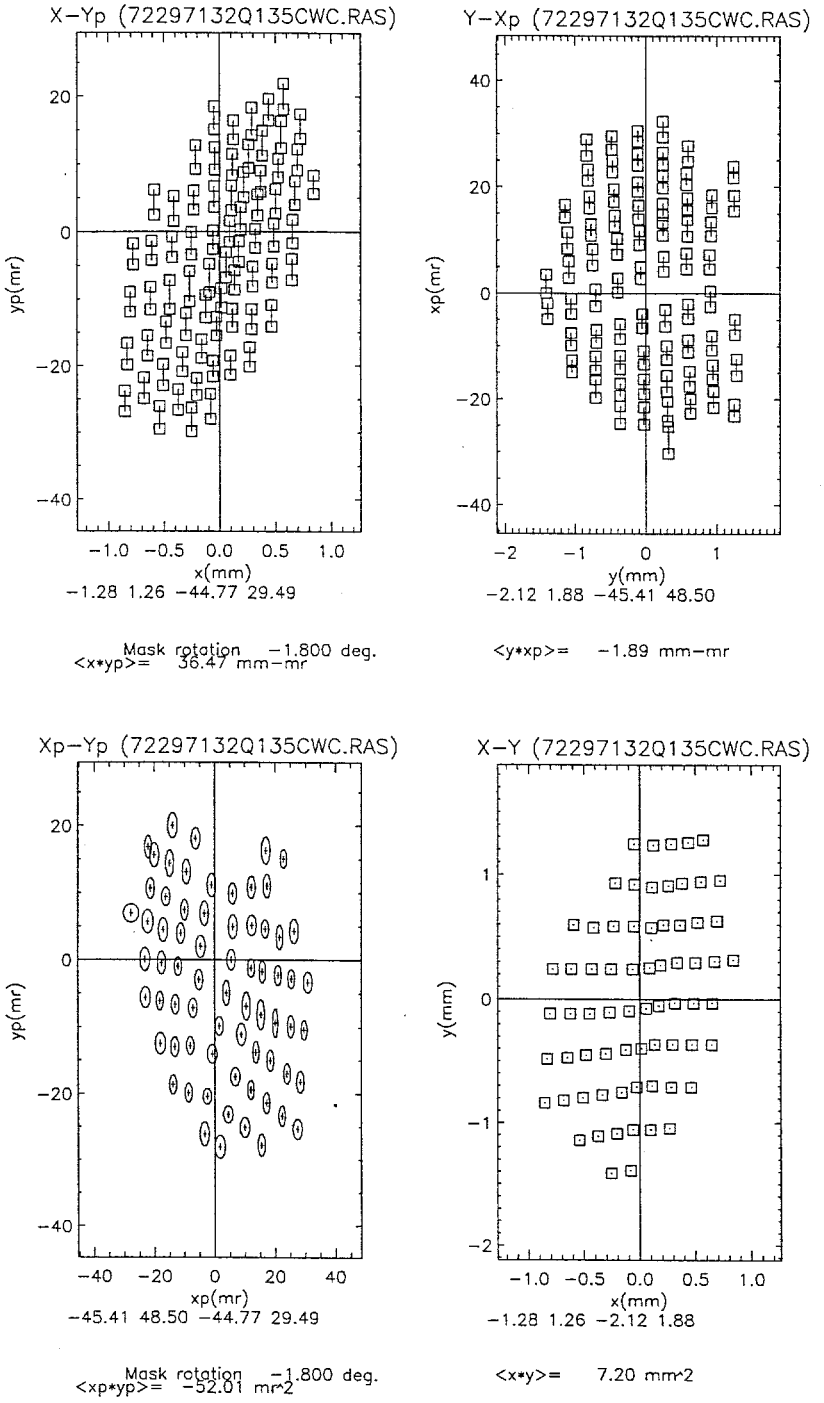


Figure 7.16: Cross moment plot with quadrupole 13 rotated 5° clockwise.

Figure 7.10(d) shows the results when quadrupole 8 and 13 are counter-rotated by 5° approximating the rudimantary beam correction for a known quadrupole rotation error. In this case the error in quadrupole 8 being corrected by quadrupole 13. Again it is an approximation to the special case simulation results shown in Figure 6.4 due to the physical limitations previously discussed. The image itself is upright and the rows and columns are parallel to the x and y axes. As all the other images indicated the 1.7° rotation of the pepperpot, this parallelism indicates that the "correction" is less than optimal as expected. The calculated moments and emittances do however substantially show values much improved over those with only one quadrupole rotated. Figures 7.17, 7.4, and 7.18 show the emittance ellipse and cross moment data plots.

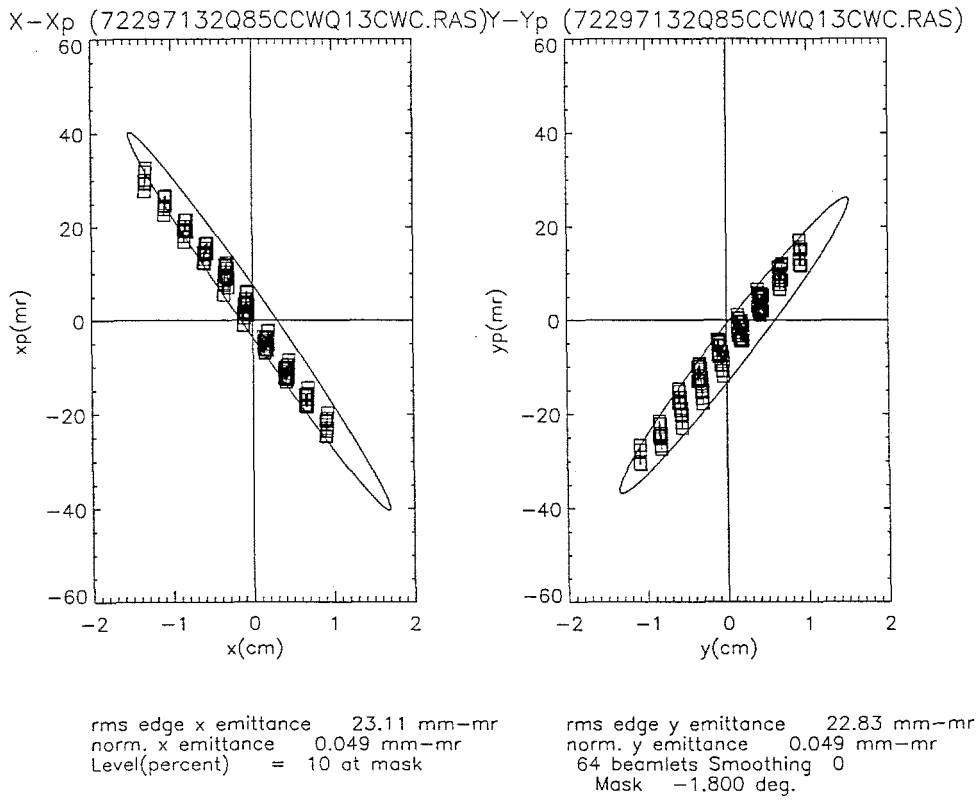


Figure 7.17: Phase space emittance plots with quadrupoles 8 and 13 counter-rotated by 5°.

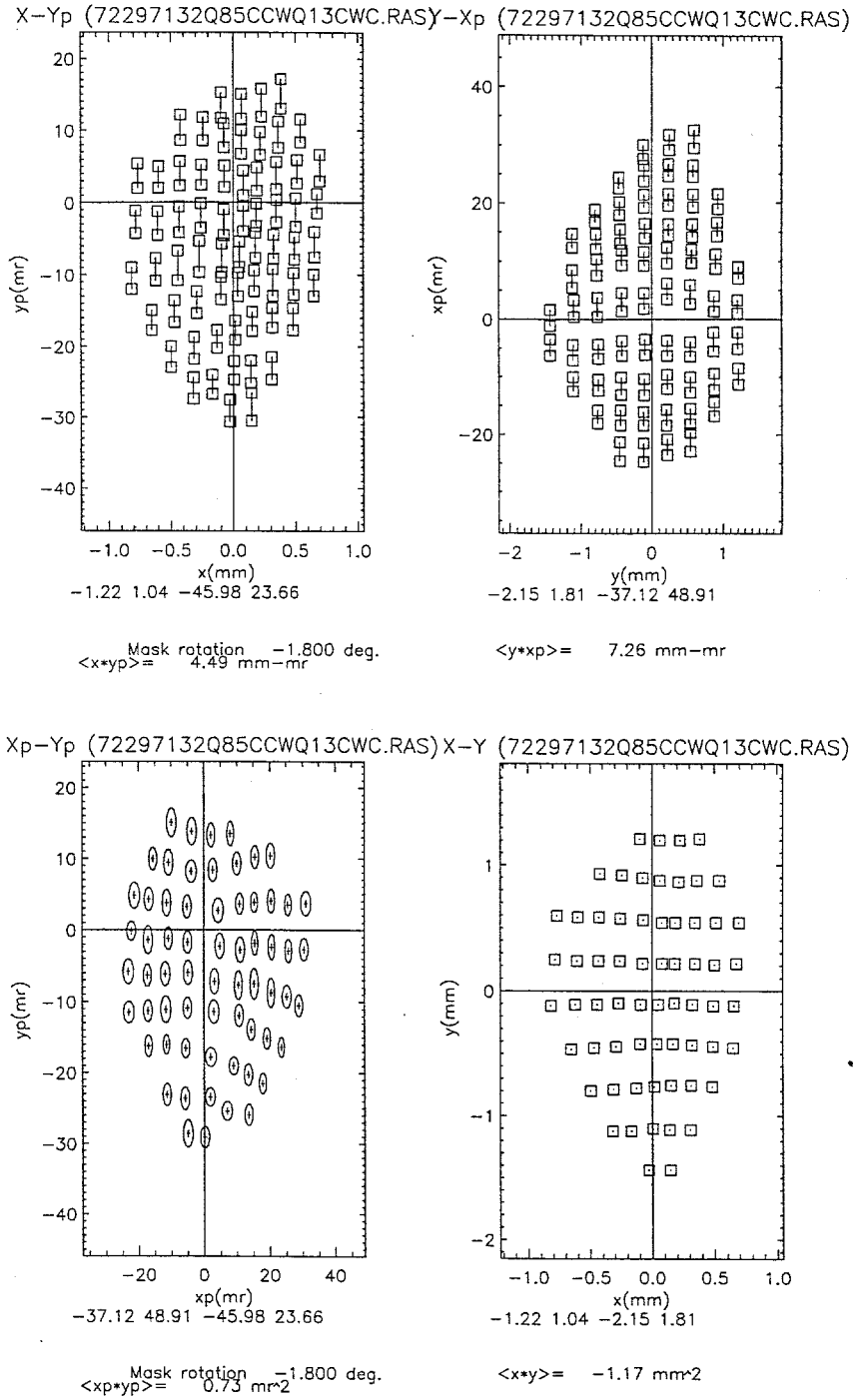


Figure 7.18: Cross moment plot with quadrupoles 8 and 13 counter-rotated by 5°.

Table 7.3: Summary of beam parameters with known quadrupole rotation errors. **a.** Quadrupole 8 rotated 5° counter-clockwise only **b.** Quadrupole 13 rotated 5° clockwise only **c.** Rudimentary beam correction applied by counter-rotation of quadrupoles 8 and 13 **d.** Parameters from a beam with no rotation errors.

Beam parameter	Q8 CCW	Q13 CW	"Corrected"	No Rotation
$<x>$ (mm)	0.96	-0.52	0.084	0.67
$<x'>$ (mr)	-0.5	0.6	0	1.1
$<y>$ (mm)	0.40	0.63	0.83	0.39
$<y'>$ (mr)	-4.3	-3.3	-5.2	-3.3
Δx^2 (mm^2)	32.3	35.1	32.9	34.6
$\Delta xx'$ (mm-mr)	-80.7	-83.4	-81.2	-87.3
$\Delta x'^2$ (mr^2)	208.2	218.3	204.1	223.8
Δy^2 (mm^2)	26.0	26.8	25.9	27.4
$\Delta yy'$ (mm-mr)	59.3	61.5	55.8	60.0
$\Delta y'^2$ (mr^2)	142.0	157.0	125.0	135.3
Δxy (mm^2)	-7.7	7.2	-1.2	0.7
$\Delta x'y$ (mm-mr)	10.6	-1.9	7.3	-3.8
$\Delta xy'$ (mm-mr)	-27.6	36.5	4.5	0.8
$\Delta x'y'$ (mr^2)	52.4	-52.0	0.7	-3.9
ϵ_x (rms) (mm-mr)	29.2	53.0	23.1	22.7
ϵ_y (rms) (mm-mr)	26.6	41.2	22.8	20.1
ϵ_{nx} (mm-mr)	0.062	0.11	0.049	0.048
ϵ_{ny} (mm-mr)	0.057	0.09	0.049	0.043
x envelope (mm)	11.3	11.8	11.4	11.7
x' envelope (mr)	-28.4	-28.1	-28.2	-29.6
y envelope (mm)	10.2	10.3	10.1	10.4
y' envelope (mr)	23.3	23.7	21.9	22.9
angular momentum (mm-mr)	-38.2	38.4	-2.8	4.6

7.5 Beam Transverse Intensity Distribution Measurement

An examination of the intensity distribution in the x-y plane is also of great interest in determining the performance of the ion source. The calculations of the emittance yields a value for an equivalent K-V beam (uniform distribution in phase space) but the actual distribution may be quite different. A surface plot of the beamlet total intensity vs. x-y position with interpolation between spots is shown in Figure 7.19. As one can see the Small Recirculator beam at the end of 45° was quite different from a uniform distribution. A surface plot of this kind is very useful as the difference in intensity level is less evident looking at the image or a 1-D cut along a line of beamlet spots shown in Figure 7.21. The resulting conclusion that the intensity distribution is far from uniform across the beam was not evident (nor easily observable without a lot of effort) from previous slit scanner measurements. Using the slit scanner data the degradation of the ion source had not been detected, although a re-examination of the data shows correlation between it and the GBI intensity data. A scan of the source near its output aperature using a pair of 1 mm cross slits and a Faraday cup was performed on August 2, 1997. This scan shown in Figure 7.20 also showed a large variation in source intensity prior to beam transport through the electric and magnetic quadrupole lattice. Subsequent physical analysis of the ion source at Lawrence Berkeley Laboratory showed that the source was largely depleted of potassium.[54] The attributes of the GBI that allow it to quickly measure the 2-D intensity profile of the beam make it most useful in gaging source performance degradation once calibrated against a known good source and any anomalies in beam transport are eliminated in contributing to a changed intensity profile.

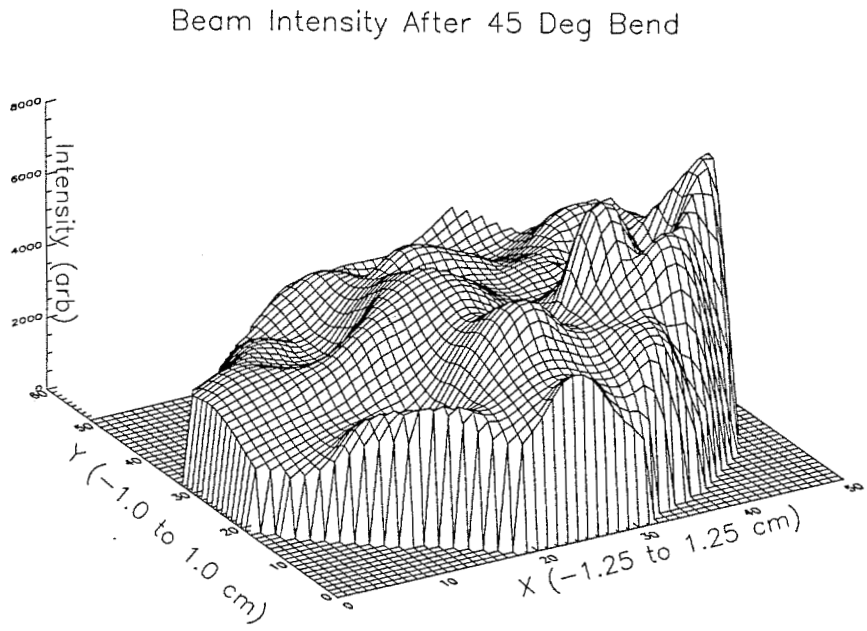


Figure 7.19: Small Recirculator beamlet intensity surface plot after the 45° bend showing the uneven emission of the potassium ion source.

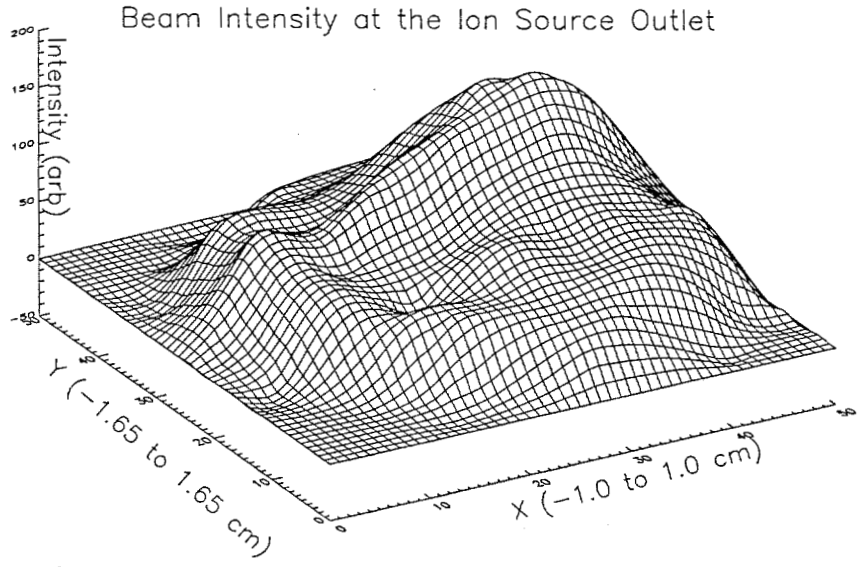


Figure 7.20: Small Recirculator ion source scan at the source indicating non-uniformity of emission.

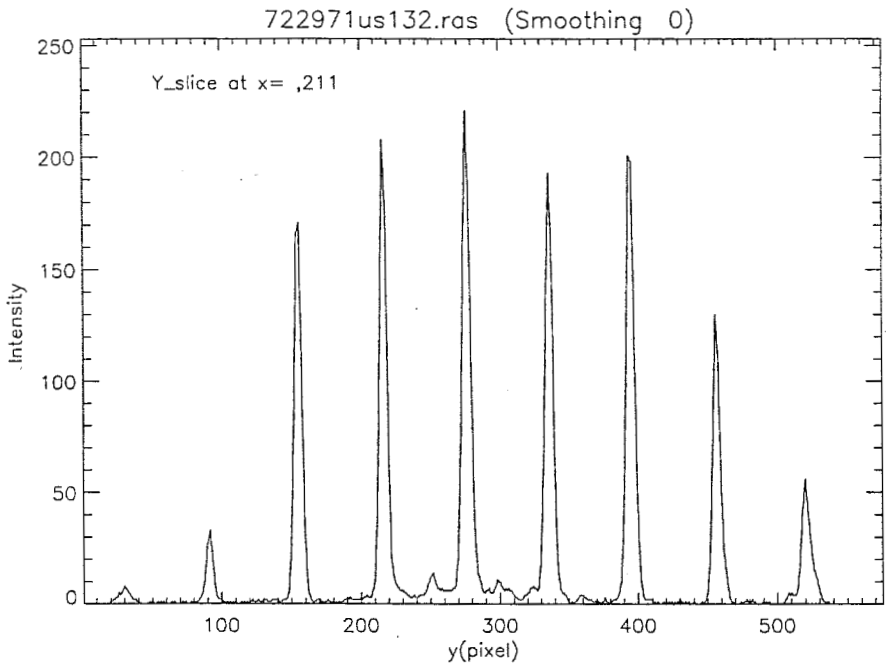


Figure 7.21: GBI image pixel row average plot for a column of beamlets.

Chapter 8

Summary and Conclusions

The principle goal of this dissertation was to design, build, and test a new diagnostic instrument capable of rapidly, and durably measuring the four-dimensional transverse phase space characteristics of an intense space- charge-dominated heavy ion beam. The Gated Beam Imager was developed and uses the pepperpot method of analysis to statistically extract the beam ion's positions and corresponding transverse velocities. The GBI provides durable, and repeatable measurements by employing a microchannel plate as the ion beamlet detector. The GBI was successfully deployed on LLNL's Small Recirculator experiment and measured the characteristics of a 2 mA potassium ion beam. In the construction and testing of this instrument, a few of the more challenging issues included the selection of a material or device to detect the beamlets reliably with no loss of sensitivity with high exposure to the ion beam, optimizing the design of the instrument and the camera optical system, and developing software analysis tools to analyze the images and provide a convenient user interface.

A secondary goal of this work was to show the utility of the additional information available from the GBI, but not from existing diagnostic instruments. This was done by developing a conceptual design for a beam correction system addressing how to correct for rotational misalignments of quadrupoles in an accelerator's alternating gradient focusing system. A demonstration of a computational method, using the information available from a GBI, for determining the settings of such a correction system was performed. A practical use for this correction system would be to increase

the focusing quadrupole alignment tolerances during the construction of a driver-scale accelerator which would presumably lead to a decrease in construction and operating costs.

The predominant phase space diagnostic in use on heavy ion beams is the mechanical slit scanner. This diagnostic uses two parallel, narrow slits moved through the beam on successive beam shots to measure the first and second transverse moments of the beam distribution in the direction perpendicular to the slits. Averaging beam intensity information occurs in the direction parallel to the slits resulting in the measurement containing a contribuiton from any coupled motion between the orthogonal transverse directions (x and y). The time to scan the beam is about 15 minutes and requires about 400 beam shots. To obtain the moments in the orthogonal direction the slit scanners are rotated 90° or a second set of scanners is installed at the same position. With the GBI, one is able to obtain in a single beam shot, the first and second moments of the beam in both transverse directions and the four cross-moments between the x and y directions which are indicative of coupling between motion in the x and y directions. A cause of this coupling in an alternate gradient focusing lattice are quadrupole rotational misalignments. The transverse cross-moments cannot be measured with the existing slit scanners due to the averaging of beam information in the direction parallel to the slits.The moments are intensity-weighted as the GBI preserves the intensity characteristics of each beamlet as it is transformed to an optical signal and digitized by the CCD camera. A time history of the beam is obtainable by gating the GBI detector on for a short time and moving the gate time through the beam on successive shots. Thus, a transverse phase space time-dependent map of the beam is obtained which can be compared with detailed numerical simulations or used as input for a beam correction system.

The unique design of the GBI includes the use of a microchannel plate to detect the ion beamlets produced by the pepperpot mask, and convert this signal to an visible optical ouput that preserves the intensity distribution of the beamlets. The spatial orientation of the beamlets is also preserved allowing the hole in the mask, from which the beamlet originated, to be identified for moment calculations. In the

work to develop the GBI, fast plastic scintillator material was first investigated as a possible beamlet detector. Plastic scintillator was investigated based on its rapid scintillation decay constant which would allow obtaining narrow time slices from a beam by gating the observing camera rapidly "on" and "off". This would allow the construction of an inexpensive detector and take advantage of existing camera capabilities. During tests, plastic scintillator proved to be rapidly de-sensitized under repeated exposure to the ion beam. Additionally, its initial light output was too low in response to the ions to obtain accurate images of the beamlet spots. Presumably, the short range of the heavy ions precluded sufficient interaction with the scintillator to produce adequate visible output.

The microchannel plate detector eventually incorporated into the GBI has several features that allow it to produce accurate, and repeatable detection of the beamlets and produce an optical signal strong enough to be useful for transverse phase space moment analysis. To avoid the damaging effects of the heavy ions, the microchannel plate selected has a thin stainless steel coating on the front face of the microchannels. The primary purpose of this coating is to provide an anode to allow placing a voltage across the microchannels to produce amplification of an incoming signal. The anode is thick enough to stop heavy ions in the energy range of the Small Recirculator. The secondary electrons produced during ion stopping enter the microchannel plate and are amplified and detected with a phosphor screen at the output of the microchannel plate. This "shielding" of the microchannel plate from the heavy ions side steps any damage issues. Additionally the gain of the microchannel plate is linear allowing optical signal amplification while maintaining the intensity characteristics of the beamlets. The spatial resolution of the position and size of the beamlets is not significantly affected by the "discretizing" of the beamlets into channels of finite size and spacing as the resolution of CCD camera systems is less than the microchannel spacing. Obtaining narrow time slices of the beam proved more difficult with a microchannel plate detector as it was necessary to gate the bias voltage to the detector rather than use the built-in gate on a CCD camera. This was a consequence of commercially-available microchannel plates using phosphor screens which produce

an output with a long response time. Confirmed during testing, the microchannel plate detector produced accurate, repeatable, and flexible images well suited to data analysis.

GBI measurements on the Small Recirculator beam showed good agreement with mechanical slit scanner emittance data obtained in the y-direction. Similar magnitude error bars as those calculated for the GBI are applicable to the slit scanner data. Comparison of the GBI moment data with the output of the WARP code, a sophisticated particle-in-cell numerical beam simulation, shows that the assumed initial velocity spread of the ions in the axial direction should be adjusted. WARP calculation of the beam emittance at the position of the GBI shows much more growth in emittance in the x (in-plane with the bend) direction than measured. The simulation results are much greater than the data and well outside experimental error bars. It is known that the emittance growth is very sensitive to the assumed initial width of the semi-gaussian spread in the ion axial velocity. Beam moments from the GBI data, analyzed to yield the beam envelope and envelope convergence angle, agree well with the WARP calculations. These envelope calculations are insensitive to the assumed ion axial velocity spread and lead to the conclusion that the code axial velocity spread assumptions should be re-examined. These results are a measure of the validity of the Small Recirculator experimental program of which a main goal is to benchmark sophisticated codes such as WARP. This requires that those quantities not obtainable through first principles or direct measurement be empirically determined.

The usefulness of a 4-D phase space diagnostic such as the GBI is demonstrated in the conceptual beam correction system addressing quadrupole rotation errors proposed in this work. The ability to measure the $x - y$ cross-moments makes such a system possible. Extension of Barnard's work that found constants of the motion in the presence of rotation errors was necessary to demonstrate efficient design of the correction system. That such a system will decrease the construction and operating costs of a driver-scale accelerator is realizable in the ability to increase alignment tolerances of focusing components and to be able to withstand any component aging effects resulting in rotational misalignments. The "model-free" attributes of the Si-

multaneous Perturbation Stochastic Approximation (SPSA) algorithm for optimizing correction system parameters were successful due the nonlinear dependence of the beam moments on the optimized parameters. SPSA requires only two calculations of the parameter gradients regardless of the dimensionality of the system. This was very efficient, although it was noted that the difficulty in tuning the algorithm to converge quickly increased with the increasing dimensionality of the problem.

The GBI combined with user-friendly analysis software has been shown to rapidly, and efficiently provide accurate, and repeatable characterization of heavy ion beams. It provides additional information beyond that of other diagnostics. This information is useful in diagnosing the cause of beam emittance growth and allows the concept of "correcting" a beam to be further studied. The GBI developed for this work is a prototype and showed that reliable detection of an intense beam heavy ions could be accomplished in a manner allowing characterization of the beam's 4-D transverse phase space. The further development of the basic components of this instrument will continue allowing it to be a more fexible, and capable instrument in meeting the need to supply data on the operation of heavy ion accelerators.

Appendix A

GBI Design Optimizing Spreadsheet

The following is a printout of the spreadsheet incorporating the design equations of the GBI. It allows the user to vary the beam current, average radius, energy, length, emittance, ion mass, pepperpot mask thickness, hole diameter and spacing, drift distance, and the gate interval. The effect on beamlet spread due to emittance and space charge as well as beamlet spot spacing at the detector is calculated and checked against a minimum criteria. Additionally, estimates of the signal from the detector is estimated.

Gated Beam Imager Design Calculations		
Input Data		
Beam Current (mA)		2
Beam Radius (cm)		0.7
Beam Energy (keV)		80
Beam Length (usec)		4
Ion Mass (amu)		39.1
Beam Emittance (mm-mr)		20
Hole Plate Thickness (μ)		457.2
Hole Diameter (μ)		100
Hole Spacing (mm)		2.5
Drift Distance (cm)		15
Gate Time (nsec)		100
Constants		
electron charge (C)		1.60E-19
a.m.u. (kg)		1.66E-27
pi		3.14159
permitivity (F/m)		8.85E-12
Conversions		
μ/mil	25.4	
Å/μ	1.00E+04	
mm/mil	2.54E-02	
statcoul/coul	3.00E+09	

Calculated Quantities				
Main Beam				
# of ions in Beam pulse			5.00E+10	
Energy per Beam pulse (J)			6.40E-04	
current density (A/sq. m)			1.30E+01	
ion velocity (m/s)			6.28E+05	
Emittance Divergence angle (mradians)			2.86	
Aperatured Beam				
# ions thru an aperature (ungated)			2.55E+06	
Beamlet crossection (sq. μ)			7.85E+03	
Energy per beamlet (J)			3.27E-08	
# ions thru an aperature (gated)			6.38E+04	
Beamlet crossection (sq. μ)			7.85E+03	
Energy per beamlet (J)			8.16E-10	
Maximum acceptance angle through a hole (°)			12.34	*OK*
Beamlet spread due to space charge (%)			8.22	*OK*
Beamlet spread due to space charge (μ)			8.22	
Beamlet spread due to emittance growth (%)			428.57	*OK*
Beamlet spread due to emittance growth (μ)			428.57	
Beamlet Diameter at MCP (μ)			536.79	
Distance between beamlet images at MCP (mm)			1.53	*OK*

Photon Production/Detection Calculation						
MCP Detector						
# ions thru an aperature (gated)			6.38E+04			
# of secondary electrons per ion			3	(1-5)		
MCP detector gain			10.00			
# of photons produced per electron at phosphor			5	(1-5)		
# photons produced by phosphor			9.57E+06			
1/R^2 Spreading Loss						
distance from phosphor to camera lens (cm)			15.00			
diameter of camera lens (cm)			2.54			
solid angle subtended by camera lens (steradians)			0.53			
Image Intensifier						
# photons incident on image itensifier			4.04E+05			
image intensifier gain			1.00E+03			
# photoelectrons out of the image intensifier			4.04E+08			
# of photons produced per electron at phosphor			5	(1-5)		
# photons produced by phosphor			2.02E+09			
Image Inten. Resolution (μ)			33			
Image Inten. pore size (sq. μ)			855.30			
Camera CCD						
Quantum Efficiency at 420nm			0.14			
camera resolution (μ)			22.5			
Pixel area (sq. μ)			397.61			
# pixels per beamlet image			569.17			
photoelectrons per pixel			4.96E+05			

Assumptions				
1. Each ion produces between 1-10 secondary electrons at the MCP detector front electrode				
2. The gain of the MCP detector is kept very low (10-100X)				
3. Each electron incident on the detector phosphor produces 1-10 photons				
4. Assume 1:1 imaging by camera (i.e. size of beamlet image on phosphor is the same as on the image intensifier				
5. Gain of the image intensifier is ~ 1000X				
6. Assume due to the fiber-optic coupling between the image intensifier and the CCD that 1/2 of the photons produced in the phosphor arrive at the CCD				
7. Criteria for maximum acceptance angle based on emittance of 50 mm-mr and max beam rotion angle of 1.72°				
8. Criteria for beamlet spreading: <10% from space charge & >25% from emittance				
9. Minimum beamlet separation is 0.5 mm				
Camera CCD Information				
Number of pixels X		384		
Numberof pixels Y		576		
Total pixels		221184		
X length (mm)		8.6		
Y length (mm)		12.9		
μ/pixel X		22.4		
μ/pixel Y		22.4		

Appendix B

Derivation of Beam Moment Evolution Equations

This appendix follows the work of Barnard in deriving the moment evolution equations of Chapter 6 [footnote]. Barnard assumed that the force on an ion comes from two sources only, the external focusing force from a purely quadrupolar field, and an internal force from the space charge of the beam. The calculation assumes the space charge is evenly distributed in a uniform density ellipse but the semi-axes and rotation angle of the ellipse is allowed to evolve as a function of the axial coordinate, z. Quadrupoles can be rotated by an angle, θ from the x-axis, and the beam can be rotated by an angle, α from the x-axis as seen in Figure reffig:coordsyst

The relation between the coordinate in the quadrupole frame (subscript 0) and the lab frame (no subscript) are given by:

$$x = x_0 \cos\theta - y_0 \sin\theta \qquad y = y_0 \cos\theta + x_0 \sin\theta \tag{B.1}$$

and similarly for the beam frame:

$$x - <x> = x_b \cos\alpha - y_b \sin\alpha \qquad y - <y> = y_b \cos\alpha + x_b \sin\alpha \tag{B.2}$$

The "$< >$" indicates a statistical average over the beam ion distribution

For a non-relativistic beam, with velocity ratio β, moving along the z-axis the paraxial

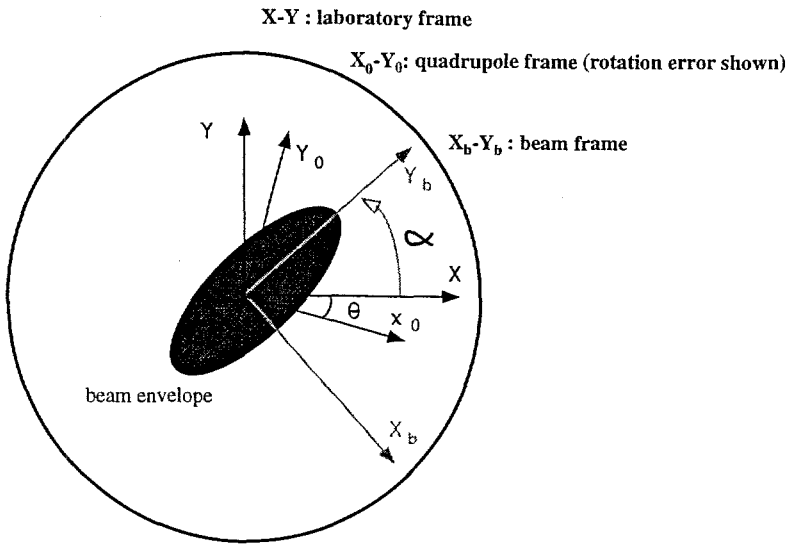


Figure B.1: Beam and laboratory coordinate system

equations of motion can then be written as:

$$x'' = \frac{d^2x}{dz^2} = K_{qxx}x + K_{qxy}y + K_{sxx}(x- <x>) + K_{sxy}(y- <y>) \tag{B.3}$$

$$y'' = \frac{d^2y}{dz^2} = \underbrace{K_{qyy}y + K_{qyx}x}_{externalforces} + \underbrace{K_{syy}(y- <y>) + K_{syx}(x- <x>)}_{internalforces} \tag{B.4}$$

$K_{qxx} \equiv K_{qx0}\cos^2\theta + K_{qy0}\sin^2\theta = K_{qx0}\cos 2\theta$

$K_{qyy} \equiv K_{qy0}\cos^2\theta + K_{qx0}\sin^2\theta = -K_{qx0}\cos 2\theta$

$K_{qxy} \equiv K_{qyx} = (K_{qx0} - K_{qy0})(\sin\theta\cos\theta) = K_{qx0}\sin 2\theta$

$K_{qyy} \equiv K_{qyo}\cos^2\theta + K_{qx0}\sin^2\theta = -K_{qx0}\cos 2\theta$

$K_{sxx} = K_{sxb}\cos 2\alpha + K_{syb}\sin 2\alpha \qquad K_{syy} = K_{syb}\cos 2\alpha + K_{sxb}\sin 2\alpha$

$K_{sxy} = K_{syx} = (K_{sxb} - K_{syb})\sin\alpha\cos\alpha$

$K_{qx0} = -K_{qy0} = \frac{B'}{B\rho}$

For instance, K_{qxx} is the proportional constant for the quadrupolar force on an ion in the x-direction caused by a dispacement in the x-direction from the center of the magnetic quadrupole which corresponds with the center of the beamline. K_{sxy} is the proportional constant for the space-charge force on an ion in the x-direction caused by a dispacement in the y-direction from the centroid of the beam. In the rotated beam frame:

$$K_{sxb} = \frac{\kappa}{2(\Delta x_b^2 + (\Delta x_b^2 \Delta y_b^2)^{1/2})} \qquad K_{syb} = \frac{\kappa}{2(\Delta y_b^2 + (\Delta x_b^2 \Delta y_b^2)^{1/2})}$$

and Δx_b and Δy_b are the moments in the rotated beam frame:

$$\Delta x_b^2 = \Delta x^2 \cos 2\alpha + \Delta y^2 \sin 2\alpha + 2\Delta xy \cos\alpha \sin\alpha \tag{B.5}$$

$$\Delta y_b^2 = \Delta y^2 \cos 2\alpha + \Delta x^2 \sin 2\alpha - 2\Delta xy \cos\alpha \sin\alpha \tag{B.6}$$

Here κ is the beam perveance and is a measure of the space charge forces relative to inertial forces.

$$\kappa = \frac{qeI_0}{2\pi\epsilon_0 m_0 (\beta\gamma c)^3} \tag{B.7}$$

where q is the charge state of beam ions, e is the proton charge, m_0 is the beam ion mass in atomic mass units, γ is the Lorentz factor, βc is the ion velocity, I_0 is the beam current, and ϵ_0 is he permittivity of free space. The symbol, Δ is shorthand for the second moment of the distribution with the contribution from non-zero first moments removed, (i.e., $\Delta ab \equiv <ab> - <a><b>$).

The beam rotation angle may be expressed in terms of the second order moments by

$$\tan 2\alpha = \frac{2\Delta xy}{\Delta x^2 - \Delta y^2} \tag{B.8}$$

Let the beam distribution function f,be the number of particles dN per unit transverse phase space volume:

$$f(x, x', y, y') = \frac{dN}{dx dx' dy dy'} \tag{B.9}$$

Since the beam can be described as a collision-less non-neutral plasma the evolution

of f can be described by the Vlasov/Collisionless Boltzman Equation

$$\frac{\partial f}{\partial z} + x'\frac{\partial f}{\partial x} + x''\frac{\partial f}{\partial x'} + y'\frac{\partial f}{\partial y} + y''\frac{\partial f}{\partial y'} + (\frac{\partial x''}{\partial x'} + \frac{\partial y''}{\partial y'})f = 0 \tag{B.10}$$

where x'' and y'' are determined by the equations of motion. The average of a variable over a continuous distribution is given by:

$$<\phi>(z) \equiv \frac{1}{N}\iiiint \phi f(x, x', y, y')dxdx'dydy' \tag{B.11}$$

substituting the moments into the above equation and integrating by parts yields the moment evolution equations:

$$\frac{d\Delta x^2}{dz} = 2\Delta xx'$$

$$\frac{d\Delta xx'}{dz} = \Delta x'^2 + K_{xx}\Delta x^2 + K_{xy}\Delta xy - (dln\beta_z/dz)\Delta xx'$$

$$\frac{d\Delta x'^2}{dz} = 2K_{xx}\Delta xx' + 2K_{xy}\Delta x'y - 2(dln\beta_z/dz)\Delta x'^2$$

$$\frac{d\Delta y^2}{dz} = 2\Delta yy'$$

$$\frac{d\Delta yy'}{dz} = \Delta y'^2 + K_{yy}\Delta y^2 + K_{yx}\Delta xy - (dln\beta_z/dz)\Delta yy'$$

$$\frac{d\Delta y'^2}{dz} = 2K_{yy}\Delta yy' + 2K_{xy}\Delta xy' - 2(dln\beta_z/dz)\Delta y'^2$$

$$\frac{d\Delta xy}{dz} = \Delta x'y + \Delta xy'$$

$$\frac{d\Delta x'y}{dz} = \Delta x'y' + K_{xx}\Delta xy + K_{yx}\Delta y^2 - (dln\beta_z/dz)\Delta x'y$$

$$\frac{d\Delta xy'}{dz} = \Delta x'y' + K_{yy}\Delta xy + K_{xy}\Delta x^2 - (dln\beta_z/dz)\Delta xy'$$

$$\frac{d\Delta x'y'}{dz} = K_{xx}\Delta xy' + K_{xy}\Delta yy' + K_{yy}\Delta x'y + K_{yx}\Delta xx' - 2(dln\beta_z/dz)\Delta x'y'$$

where

$$K_{xx} \equiv K_{qxx} + K_{sxx}$$

$$K_{yy} \equiv K_{qyy} + K_{syy}$$

$$K_{xy} \equiv K_{qxy} + K_{sxy} = K_{yx}$$

and similarly the first moment evolution equations are given by

$$\frac{d<x>}{dz} =< x' > \qquad \frac{d<x'>}{dz} = K_{qxx} < x > + K_{qxy} < y > -(dln\beta_z/dz) < x' >$$

$$\frac{d<y>}{dz} =< y' > \qquad \frac{d<y'>}{dz} = K_{qyy} < y > + K_{qyx} < x > -(dln\beta_z/dz) < y' >$$

Appendix C

Explicit Solutions for Δx^2 and Δy^2

From an engineering and cost point of view it is desirable to reduce the number of independent adjustable parameters necessary to correct the beam for random quadrupole rotatational misalignments. Using Barnard's generalized emittances the matrix of ten moment evolution equations may be reduced to eight by explicitly solving the expressions for the generalized emittances for two of the moments in terms of the remaining eight moments . It was found helpful to use the capabiltiies of *Mathematica* to do this and it was found that solving for Δx^2 and Δy^2 yielded the "simplest" result. The *Mathematica* command for doing this was:

$$Solve[<< constraints.m >>, \{dx2, dy2\}] \tag{C.1}$$

where the file "constraints.m" contained the generalized emittance expressions and "dx2,dy2" are Δx^2,Δy^2 in a form readable by *Mathematica.* The explicit solutions are very long and as discussed in Chapter 6 they were able to be simplified when the numerical values for the beam moments corresponding to the Small Recirculator were used in the formulas. The terms that were dropped from the following explicit solutions to obtain the results of Equations 6.11 and 6.16 appear in a lighter shade. The numerical value of the terms that were dropped was about 20 orders of magnitude smaller than those terms kept. These solutions are then substituted anywhere Δx^2,Δy^2 appear in the ten moment equations and the eight independent moment evolution equations of the reduced matrix of Equation 6.6 are the result. All the

complications are contained in the numbered coefficients.

$$
\begin{aligned}
\Delta x^2 \;=\; & [4\,\beta^4\,\Delta x'y'^2\,\Delta xx'^2 - 8\,\beta^4\,\Delta x'y'^3\,\Delta xy + 8\,\beta^4\,\Delta x'y\,\Delta x'y'^2\,\Delta xy' + \\
& 8\,\beta^4\,\Delta x'y'\,(\Delta x'^2)\,\Delta xx'\,\Delta xy' - 4\,\beta^4\,(\Delta x'^2)^2\,\Delta xy'^2 - \\
& 8\,\beta^4\,(\Delta x'^2)\,\Delta xx'^2\,(\Delta y'^2) + 8\,\beta^4\,\Delta x'y'\,(\Delta x'^2)\,\Delta xy\,(\Delta y'^2) - \\
& 8\,\beta^4\,\Delta x'y\,(\Delta x'^2)\,\Delta xy'\,(\Delta y'^2) + 4\,\beta^4\,\Delta x'y^2\,(\Delta y'^2)^2 - \\
& 8\,\beta^4\,\Delta x'y\,\Delta x'y'\,(\Delta y'^2)\,\Delta yy' + 4\,\beta^4\,\Delta x'y'^2\,\Delta yy'^2 + \\
& 0.5\,\beta^2\,\Delta x'y'^2\,\epsilon_g{}^2 - 0.5\,\beta^2\,(\Delta x'^2)\,(\Delta y'^2)\,\epsilon_g{}^2 - \\
& 0.015625\,[65536\,\beta^8\,\Delta x'y'^4\,\Delta xx'^4 - 262144\,\beta^8\,\Delta x'y'^5\,\Delta xx'^2\,\Delta xy + \\
& 262144\,\beta^8\,\Delta x'y'^6\,\Delta xy^2 + \\
& 262144\,\beta^8\,\Delta x'y\,\Delta x'y'^4\,\Delta xx'^2\,\Delta xy' - \\
& 262144\,\beta^8\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xx'^3\,\Delta xy' - \\
& 524288\,\beta^8\,\Delta x'y\,\Delta x'y'^5\,\Delta xy\,\Delta xy' + 524288\,\beta^8\,\Delta x'y'^4\,(\Delta x'^2)\,\Delta xx'\,\Delta xy\,\Delta xy' + \\
& 262144\,\beta^8\,\Delta x'y^2\,\Delta x'y'^4\,\Delta xy'^2 - 524288\,\beta^8\,\Delta x'y\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xx'\,\Delta xy'^2 + \\
& 393216\,\beta^8\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xx'^2\,\Delta xy'^2 - \\
& 262144\,\beta^8\,\Delta x'y'^3\,(\Delta x'^2)^2\,\Delta xy\,\Delta xy'^2 + \\
& 262144\,\beta^8\,\Delta x'y\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xy'^3 - \\
& 262144\,\beta^8\,\Delta x'y'\,(\Delta x'^2)^3\,\Delta xx'\,\Delta xy'^3 + 65536\,\beta^8\,(\Delta x'^2)^4\,\Delta xy'^4 + \\
& 262144\,\beta^8\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xx'^2\,\Delta xy\,(\Delta y'^2) - \\
& 786432\,\beta^8\,\Delta x'y'^4\,(\Delta x'^2)\,\Delta xy^2\,(\Delta y'^2) - \\
& 262144\,\beta^8\,\Delta x'y\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xx'^2\,\Delta xy'\,(\Delta y'^2) + \\
& 1.57286\,10^6\,\beta^8\,\Delta x'y\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xy\,\Delta xy'\,(\Delta y'^2) - \\
& 524288\,\beta^8\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy\,\Delta xy'\,(\Delta y'^2) - \\
& 786432\,\beta^8\,\Delta x'y^2\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xy'^2\,(\Delta y'^2) + \\
& 524288\,\beta^8\,\Delta x'y\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy'^2\,(\Delta y'^2) + \\
& 262144\,\beta^8\,\Delta x'y'\,(\Delta x'^2)^3\,\Delta xy\,\Delta xy'^2\,(\Delta y'^2) -
\end{aligned}
$$

$$262144\,\beta^8\,\Delta x'y\,(\Delta x'^2)^3\,\Delta xy'^3\,(\Delta y'^2)\,+$$
$$131072\,\beta^8\,\Delta x'y^2\,\Delta x'y'^2\,\Delta xx'^2\,(\Delta y'^2)^2\,-$$
$$262144\,\beta^8\,\Delta x'y^2\,\Delta x'y'^3\,\Delta xy\,(\Delta y'^2)^2\,-$$
$$524288\,\beta^8\,\Delta x'y\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xx'\,\Delta xy\,(\Delta y'^2)^2\,+$$
$$786432\,\beta^8\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xy^2\,(\Delta y'^2)^2\,+$$
$$262144\,\beta^8\,\Delta x'y^3\,\Delta x'y'^2\,\Delta xy'\,(\Delta y'^2)^2\,+$$
$$262144\,\beta^8\,\Delta x'y^2\,\Delta x'y'\,(\Delta x'^2)\,\Delta xx'\,\Delta xy'\,(\Delta y'^2)^2\,-$$
$$1.04858\,10^6\,\beta^8\,\Delta x'y\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xy\,\Delta xy'\,(\Delta y'^2)^2\,+$$
$$393216\,\beta^8\,\Delta x'y^2\,(\Delta x'^2)^2\,\Delta xy'^2\,(\Delta y'^2)^2\,-$$
$$262144\,\beta^8\,\Delta x'y^2\,(\Delta x'^2)\,\Delta xx'^2\,(\Delta y'^2)^3\,+$$
$$262144\,\beta^8\,\Delta x'y^2\,\Delta x'y'\,(\Delta x'^2)\,\Delta xy\,(\Delta y'^2)^3\,+$$
$$524288\,\beta^8\,\Delta x'y\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy\,(\Delta y'^2)^3\,-$$
$$262144\,\beta^8\,(\Delta x'^2)^3\,\Delta xy^2\,(\Delta y'^2)^3\,-\,262144\,\beta^8\,\Delta x'y^3\,(\Delta x'^2)\,\Delta xy'\,(\Delta y'^2)^3\,+$$
$$65536\,\beta^8\,\Delta x'y^4\,(\Delta y'^2)^4\,-\,262144\,\beta^8\,\Delta x'y\,\Delta x'y'^3\,\Delta xx'^2\,(\Delta y'^2)\,\Delta yy'\,+$$
$$524288\,\beta^8\,\Delta x'y\,\Delta x'y'^4\,\Delta xy\,(\Delta y'^2)\,\Delta yy'\,+$$
$$524288\,\beta^8\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xx'\,\Delta xy\,(\Delta y'^2)\,\Delta yy'\,-$$
$$524288\,\beta^8\,\Delta x'y^2\,\Delta x'y'^3\,\Delta xy'\,(\Delta y'^2)\,\Delta yy'\,-$$
$$524288\,\beta^8\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xy\,\Delta xy'\,(\Delta y'^2)\,\Delta yy'\,+$$
$$262144\,\beta^8\,\Delta x'y\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xy'^2\,(\Delta y'^2)\,\Delta yy'\,+$$
$$524288\,\beta^8\,\Delta x'y\,\Delta x'y'\,(\Delta x'^2)\,\Delta xx'^2\,(\Delta y'^2)^2\,\Delta yy'\,-$$
$$524288\,\beta^8\,\Delta x'y\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xy\,(\Delta y'^2)^2\,\Delta yy'\,-$$
$$524288\,\beta^8\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy\,(\Delta y'^2)^2\,\Delta yy'\,+$$
$$524288\,\beta^8\,\Delta x'y^2\,\Delta x'y'\,(\Delta x'^2)\,\Delta xy'\,(\Delta y'^2)^2\,\Delta yy'\,-$$
$$524288\,\beta^8\,\Delta x'y\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy'\,(\Delta y'^2)^2\,\Delta yy'\,+$$
$$524288\,\beta^8\,(\Delta x'^2)^3\,\Delta xy\,\Delta xy'\,(\Delta y'^2)^2\,\Delta yy'\,-$$
$$262144\,\beta^8\,\Delta x'y^3\,\Delta x'y'\,(\Delta y'^2)^3\,\Delta yy'\,+$$
$$131072\,\beta^8\,\Delta x'y'^4\,\Delta xx'^2\,\Delta yy'^2\,-\,262144\,\beta^8\,\Delta x'y'^5\,\Delta xy\,\Delta yy'^2\,+$$

$$262144\,\beta^8\,\Delta x'y\,\Delta x'y'^4\,\Delta xy'\,\Delta yy'^2 - 262144\,\beta^8\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xx'\,\Delta xy'\,\Delta yy'^2 +$$
$$131072\,\beta^8\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xy'^2\,\Delta yy'^2 -$$
$$262144\,\beta^8\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xx'^2\,(\Delta y'^2)\,\Delta yy'^2 +$$
$$262144\,\beta^8\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xy\,(\Delta y'^2)\,\Delta yy'^2 -$$
$$262144\,\beta^8\,\Delta x'y\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xy'\,(\Delta y'^2)\,\Delta yy'^2 +$$
$$524288\,\beta^8\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy'\,(\Delta y'^2)\,\Delta yy'^2 -$$
$$262144\,\beta^8\,(\Delta x'^2)^3\,\Delta xy'^2\,(\Delta y'^2)\,\Delta yy'^2 +$$
$$393216\,\beta^8\,\Delta x'y^2\,\Delta x'y'^2\,(\Delta y'^2)^2\,\Delta yy'^2 -$$
$$262144\,\beta^8\,\Delta x'y\,\Delta x'y'^3\,(\Delta y'^2)\,\Delta yy'^3 + 65536\,\beta^8\,\Delta x'y'^4\,\Delta yy'^4 +$$
$$16384\,\beta^6\,\Delta x'y'^4\,\Delta xx'^2\,\epsilon_g{}^2 - 32768\,\beta^6\,\Delta x'y'^5\,\Delta xy\,\epsilon_g{}^2 +$$
$$32768\,\beta^6\,\Delta x'y\,\Delta x'y'^4\,\Delta xy'\,\epsilon_g{}^2 -$$
$$32768\,\beta^6\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xx'\,\Delta xy'\,\epsilon_g{}^2 +$$
$$16384\,\beta^6\,\Delta x'y'^2\,(\Delta x'^2)^2\,\Delta xy'^2\,\epsilon_g{}^2 -$$
$$16384\,\beta^6\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xx'^2\,(\Delta y'^2)\,\epsilon_g{}^2 +$$
$$65536\,\beta^6\,\Delta x'y'^3\,(\Delta x'^2)\,\Delta xy\,(\Delta y'^2)\,\epsilon_g{}^2 -$$
$$65536\,\beta^6\,\Delta x'y\,\Delta x'y'^2\,(\Delta x'^2)\,\Delta xy'\,(\Delta y'^2)\,\epsilon_g{}^2 +$$
$$32768\,\beta^6\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xx'\,\Delta xy'\,(\Delta y'^2)\,\epsilon_g{}^2 -$$
$$16384\,\beta^6\,(\Delta x'^2)^3\,\Delta xy'^2\,(\Delta y'^2)\,\epsilon_g{}^2 +$$
$$16384\,\beta^6\,\Delta x'y^2\,\Delta x'y'^2\,(\Delta y'^2)^2\,\epsilon_g{}^2 -$$
$$32768\,\beta^6\,\Delta x'y'\,(\Delta x'^2)^2\,\Delta xy\,(\Delta y'^2)^2\,\epsilon_g{}^2 +$$
$$32768\,\beta^6\,\Delta x'y\,(\Delta x'^2)^2\,\Delta xy'\,(\Delta y'^2)^2\,\epsilon_g{}^2 -$$
$$16384\,\beta^6\,\Delta x'y^2\,(\Delta x'^2)\,(\Delta y'^2)^3\,\epsilon_g{}^2 -$$
$$32768\,\beta^6\,\Delta x'y\,\Delta x'y'^3\,(\Delta y'^2)\,\Delta yy'\,\epsilon_g{}^2 +$$
$$32768\,\beta^6\,\Delta x'y\,\Delta x'y'\,(\Delta x'^2)\,(\Delta y'^2)^2\,\Delta yy'\,\epsilon_g{}^2 +$$
$$16384\,\beta^6\,\Delta x'y'^4\,\Delta yy'^2\,\epsilon_g{}^2 - 16384\,\beta^6\,\Delta x'y'^2\,(\Delta x'^2)\,(\Delta y'^2)\,\Delta yy'^2\,\epsilon_g{}^2 +$$
$$1024\,\beta^4\,\Delta x'y'^4\,(\epsilon_g)^4 -$$
$$2048\,\beta^4\,\Delta x'y'^2\,(\Delta x'^2)\,(\Delta y'^2)\,(\epsilon_g{}^2)^2 + 1024\,\beta^4\,(\Delta x'^2)^2\,(\Delta y'^2)^2\,(\epsilon_g)^4 +$$

$$1024\,\beta^4\,\Delta x'y'^2\,(\Delta x'^2)\,(\Delta y'^2)\,(\epsilon_h)^4 - 1024\,\beta^4\,(\Delta x'^2)^2\,(\Delta y'^2)^2\,(\epsilon_h)^4]^{0.5}]/$$
$$[\beta^4\,\mathbf{\Delta x'^2}\,(8\,\mathbf{\Delta x'y'^2} - 8\,\mathbf{\Delta x'^2}\,\mathbf{\Delta y'^2})]$$

(C.3)

$$\Delta y^2 = \mathbf{0.5}\,[\beta^2\,[\mathbf{256}\,\beta^2\,\mathbf{\Delta x'y'^2}\,\mathbf{\Delta xx'^2} -$$
$$\mathbf{512}\,\beta^2\,\mathbf{\Delta x'y'^3}\,\mathbf{\Delta xy} +$$
$$\mathbf{512}\,\beta^2\,\mathbf{\Delta x'y}\,\mathbf{\Delta x'y'^2}\,\mathbf{\Delta xy'} - \mathbf{512}\,\beta^2\,\mathbf{\Delta x'y'}\,\mathbf{\Delta x'^2}\,\mathbf{\Delta xx'}\,\mathbf{\Delta xy'} +$$
$$\mathbf{256}\,\beta^2\,(\mathbf{\Delta x'^2})^2\,\mathbf{\Delta xy'^2} +$$
$$\mathbf{512}\,\beta^2\,\mathbf{\Delta x'y'}\,\mathbf{\Delta x'^2}\,\mathbf{\Delta xy}\,\mathbf{\Delta y'^2} - \mathbf{512}\,\beta^2\,\mathbf{\Delta x'y}\,\mathbf{\Delta x'^2}\,\mathbf{\Delta xy'}\,\mathbf{\Delta y'^2} -$$
$$\mathbf{256}\,\beta^2\,\mathbf{\Delta x'y^2}\,(\mathbf{\Delta y'^2})^2 + \mathbf{512}\,\beta^2\,\mathbf{\Delta x'y}\,\mathbf{\Delta x'y'}\,\mathbf{\Delta y'^2}\,\mathbf{\Delta yy'} +$$
$$\mathbf{256}\,\beta^2\,\mathbf{\Delta x'y'^2}\,\mathbf{\Delta yy'^2} - \mathbf{512}\,\beta^2\,\mathbf{\Delta x'^2}\,\mathbf{\Delta y'^2}\,\mathbf{\Delta yy'^2} +$$
$$\mathbf{32}\,\Delta x'y'^2\,\epsilon_g{}^2 - \mathbf{32}\,\mathbf{\Delta x'^2}\,\mathbf{\Delta y'^2}\,\epsilon_g{}^2] -$$
$$[\beta^4\,[256\,\beta^2\,\Delta x'y'^2\,\Delta xx'^2 - 512\,\beta^2\,\Delta x'y'^3\,\Delta xy + 512\,\beta^2\,\Delta x'y\,\Delta x'y'^2\,\Delta xy' -$$
$$512\,\beta^2\,\Delta x'y'\,\Delta x'^2\,\Delta xx'\,\Delta xy' +$$
$$256\,\beta^2\,(\Delta x'^2)^2\,\Delta xy'^2 + 512\,\beta^2\,\Delta x'y'\,\Delta x'^2\,\Delta xy\,\Delta y'^2 - 512\,\beta^2\,\Delta x'y\,\Delta x'^2\,\Delta xy'\,\Delta y'^2 -$$
$$256\,\beta^2\,\Delta x'y^2\,(\Delta y'^2)^2 + 512\,\beta^2\,\Delta x'y\,\Delta x'y'\,\Delta y'^2\,\Delta yy' +$$
$$256\,\beta^2\,\Delta x'y'^2\,\Delta yy'^2 - 512\,\beta^2\,\Delta x'^2\,\Delta y'^2\,\Delta yy'^2 + 32\,\Delta x'y'^2\,\epsilon_g{}^2 - 32\,\Delta x'^2\,\Delta y'^2\,\epsilon_g{}^2]^2 -$$

$$4\,\beta^4\,\Delta y'^2\,[-256\,\Delta x'y'^2 + 256\,\Delta x'^2\,\Delta y'^2]\,[-256\,\beta^4\,\Delta x'y'^2\,\Delta x'^2\,\Delta xy^2 +$$

$$512\,\beta^4\,\Delta x'y\,\Delta x'y'\,\Delta x'^2\,\Delta xy\,\Delta xy' - 256\,\beta^4\,\Delta x'y^2\,\Delta x'^2\,\Delta xy'^2 +$$

$$256\,\beta^4\,\Delta x'y^2\,\Delta xx'^2\,\Delta y'^2 - 512\,\beta^4\,\Delta x'y^2\,\Delta x'y'\,\Delta xy\,\Delta y'^2 -$$

$$512\,\beta^4\,\Delta x'y\,\Delta x'^2\,\Delta xx'\,\Delta xy\,\Delta y'^2 + 256\,\beta^4\,(\Delta x'^2)^2\,\Delta xy^2\,\Delta y'^2 +$$

$$512\,\beta^4\,\Delta x'y^3\,\Delta xy'\,\Delta y'^2 -$$

$$512\,\beta^4\,\Delta x'y\,\Delta x'y'\,\Delta xx'^2\,\Delta yy' + 1024\,\beta^4\,\Delta x'y\,\Delta x'y'^2\,\Delta xy\,\Delta yy' +$$

$$512\,\beta^4\,\Delta x'y'\,\Delta x'^2\,\Delta xx'\,\Delta xy\,\Delta yy' -$$

$$1024\,\beta^4\,\Delta x'y^2\,\Delta x'y'\,\Delta xy'\,\Delta yy' + 512\,\beta^4\,\Delta x'y\,\Delta x'^2\,\Delta xx'\,\Delta xy'\,\Delta yy' -$$

$$512\,\beta^4\,(\Delta x'^2)^2\,\Delta xy\,\Delta xy'\,\Delta yy' - 512\,\beta^4\,\Delta x'y'\,\Delta x'^2\,\Delta xy\,\Delta yy'^2 +$$

$$512\,\beta^4\,\Delta x'y\,\Delta x'^2\,\Delta xy'\,\Delta yy'^2 +$$

$$256\,\beta^4\,\Delta x'y^2\,\Delta y'^2\,\Delta yy'^2 - 512\,\beta^4\,\Delta x'y\,\Delta x'y'\,\Delta yy'^3 +$$

$$256\,\beta^4\,\Delta x'^2\,\Delta yy'^4 +$$

$$32\,\beta^2\,\Delta x'y^2\,\Delta y'^2\,\epsilon_g{}^2 - 64\,\beta^2\,\Delta x'y\,\Delta x'y'\,\Delta yy'\,\epsilon_g{}^2 +$$

$$32\,\beta^2\,\Delta x'^2\,\Delta yy'^2\,\epsilon_g{}^2 + \Delta x'^2\,\epsilon_h{}^4]^{0.5}]/$$

$$(\beta^4\,\mathbf{\Delta y'^2\,[-256\,\Delta x'y'^2 + 256\,\Delta x'^2\,\Delta y'^2]})$$

Appendix D

GBI Equipment Settings for Acquiring Gated Images

Table D.1: GBI Equipment and Software Settings for GBI Measurements at 45°

CCD camera settings	Lens aperature	f/1.4
ST-138 Controller	CCD temperature	-30°C
PG-200 Voltage P/S	Gate width	200μs
	Voltage for 100 ns/1μs gates	900/700 V
	Mode	"Gate"
	Brightness Control	"Off"
GBI Settings	Gate width	100 ns /1μs
	Int. Voltage	-350 V
	Proximity focusing voltage	3000 V
IPLAB Settings	Mode	"Synchronous"
	Exposure	200μs
	Timing	"Ext. Sync"
	Cleans	"Continuous"
	Region	384 x 576 pixels
	A/D Converter	"16 bit 50 kHz readout"
	CCD Selected	"Electronic 3 ph"
Small Recirculator	Bending dipole volt.	±6570 V
	Vacuum	4E-7 torr

Table D.2: GBI Measurement Parameters at 45°

Date of Measurement	7/22/97
GBI Position (descriptive)	after 45° bend
GBI Postion (cm along beamline)	670 cm
Measured after Quad No.	13
Beamlet drift distance	15 cm
Camera field of view	2.07 by 3.11 cm
Lens Extension needed	32 mm

Table D.3: 45° Integrated image file information

Image File name	Gate start time (μs)	Comments
"72297.ras"	no gate	integrated image
"722971us132.ras"	13.2	flattop image
"722971us142.ras"	14.2	flattop image
"722971us152.ras"	15.2	flattop image
"Q8CCWQ13CW132.ras"	13.2	rotated quadrupole error
"Q13CW132.ras"	13.2	rotated quadrupole error
"Q13CCW132.ras"	13.2	rotated quadrupole error
"Q8CCW132.ras"	13.2	rotated quadrupole error

Table D.4: 45° 100 ns gated image file information

Image Filename	Gate start (μs)	Image Filename	Gate start (μs)	Image Filename (μs)	Gate start
"72297117.ras"	11.7	"72297133.ras"	13.3	"72297149.ras"	14.9
"72297118.ras"	11.8	"72297134.ras"	13.4	"72297150.ras"	15.0
"72297119.ras"	11.9	"72297135.ras"	13.5	"72297151.ras"	15.1
"72297120.ras"	12.0	"72297136.ras"	13.6	"72297152.ras"	15.2
"72297121.ras"	12.1	"72297137.ras"	13.7	"72297153.ras"	15.3
"72297122.ras"	12.2	"72297138.ras"	13.8	"72297154.ras"	15.4
"72297123.ras"	12.3	"72297139.ras"	13.9	"72297155.ras"	15.5
"72297124.ras"	12.4	"72297140.ras"	14.0	"72297156.ras"	15.6
"72297125.ras"	12.5	"72297141.ras"	14.1	"72297157.ras"	15.7
"72297126.ras"	12.6	"72297142.ras"	14.2	"72297158.ras"	15.8
"72297127.ras"	12.7	"72297143.ras"	14.3	"72297159.ras"	15.9
"72297128.ras"	12.8	"72297144.ras"	14.4	"72297160.ras"	16.0
"72297129.ras"	12.9	"72297145.ras"	14.5	"72297161.ras"	16.1
"72297130.ras"	13.0	"72297146.ras"	14.6	"72297162.ras"	16.2
"72297131.ras"	13.1	"72297147.ras"	14.7	"72297163.ras"	16.3
"72297132.ras"	13.2	"72297148.ras"	14.8	"72297164.ras"	16.4

Appendix E

Simulation and Data Analysis Codes

The following are the codes used in the course of this work, to simulate the Small Recirculator beam using the moment evolution equations, to find the optimized values of the strength and rotation angle of the quadrupoles in the beam correction system, and to perform the GBI image analysis.

E.1 "FULLAG"

"FULLAG" is a FORTRAN 77 code modified from J. Barnard's MOMENT code. Its algorithm incorporates the full set of 10 second moment evolution equations and allows the setting of either random quadrupole rotation errors or discrete errors of the user's choice. It was used to produce moment data to compare against that produced by the reduced set of 8 second moment evolution equations in their approximate form. This comparison is shown in Figure 6.7.

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c  'fullag.f'  Integrates the full set of Barnard moment evolution c
c   equations through a Small Recirculator alternating gradient    c
c   quadrupole focusing lattice.                                   c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      PROGRAM moment
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
c  *** SETUP PROGRAM ***********************************************
      EXTERNAL dfdzsub
c  *** filename characters (Aa,etc.)*********************************
      PARAMETER (nprtmax=1,nthet=100000)
c  *** allocate space for arrays*************************************
      DIMENSION g0(10,nprtmax),g1(10,nprtmax),dfdzw0(10,nprtmax)
      DIMENSION dthetatab(nthet)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2,fkbtx02,fkbty02,pervo,betao,pervpower
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab
      COMMON/blk31/iprint,nprint
            nprintsum=0
            ilattno=0
            jfodo=1
            OPEN(21,file='q2ml.dat')
            OPEN(50,file='7_out')
            OPEN(51,file='goodmom')
c *** Initialize constants  ***************************************
c***Numerical constants*****************************
            pi=3.141592654d0
            fpi=3.141592654d0
            half(pi=pi/2.d0
            fmhc2=938.d6
            fmec2=511.d3
            frpieps0=1.1127d-10
            curra=31.3d6
c***Physical data*********************************
            etaq=0.29d0
            flattic=38.1d0
            nhlp=30
            nlaps=1
            flength=nlaps*nhlp*flattic
            qcharge=1.0d0
            energyo=80.0d3
            energyf=80.0d3
            amu=39.9d0
            beta=DSQRT(2.0d0*energyo/(amu*fmhc2))
            betao=beta
            betaf=DSQRT(2.0d0*energyf/(amu*fmhc2))
            betaconl=(energyf/energyo-1.0d0)/flength
            halfbetaconl=.5d0*betaconl
            current=2.0d-3
            sigma0=72.0d0
            nstepsperhlp=200
            nprint=1
            a0=1.4165d+00
            b0=8.7542d-01
            ap0=+2.5526d-02
            bp0=-1.7423d-02
            z0=0.0d0
            emitx0=2.5169d-03
            emity0=2.4811d-03
            pervalt=2.0d0*current*qcharge/(amu*beta**3*curra)
            pervo=.000357d0
            pervf=.000353d0
             PRINT *,'BETA=',beta
c*************************************************************
c       Get a divide by zero in Pervpower if betao=betaf so for
c       a drifting beam calculate betas based on 80keV and 81keV
c*************************************************************
```



```
            pervpower=-1.81444d+00
           print *,'pervpower=',pervpower
c***initialization parameters**********************
            nprintsum=0
            ilattno=0
            jfodo=1
c***rotation error angles***************************
c***random errors***********************************
c         Quad rotation error data
            iseed=22221
            icorrelation=1
            thetamax=.0044d0
            if(icorrelation.eq.0)then
              do j=1,nhlp*nlaps
               dthetatab(j)=2.0d0*thetamax*(DBLE(ran(iseed))-0.5)
             print *, dthetatab(j)
              enddo
            endif
            if(icorrelation.eq.1)then
              do j=1,nhlp
               dthetatab(j)=2.0d0*thetamax*(DBLE(ran(iseed))-0.5)
              enddo
              do k=2,nlaps
                 do j=1,nhlp
                  dthetatab(k-1)*nhlp+j)=dthetatab(j)
             print *, dthetatab(j)
                 enddo
              enddo
            endif
c***setable errors*************************************
           thetamax=1.0d0*0.00873d0
           do j=1,nhlp
           dthetatab(j)=0.0
           enddo
c          dthetatab(2)=-2.0*thetamax
c          dthetatab(3)=1.0*thetamax
c          dthetatab(4)=-0.5*thetamax
c          dthetatab(5)=-0.35*thetamax
c         etc, etc....
c*************************************************
            thetasum=0.0d0
            theta2sum=0.0d0
            nlatt=nhlp*nlaps
            do j=1,nlatt
               thetasum=thetasum+ dthetatab(j)
               theta2sum=theta2sum+ dthetatab(j)**2
            enddo
            thetaavg=thetasum/nlatt
            theta2avg=theta2sum/nlatt
            dthetarms=DSQRT(theta2avg-thetaavg**2)
            dthetarmstheory=2.*thetamax/3.
            ilattno=1
            costwotheta=1.0d0
            sintwotheta=0.0d0
           dalpha0=dthetatab(1)
           dz0=flattic/float(nstepsperhlp)
           epsdyinit=0.0d-5
           dx2b0=a0**2/4.0d0
           dy2b0=b0**2/4.0d0
           dxxpb0=ap0*a0/4.0d0
           dyypb0=bp0*b0/4.0d0
           dxp2b0=(emitx0**2/16.0d0+ dxxpb0**2)/dx2b0
           dyp2b0=(emity0**2/16.0d0+ dyypb0**2)/dy2b0
           dx20=(1.0d0+epsdyinit)*(dx2b0*DCOS(dalpha0)**2+
     .          dy2b0*DSIN(dalpha0)**2)
           dxxp0=dxxpb0*DCOS(dalpha0)**2+dyypb0*DSIN(dalpha0)**2
           dxp20=(dxp2b0*DCOS(dalpha0)**2+dyp2b0**2*DSIN(dalpha0)**2)
           dy20=(1.0d0-epsdyinit)*(dy2b0*DCOS(dalpha0)**2+
     .          dx2b0*DSIN(dalpha0)**2)
           dyyp0=dyypb0*DCOS(dalpha0)**2+dxxpb0*DSIN(dalpha0)**2
           dyp20=(dyp2b0*DCOS(dalpha0)**2+dxp2b0**2*DSIN(dalpha0)**2)
           dxy0=(dx2b0-dy2b0)*DSIN(dalpha0)*DCOS(dalpha0)
           dxpyp0=(dxp2b0-dyp2b0)*DSIN(dalpha0)*DCOS(dalpha0)
```

```
            dxyp0=(dxxpb0-dyypb0)*DSIN(dalpha0)*DCOS(dalpha0)
            dxpy0=(dxxpb0-dyypb0)*DSIN(dalpha0)*DCOS(dalpha0)
            dx2=dx20
            dxxp=dxxp0
            dxp2=dxp20
            dy2=dy20
            dyyp=dyyp0
            dyp2=dyp20
            dxy=dxy0
            dxpy=dxpy0
            dxyp=dxyp0
            dxpyp=dxpyp0
c *** Setup fodo stuff ==========================================
c *** Initial dx^2*k_beta^2(including space charge) ==============
            z=a
            dz=dz0
            IF (jfodo.EQ.1) THEN
c   jfend is the value of j at end of focusing magnet
c   focusing magnet has values of j between 0 and jfend
c   first drift                                jfend and jdbeg
c   defocusing magnet                          jdbeg and jdend
c   second drift                               jdend and jlatt
                jfend=AINT(etaq*flattic/dz0 +.5)
                IF (jfend.LT.1) jfend=1
                dzmag=etaq*flattic/jfend
                jdrift=AINT((1.0d0-etaq)*flattic/dz0+.5)
                IF (jdrift.LT.1)jdrift=1
                dzdrift=(1.0d0-etaq)*flattic/jdrift
                jdbeg=jfend+jdrift
                jdend=jdbeg+jfend
                jlatt=jdend+jdrift
                nz=jdbeg*nhlp*nlaps
                sigma0rad=sigma0*(pi/180.)
               fkbet2=DSQRT(2.*(1.-DCOS(sigma0rad))/(1.-2.*etaq/3.)/
                        (etaq*flattic**2)
         print *,'nz=',nz, '   fkbet2=',fkbet2
                fkfodo=fkbet2**(.5)
                fkbtx02=fkbet2
                fkbty02=-fkbet2
            ENDIF
c
c *** write headings of emitout files ============================
c
c ***  initialize and write down beginning values ================
                z=z0
                g0(1,1)=dx2
                g0(2,1)=dxxp
                g0(3,1)=dxp2
                g0(4,1)=dy2
                g0(5,1)=dyyp
                g0(6,1)=dyp2
                g0(7,1)=dxy
                g0(8,1)=dxpy
                g0(9,1)=dxyp
                g0(10,1)=dxpyp
c  *** Call print here to get the first line with z=z0  ============
                 CALL prntsub(xsum,ysum,xpsum,ypsum,z,npart)
                 nprintsum=nprintsum+1
                 iprint=1
                 isnap=1
                 jstep=0
c ****************************************************************
c ********************** MAIN Z-LOOP ****************************
c ****************************************************************
                do 1000 iz=1,nz
                    IF (jfodo.EQ.1)CALL fodostep
c *** Advance all particles ==================================================
                    CALL dfdzsub(z,g0,dfdzw0)
                    CALL fint(g0,g1,z,dz,dfdzsub,dfdzw0)
                         z=z+dz
                        dx2=g1(1,1)
                        dxxp=g1(2,1)
                        dxp2=g1(3,1)
```



```
                        dy2=g1(4,1)
                        dyyp=g1(5,1)
                        dyp2=g1(6,1)
                        dxy=g1(7,1)
                        dxpy=g1(8,1)
                        dxyp=g1(9,1)
                        dxpyp=g1(10,1)
c
                        do 400 icoord=1,10
                           g0(icoord,1)=g1(icoord,1)
  400                   continue
c *** Printing loop ==========================================
c                       IF (iprint.EQ.nprint) THEN
c                          IF (z .ge. 21412.2) THEN
                         CALL prntsub(z)
                         nprintsum=nprintsum+1
                         iprint=1
                        ENDIF
 1000                   CONTINUE

       print *,'dthetarms=',dthetarms,' thetaavg=',thetaavg
       print *,'dthetarmstheory=',dthetarmstheory,
     .         ' thetaavgtheory=',0.0d0
       print *,nprintsum,' points in z in each history plot.'

       CLOSE(21)
       CLOSE(50)
       CLOSE(51)
       CALL EXIT(1)
       END
c*****************************************************************
c *** Printing routine ========================================
c*****************************************************************
c
       SUBROUTINE prntsub(z)
       PARAMETER (nprtmax=1, nthet=100000)
       IMPLICIT DOUBLE PRECISION (a-h,o-z)
       DIMENSION dthetatab(nthet)
       common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
       common/blk2/fkbet2,fkbtx02,fkbty02,pervo,betao,pervpower
       common/blk3/betaconl,half(betaconl,betaovbetao,dbdzovb
       COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
       COMMON/blk12/dz,dz0,dzpre
       COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
      .llattno,dthetatab
       xav=0.
       yav=0.
       xpav=0.
       ypav=0.
       call betavals(z)
       beta=betao*betaovbetao
       dalpha=.5*DATAN(2*dxy/(dx2-dy2))
       cosalpha=DCOS(dalpha)
       sinalpha=DSIN(dalpha)
       cosalpha2=cosalpha**2
       sinalpha2=sinalpha**2
       costwoalpha=DCOS(2.*dalpha)
       sintwoalpha=DSIN(2.*dalpha)
       transenergy=.5*(dxp2 + dyp2
     .      - fkbeta02x(z)*( (dx2-dy2)*costwotheta +
     .                          2.0d0*dxy*sintwotheta )
     .      - pervo*DLOG(
     .                     DSQRT(dx2*cosalpha2 +
     .                          dy2*sinalpha2 +
     .                          dxy*sintwoalpha ) +
     .                     DSQRT(dy2*cosalpha2 +
     .                          dx2*sinalpha2 -
     .                          dxy*sintwoalpha )  ) )
       dxb2=dx2*cosalpha2+dy2*sinalpha2+2*dxy*sinalpha*cosalpha
       dyb2=dy2*cosalpha2+dx2*sinalpha2-2*dxy*sinalpha*cosalpha
       sbeam=2*DSQRT(dxb2)
       bbeam=2*DSQRT(dyb2)
       angmom=dxyp-dxpy
```

```
      emitnx2=16.0d0*beta**2*(dx2*dxp2-dxxp**2)
      emitnx=SQRT(emitnx2)
      emitny2=16.0d0*beta**2*(dy2*dyp2-dyyp**2)
      emitny=SQRT(emitny2)
      emitng2=.5*(emitnx2+emitny2)+
     .         16.*beta**2*(dxy*dxpyp-dxyp*dxpy)
      emitng=SQRT(emitng2)
      emitnh= (emitnx2*emitny2 + (4.0d0*beta)**4*(dxpyp**2*dxy**2 -
     .   2*dxpy*dxpyp*dxy*dxyp +
     .   dxpy**2*dxyp**2 + 2*dxpy*dxxp*dxy*dyp2 -
     .   dxp2*dxy**2*dyp2 - dxpy**2*dx2*dyp2 -
     .   2*dxpyp*dxxp*dxy*dyyp - 2*dxpy*dxxp*dxyp*dyyp +
     .   2*dxp2*dxy*dxyp*dyyp +
     .   2*dxpy*dxpyp*dx2*dyyp + 2*dxpyp*dxxp*dxyp*dy2 -
     .   dxp2*dxyp**2*dy2 - dxpyp**2*dx2*dy2))**0.25
      fnp=1.0d0
      dum=1.0d-8
        IF (z .GE. 21412.0 .AND. z .LE. 21412.4) then
            WRITE(50,*) "z=",z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,
     .  dxpyp
        ELSEIF (z .GE. 21564.2) then
            WRITE(50,*) "z=",z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,
     .  dxpyp

c           WRITE(51,*) dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
c           print *,"z=",z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
        ENDIF
711          FORMAT(2(1x,1pe11.4))
715          FORMAT(11(1x,f12.8))
        ddx2=dx2*1e2
        ddxxp=dxxp*1e4
        ddxp2=dxp2*1e6
        ddy2=dy2*1e2
        ddyyp=dyyp*1e4
        ddyp2=dyp2*1e6
        ddxy=dxy*1e2
        ddxpy=dxpy*1e4
        ddxyp=dxyp*1e4
        ddxpyp=dxpyp*1e6
        eemitnx=emitnx*1e4
        eemitny=emitny*1e4
        aangmom=angmom*1e4
        eemitng=emitng*1e4
        eemitnh=emitnh*1e4
            WRITE(21,*) z,ddx2,ddxxp,ddxp2,ddy2,ddyyp,ddyp2,ddxy,
     .  ddxpy,ddxyp,ddxpyp,eemitnx,eemitny,aangmom,eemitng,eemitnh
      RETURN
      END
c
c
c*****************************************************************
c *** Integration routine ***************************************
c*****************************************************************
c *** This subroutine is adapted from the subroutine RK4 from  *******
c *** Numerical Recipes by Press et al. The main difference is the ***
c *** use of two dimensional arrays rather than linear arrays.  ******
c *** Input variables:                                         ******
c ***  Y(ieqn,it):   The 2-D array of dependent variables      ******
c ***    DYDX(ieqn,it): Initial derivatives calculated in calling *****
c ***    subroutine.                                            *****
c ***    ieqnmax,nprtmax: Maximum dimensions of the arrays      *****
c ***    X: The independent variable                            *****
c ***    H: The step size in X.                                 *****
c ***    DERIVS: The name of the subroutine with the derivatives. *****
c *** Output variables:                                         *****
c ***    YOUT(ieqn,it): The new values of Y at X+H              *****
c*****************************************************************
      SUBROUTINE fint(Y,YOUT,X,H,DERIVS,DYDX)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(ieqnlim=10,nprtmax=1)
      EXTERNAL DERIVS
      DIMENSION Y(ieqnlim,nprtmax),DYDX(ieqnlim,nprtmax),
     .     YOUT(ieqnlim,nprtmax)
```



```
      DIMENSION YT(ieqnlim,nprtmax),DYT(ieqnlim,nprtmax),
     .    DYM(ieqnlim,nprtmax)
      ieqnmax=10
      itmax=nprtmax
      HH=H*0.5
      H6=H/6.
      XH=X+HH
      XF=X+H
      do 110 it=1,itmax
         DO 11 ieqn=1,ieqnmax
            YT(ieqn,it)=Y(ieqn,it)+HH*DYDX(ieqn,it)
11       CONTINUE
110   CONTINUE
      CALL DERIVS(XH,YT,DYT)
      do 120 it=1,itmax
         DO 12 ieqn=1,ieqnmax
            YT(ieqn,it)=Y(ieqn,it)+HH*DYT(ieqn,it)
12       CONTINUE
120   CONTINUE
      CALL DERIVS(XH,YT,DYM)
      do 130 it=1,itmax
         DO 13 ieqn=1,ieqnmax
            YT(ieqn,it)=Y(ieqn,it)+H*DYM(ieqn,it)
            DYM(ieqn,it)=DYT(ieqn,it)+DYM(ieqn,it)
13       CONTINUE
130   CONTINUE
      CALL DERIVS(XF,YT,DYT)
      do 140 it=1,itmax
         DO 14 ieqn=1,ieqnmax
            YOUT(ieqn,it)=Y(ieqn,it)
     .          +H6*(DYDX(ieqn,it)+DYT(ieqn,it)+2.*DYM(ieqn,it))
14       CONTINUE
140   CONTINUE
      RETURN
      END


c*****************************************************************
c *** Runge-kutta integration routine ***************************
c*****************************************************************

      SUBROUTINE dfdzsub(zw,fw,dfdzw)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(nprtmax=1,nthet=100000)
      DIMENSION fw(10,nprtmax),dfdzw(10,nprtmax)
      DIMENSION dthetatab(nthet)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2,fkbtx02,fkbty02,pervo,betao,pervpower
      common/blk3/betaconi,halfbetaconi,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab
            dx2=fw(1,1)
            dxxp=fw(2,1)
            dxp2=fw(3,1)
            dy2=fw(4,1)
            dyyp=fw(5,1)
            dyp2=fw(6,1)
            dxy=fw(7,1)
            dxpy=fw(8,1)
            dxyp=fw(9,1)
            dxpyp=fw(10,1)
            call betavals(zw)
c            perv=pervo/(betaovbetao**2)  (constant bunchlength)
            perv=pervo*betaovbetao**pervpower
            alpha=.5*DATAN(2*dxy/(dx2-dy2))
            cosalph=DCOS(alpha)
            sinalph=DSIN(alpha)
            cos2alpha=cosalph**2
            sin2alpha=sinalph**2
            sinalphcosalph=sinalph*cosalph
            dxb2=dx2*cos2alpha+dy2*sin2alpha+2*dxy*sinalphcosalph
```

```
            dyb2=dy2*cos2alpha+dx2*sin2alpha-2*dxy*sinalphcosalph
            dxbdyb=SQRT(dxb2*dyb2)
            fksxb=.5*perv/(dxb2+dxbdyb)
            fksyb=.5*perv/(dyb2+dxbdyb)
            fksxx=fksxb*cos2alpha+fksyb*sin2alpha
            fksxy=(fksxb-fksyb)*sinalphcosalph
            fksyx=fksxy
            fksyy=fksyb*cos2alpha+fksxb*sin2alpha
            fkqxx=fkbeta02x(zw)*costwotheta/betaovbetao
            fkqyy=fkbeta02y(zw)*costwotheta/betaovbetao
            fkqxy=fkbeta02x(zw)*sintwotheta/betaovbetao
            fkqyx=fkqxy
            fkxx=fkqxx+fksxx
            fkxy=fkqxy+fksxy
            fkyx=fkqyx+fksyx
            fkyy=fkqyy+fksyy
c dx2'
               dfdzw(1,1)= 2*fw(2,1)
c dxxp'=
               dfdzw(2,1)=fw(3,1) + fkxx*fw(1,1) +
     .                              fkxy*fw(7,1) - dbdzovb*fw(2,1)
c dxp2'=
               dfdzw(3,1)= 2.0d0*fkxx*fw(2,1)+
     .                     2.0d0*fkxy*fw(8,1) - 2.0d0*dbdzovb*fw(3,1)
c dy2'=
               dfdzw(4,1)= 2*fw(5,1)
c dyyp'=
               dfdzw(5,1)= fw(6,1)+ fkyy*fw(4,1) +
     .                              fkyx*fw(7,1)- dbdzovb*fw(5,1)
c dyp2'=
               dfdzw(6,1)= 2.0d0*fkyy*fw(5,1)+
     .                     2.0d0*fkxy*fw(9,1) - 2.0d0*dbdzovb*fw(6,1)
c dxy'
               dfdzw(7,1)=  fw(8,1)+fw(9,1)
c dxpy'=
               dfdzw(8,1)= fw(10,1)+fkxx*fw(7,1)+
     .                              fkxy*fw(4,1)- dbdzovb*fw(8,1)
c dxyp'=
               dfdzw(9,1)=  fw(10,1)+fkyy*fw(7,1)+
     .                              fkyx*fw(1,1)- dbdzovb*fw(9,1)
c dxpyp'=
               dfdzw(10,1)= fkxx*fw(9,1) + fkxy*fw(5,1) +
     .                      fkyy*fw(8,1) + fkyx*fw(2,1)
     .                      - 2.0d0*dbdzovb*fw(10,1)
c
      RETURN
      END
c
c
c
c*****************************************************************
c *** fodo routine ***************************************
c*****************************************************************
      SUBROUTINE fodostep
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(nthet=100000)
      DIMENSION dthetatab(nthet)
      common/blk2/fkbet2,fkbtx02,fkbty02,pervo,betao,pervpower
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab
      COMMON/blk31/iprint,nprint
      IF (0.LE.jstep.and.jstep.LT.jfend) THEN
         dz=dzmag
         fkbtx02=fkbet2
         fkbty02=-fkbet2
         jstep=jstep+1
         RETURN
      ENDIF
      IF (jstep.EQ.jdbeg)THEN
         ilattno=ilattno+1
```



```
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         if(iprint.eq.nprint)
     .      print *,'Quad no.=',ilattno,' dtheta=',dtheta
      ENDIF
      IF (jdbeg.LE.jstep.and.jstep.LT.jdend) THEN
         dz=dzmag
         fkbtx02=-fkbet2
         fkbty02=fkbet2
         jstep=jstep+1
         RETURN
      ENDIF
      IF (jstep.GE.jlatt) THEN
         dz=dzmag
         fkbtx02=fkbet2
         fkbty02=-fkbet2
         jstep=1
         ilattno=ilattno+1
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         if(iprint.eq.nprint)
     .      print *,'Quad no.=',ilattno,' dtheta=',dtheta
         RETURN
      ENDIF
      dz=dzdrift
      fkbtx02=0.0d0
      fkbty02=0.0d0
      jstep=jstep+1
      RETURN
      END
c
c
c
      FUNCTION fkbeta02x(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2,fkbtx02,fkbty02,pervo,betao,pervpower
      fkbeta02x=-fkbtx02
      RETURN
      END
c
c
c
      FUNCTION fkbeta02y(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2,fkbtx02,fkbty02,pervo,betao,pervpower
      fkbeta02y=-fkbty02
      RETURN
      END
c
c
c
      SUBROUTINE betavals(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
c betaconl=(energyf/energyo-1.0d0)/(nlaps*flength)
c halfbetaconl=.5*betaconl
      betaovbetao=DSQRT(1.0d0 + betaconl*z)
      dbdzovb=halfbetaconl/(1.0d0 + betaconl*z)
      return
      end
```

E.2 "REDUCEDAG"

"REDUCEDAG" is a FORTRAN 77 code modified from J. Barnard's MOMENT code. Its algorithm incorporates the reduced set of 8 second moment evolution equations by substituting in the approximate explicit solutions for Δx^2 and Δy^2 (Equations 6.11 and 6.16) wherever they occur in the moment evolution equations. The code allows the setting of either random quadrupole rotation errors or discrete errors of the user's choice. It was used to produce moment data to compare against that produced by the set of 10 second moment evolution equations of "FULLAG". This comparison is shown in Figure 6.7.

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                      c
c  reducedag.f:  Integrates a reduced set of Barnard moment            c
c  evolution equations through an alternating gradient quadrupole     c
c  focusing lattice.  The set of equations were reduced by Hopkins    c
c  from 10 to 8 by applying the generalized emittance constraints.    c
c  dx2 and dy2 are calculated using a simplified form of the          c
c  constraint equations.  Quad strength is calculated using B'/[Bp]   c
c  instead of the Lee, et. al. equation relating sigma0 to K.          c
c                                                                      c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      PROGRAM moment
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
c  *** SETUP PROGRAM ***********************************************
      EXTERNAL dfdzsub
c  *** filename characters (ha,etc.)*******************************
      PARAMETER (nprtmax=1,nthet=1000)
c  *** allocate space for arrays***********************************
      DIMENSION q0(10,nprtmax),q1(10,nprtmax),dfdzw0(10,nprtmax)
      DIMENSION fkfodo(nthet)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2(566),fkbtx02,fkbty02,pervo,betao,
     .            pervpower
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab(566)
      COMMON/blk31/iprint,nprint,quadfld(566)
      COMMON/blk40/emitg2,emith2,beta
            nprintsum=0
            ilattno=0
            jfodo=1
            OPEN(20,file='z_dx2.dat')
            OPEN(21,file='q2m1reduced.dat')
            OPEN(50,file='7_out')
c *** Initialize constants  ***************************************
c***Numerical constants*****************************
            pi=3.141592654d0
            fpi=3.141592654d0
            halfpi=pi/2.d0
            fmhc2=938.d6
            fmec2=511.d3
            frpieps0=1.1127d-10
            curra=31.3d6
c***Physical data*********************************
            etaq=0.29d0
            flattic=38.1d0
            nhlp=566
            nlaps=1
            flength=nlaps*nhlp*flattic
            qcharge=1.0d0
            energyo=80.0d3
            energyf=80.0d3
            amu=39.9d0
            beta=DSQRT(2.0d0*energyo/(amu*fmhc2))
            betao=beta
            betaf=DSQRT(2.0d0*energyf/(amu*fmhc2))
            betaconl=(energyf/energyo-1.0d0)/flength
            halfbetaconl=.5d0*betaconl
            current=2.0d-3
            sigma0=72.0d0
            nstepsperhlp=200
            nprint=1
            a0=1.4165d+00
            b0=8.7542d-01
            ap0=-2.5526d-02
            bp0=-1.7423d-02
            z0=0.0d0
             quad=0.25258d0
             brho=25.798d0
             rpipe=1.149d0
            emitx0=2.5169d-03
```



```
            emity0=2.4811d-03
            pervalt=2.0d0*current*qcharge/(amu*beta**3*curra)
            pervo=.000357d0
            pervf=.000353d0
             PRINT *,'BETA=',beta
c*****************************************************************
c       Get a divide by zero in Pervpower if betao=betaf so for
c      a drifting beam calculate betas based on 80keV and 81keV
c*****************************************************************
             pervpower=-1.81444d+00
            print *,'pervalt=',pervalt,' pervo=',pervo
            print *,'pervpower=',pervpower
c***initialization parameters*********************
             nprintsum=0
             ilattno=0
             jfodo=1
c***setable rotation angles and field strengths***************
            thetamax=0.017453)3d0
            DO j=1,nhlp
            dthetatab(j)=0.0
            quadfld(j)=quad
            ENDDO
c           dthetatab(1)=-1.0*thetamax
c           dthetatab(2)=-1.0*thetamax
c           dthetatab(3)=1.0*thetamax
c           dthetatab(4)=-0.5*thetamax
c           dthetatab(5)=-0.35*thetamax
c           dthetatab(7)=1.0*thetamax
c       etc, etc...
c           quadfld(2)=0.28d0
c           quadfld(3)=0.28d0
c           quadfld(4)=0.28d0
c           quadfld(5)=0.28d0
c       etc, etc...
c*****************************************************
c***random errors*********************************
c         Quad rotation error data
             iseed=22221
             icorrelation=1
             thetamax=.0044d0
             if(icorrelation.eq.0)then
               do j=1,nhlp*nlaps
                dthetatab(j)=2.0d0*thetamax*(DBLE(ran(iseed))-0.5)
              print *, dthetatab(j)
               enddo
             endif
             if(icorrelation.eq.1)then
               do j=1,nhlp
               dthetatab(j)=2.0d0*thetamax*(DBLE(ran(iseed))-0.5)
               enddo

              do k=2,nlaps
                  do j=1,nhlp
                   dthetatab((k-1)*nhlp+j)=dthetatab(j)
                  enddo
               enddo
             endif
         print *,'Line 143'
c*****************************************************
             thetasum=0.0d0
             theta2sum=0.0d0
             nlatt=nhlp*nlaps
             do j=1,nlatt
                thetasum=thetasum+ dthetatab(j)
                theta2sum=theta2sum+ dthetatab(j)**2
             enddo
             thetasvg=thetasum/nlatt
             theta2avg=theta2sum/nlatt
             dthetarms=DSQRT(theta2avg-thetaavg**2)
             dthetarmstheory=2.*thetamax/3.
             ilattno=1
             costwotheta=1.0d0
             sintwotheta=0.0d0
```

```
            dalpha0=dthetatab(1)
            dz0=flattic/float(nstepsperhlp)
            epsdyinit=0.0d-5
            dx2b0=a0**2/4.0d0
            dy2b0=b0**2/4.0d0
            dxxpb0=ap0*a0/4.0d0
            dyypb0=bp0*b0/4.0d0
            dxp2b0=(emitx0**2/16.0d0+ dxxpb0**2)/dx2b0
            dyp2b0=(emity0**2/16.0d0+ dyypb0**2)/dy2b0
            dx20=(1.0d0+epsdyinit)*(dx2b0*DCOS(dalpha0)**2+
                dy2b0*DSIN(dalpha0)**2)
            dxxp0=dxxpb0*DCOS(dalpha0)**2+dyypb0*DSIN(dalpha0)**2
            dxp20=(dxp2b0*DCOS(dalpha0)**2+dyp2b0**2*DSIN(dalpha0)**2)
            dy20=(1.0d0-epsdyinit)*(dy2b0*DCOS(dalpha0)**2+
                dx2b0*DSIN(dalpha0)**2)
            dyyp0=dyypb0*DCOS(dalpha0)**2+dxxpb0*DSIN(dalpha0)**2
            dyp20=(dyp2b0*DCOS(dalpha0)**2+dxp2b0**2*DSIN(dalpha0)**2)
            dxy0=(dx2b0-dy2b0)*DSIN(dalpha0)*DCOS(dalpha0)
            dxpyp0=(dxp2b0-dyp2b0)*DSIN(dalpha0)*DCOS(dalpha0)
            dxyp0=(dxxpb0-dyypb0)*DSIN(dalpha0)*DCOS(dalpha0)
            dxpy0=(dxxpb0-dyypb0)*DSIN(dalpha0)*DCOS(dalpha0)
            dx2=dx20
            dxxp=dxxp0
            dxp2=dxp20
            dy2=dy20
            dyyp=dyyp0
            dyp2=dyp20
            dxy=dxy0
            dxpy=dxpy0
            dxyp=dxyp0
            dxpyp=dxpyp0
        write(50,*) z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
c*****************************************************************
c *** Calculate the generalized emittances based on the above******
c*****************************************************************
c *** generalized emittance constraint equations
c
        emitg2=EQ(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
        emith2=EH(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
c
c*****************************************************************
c *** Setup fodo stuff ***************************************
c *** Initial dx^2*k_beta^2(including space charge) ***************
            z=a
            dz=dz0
            IF (jfodo.EQ.1) THEN
c   jfend is the value of j at end of focusing magnet
c   focusing magnet has values of j between 0 and jfend
c   first drift                          jfend and jdbeg
c   defocusing magnet                    jdbeg and jdend
c   second drift                         jdend and jlatt
              jfend=AINT(etaq*flattic/dz0 +.5)
              IF (jfend.LT.1) jfend=1
              dzmag=etaq*flattic/jfend
              jdrift=AINT((1.0d0-etaq)*flattic/dz0+.5)
              IF (jdrift.LT.1)jdrift=1
              dzdrift=(1.0d0-etaq)*flattic/jdrift
              jdbeg=jfend+jdrift
              jdend=jdbeg+jfend
              jlatt=jdend+jdrift
c               nz=(3.+etaq)*nstepsperhlp+1
              nz=jdbeg*nhlp*nlaps
c               nz=113200
               PRINT *, 'nz=',nz
              sigma0rad=sigma0*(fpi/180.)
               DO j=1,nhlp
                fkbet2(j)=quadfld(j)/(rpipe*brho)
              fkfodo(j)=fkbet2(j)**(.5)
               ENDDO
               print *,'nz=',nz,'    fkbet2=',fkbet2
c              fkbtx02=fkbet2
c              fkbty02=-fkbet2
            ENDIF
```



```
c
c ***  initialize and write down beginning values ****************
                    z=z0
                    g0(1,1)=dx2
                    g0(2,1)=dxxp
                    g0(3,1)=dxp2
                    g0(4,1)=dy2
                    g0(5,1)=dyyp
                    g0(6,1)=dyp2
                    g0(7,1)=dxy
                    g0(8,1)=dxpy
                    g0(9,1)=dxyp
                    g0(10,1)=dxpyp
C  *** Call print here to get the first line with z=z0  *************
                     CALL prntsub(xsum,ysum,xpsum,ypsum,z,npart)
                     nprintsum=nprintsum+1
                     iprint=1
                     isnap=1
                     jstep=0
        print *,z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
c ***************************************************************
c ********************** MAIN Z-LOOP ****************************
c ***************************************************************
                  do 1000 iz=1,nz
c       print *,iz
                       IF (jfodo.EQ.1)CALL fodostep
c *** Advance all particles ***************************************
                       CALL dfdzsub(z,g0,dfdzw0)
                       CALL fint(g0,g1,z,dz,dfdzsub,dfdzw0)
                           dxxp=g1(2,1)
                           dxp2=g1(3,1)
                           dyyp=g1(5,1)
                           dyp2=g1(6,1)
                           dxy=g1(7,1)
                           dxpy=g1(8,1)
                           dxyp=g1(9,1)
                           dxpyp=g1(10,1)
                dx2=DEEX2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
                dy2=DEEY2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
        IF (z .GE. 21412 .and. z .lt. 21412.4) then
            WRITE(50,*) 'z=',z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,
     .  dxpyp
        ENDIF
        IF (z .GE. 21564.4 .and. z .lt. 21564.8) then
            WRITE(50,*) 'z=',z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,
     .  dxpyp
        ENDIF
                z=z+dz
                g1(1,1)=dx2
                g1(4,1)=dy2
                         do 400 icoord=1,10
                            g0(icoord,1)=g1(icoord,1)
  400                    continue
c *** Printing loop ***********************************************
c
       CALL prntsub(z)
 1000  CONTINUE
      print *,'dthetarms=',dthetarms,' thetaavg=',thetaavg
      print *,'dthetarmstheory=',dthetarmstheory,
     .        ' thetaavgtheory=',0.0d0
      print *,nprintsum,' points in z in each history plot.'

        CLOSE(20)
      CLOSE(21)
      CLOSE(50)
      CALL EXIT(1)
      END
c*****************************************************************
c *** Printing routine **********************************************
c*****************************************************************
c
      SUBROUTINE prntsub(z)
      PARAMETER (nprtmax=1, nthet=1000)
```

```
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2(566),fkbtx02,fkbty02,pervo,betao,
     .            pervpower
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab(566)
      xav=0.
      yav=0.
      xpav=0.
      ypav=0.
      call betavals(z)
      beta=betao*betaovbetao
      dalpha=.5*DATAN(2*dxy/(dx2-dy2))
      cosalpha=DCOS(dalpha)
      sinalpha=DSIN(dalpha)
      cosalpha2=cosalpha**2
      sinalpha2=sinalpha**2
      costwoalpha=DCOS(2.*dalpha)
      sintwoalpha=DSIN(2.*dalpha)
      transenergy=.5*(dxp2 + dyp2
     .     - fkbeta02x(z)*( (dx2-dy2)*costwotheta +
     .                      2.0d0*dxy*sintwotheta )
     .     - pervo*DLOG(
     .                  DSQRT(dx2*cosalpha2 +
     .                      dy2*sinalpha2 +
     .                      dxy*sintwoalpha ) +
     .                  DSQRT(dy2*cosalpha2 +
     .                      dx2*sinalpha2 -
     .                      dxy*sintwoalpha )  ) )
      dxb2=dx2*cosalpha2+dy2*sinalpha2+2*dxy*sinalpha*cosalpha
      dyb2=dy2*cosalpha2+dx2*sinalpha2-2*dxy*sinalpha*cosalpha
      abeam=2*DSQRT(dxb2)
      bbeam=2*DSQRT(dyb2)
      angmom=dxyp-dxpy
      emitnx2=16.0d0*beta**2*(dx2*dxp2-dxxp**2)
      emitnx=SQRT(emitnx2)
      emitny2=16.0d0*beta**2*(dy2*dyp2-dyyp**2)
      emitny=SQRT(emitny2)
      emitng2=.5*(emitnx2+emitny2)+
     .        16.*beta**2*(dxy*dxpyp-dxyp*dxpy)
      emitng=SQRT(emitng2)
      emitnh= (emitnx2*emitny2 + (4.0d0*beta)**4*(dxpyp**2*dxy**2 -
     .   2*dxpy*dxpyp*dxy*dxyp +
     .   dxpy**2*dxyp**2 + 2*dxpy*dxxp*dxy*dyp2 -
     .   dxp2*dxy**2*dyp2 - dxpy**2*dx2*dyp2 -
     .   2*dxpyp*dxxp*dxy*dyyp - 2*dxpy*dxxp*dxyp*dyyp +
     .   2*dxp2*dxy*dxyp*dyyp +
     .   2*dxpy*dxpyp*dx2*dyyp + 2*dxpyp*dxxp*dxyp*dy2 -
     .   dxp2*dxyp**2*dy2 - dxpyp**2*dx2*dy2))**0.25
      fnp=1.0d0
      dum=1.0d-8
c       IF (z .GE. 2.287d+02 .AND. z .LE. 2.2882d+02) then
c           WRITE(50,*) "z=",z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,
c     .  dxpyp
c       IF (z .GE. 3.541d+02 .AND. z .LE. 3.542d+02) then
c           WRITE(50,*) "z=",z,dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,
c     .  dxpyp
c
c       ENDIF
711         FORMAT(2(1x,1pe11.4))
 710        FORMAT(6(1x,1pe11.4))
c*The following puts everything in terms of mm and mr vice cm and rads
        ddx2=dx2*1e2
        ddxxp=dxxp*1e4
        ddxp2=dxp2*1e6
        ddy2=dy2*1e2
        ddyyp=dyyp*1e4
        ddyp2=dyp2*1e6
        ddxy=dxy*1e2
        ddxpy=dxpy*1e4
```



```
        ddxyp=dxyp*1e4
        ddxpyp=dxpyp*1e6
        eemitnx=emitnx*1e4
        eemitny=emitny*1e4
        aangmom=angmom*1e4
        eemitng=emitng*1e4
        eemitnh=emitnh*1e4
      RETURN
      END
c
c
c=================================================================
c *** Integration routine ***************************************
c=================================================================
c *** This subroutine is adapted from the subroutine RK4 from  *******
c *** Numerical Recipes by Press et al. The main difference is the ***
c *** use of two dimensional arrays rather than linear arrays.  ******
c *** Input variables:                                          ******
c ***  Y(ieqn,it):   The 2-D array of dependent variables       ******
c ***    DYDX(ieqn,it): Initial derivatives calculated in calling *****
c ***    subroutine.                                             *****
c ***    ieqnmax,nprtmax: Maximum dimensions of the arrays       *****
c ***    X: The independent variable                             *****
c ***    H: The step size in X.                                  *****
c ***    DERIVS: The name of the subroutine with the derivatives. *****
c *** Output variables:                                          *****
c ***    YOUT(ieqn,it): The new values of Y at X+H               *****
c=================================================================
      SUBROUTINE fint(Y,YOUT,X,H,DERIVS,DYDX)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(ieqnlim=10,nprtmax=1)
      EXTERNAL DERIVS
      DIMENSION Y(ieqnlim,nprtmax),DYDX(ieqnlim,nprtmax),
     .     YOUT(ieqnlim,nprtmax)
      DIMENSION YT(ieqnlim,nprtmax),DYT(ieqnlim,nprtmax),
     .   DYM(ieqnlim,nprtmax)
      ieqnmax=10
      itmax=nprtmax
      HH=H*0.5
      H6=H/6.
      XH=X+HH
      XF=X+H
      do 110 it=1,itmax
         DO 11 ieqn=1,ieqnmax
            YT(ieqn,it)=Y(ieqn,it)+HH*DYDX(ieqn,it)
 11      CONTINUE
 110  CONTINUE
      CALL DERIVS(XH,YT,DYT)
      do 120 it=1,itmax
         DO 12 ieqn=1,ieqnmax
            YT(ieqn,it)=Y(ieqn,it)+HH*DYT(ieqn,it)
 12      CONTINUE
 120  CONTINUE
      CALL DERIVS(XH,YT,DYM)
      do 130 it=1,itmax
         DO 13 ieqn=1,ieqnmax
         .  YT(ieqn,it)=Y(ieqn,it)+H*DYM(ieqn,it)
            DYM(ieqn,it)=DYT(ieqn,it)+DYM(ieqn,it)
 13      CONTINUE
 130  CONTINUE
      CALL DERIVS(XF,YT,DYT)
      do 140 it=1,itmax
         DO 14 ieqn=1,ieqnmax
            YOUT(ieqn,it)=Y(ieqn,it)
     .          +H6*(DYDX(ieqn,it)+DYT(ieqn,it)+2.*DYM(ieqn,it))
 14      CONTINUE
 140  CONTINUE
      RETURN
      END


c=================================================================
c *** Runge-kutta integration routine ***************************
```

```
c*****************************************************************
      SUBROUTINE dfdzsub(zw,fw,dfdzw)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(nprtmax=1,nthet=1000)
      DIMENSION fw(10,nprtmax),dfdzw(10,nprtmax)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2(566),fkbtx02,fkbty02,pervo,betao,
     .            pervpower
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab(566)
          dx2=fw(1,1)
          dxxp=fw(2,1)
          dxp2=fw(3,1)
          dy2=fw(4,1)
          dyyp=fw(5,1)
          dyp2=fw(6,1)
          dxy=fw(7,1)
          dxpy=fw(8,1)
          dxyp=fw(9,1)
          dxpyp=fw(10,1)
          call betavala(zw)
c          perv=pervo/(betaovbetao**2)  (constant bunchlength)
          perv=pervo*betaovbetao**pervpower
          alpha=.5*DATAN(2*dxy/(dx2-dy2))
          cosalph=DCOS(alpha)
          sinalph=DSIN(alpha)
          cos2alpha=cosalph**2
          sin2alpha=sinalph**2
          sinalphcosalph=sinalph*cosalph
          dxb2=dx2*cos2alpha+dy2*sin2alpha+2*dxy*sinalphcosalph
          dyb2=dy2*cos2alpha+dx2*sin2alpha-2*dxy*sinalphcosalph
          dxbdyb=SQRT(dxb2*dyb2)
          fksxb=.5*perv/(dxb2+dxbdyb)
          fksyb=.5*perv/(dyb2+dxbdyb)
          fksxx=fksxb*cos2alpha+fksyb*sin2alpha
          fksxy=(fksxb-fksyb)*sinalphcosalph
          fksyx=fksxy
          fksyy=fksyb*cos2alpha+fksxb*sin2alpha
          fkqxx=fkbeta02x(zw)*costwotheta/betaovbetao
          fkqyy=fkbeta02y(zw)*costwotheta/betaovbetao
          fkqxy=fkbeta02x(zw)*sintwotheta/betaovbetao
          fkqyx=fkqxy
          fkxx=fkqxx+fksxx
          fkxy=fkqxy+fksxy
          fkyx=fkqyx+fksyx
          fkyy=fkqyy+fksyy
c dx2'
            dfdzw(1,1)= 2*fw(2,1)
c dxxp'=
            dfdzw(2,1)=fw(3,1) + fkxx*fw(1,1) +
                                 fkxy*fw(7,1) - dbdzovb*fw(2,1)
c dxp2'=
            dfdzw(3,1)= 2.0d0*fkxx*fw(2,1)+
                      2.0d0*fkxy*fw(8,1) - 2.0d0*dbdzovb*fw(3,1)
c dy2'=
            dfdzw(4,1)= 2*fw(5,1)
c dyyp'=
            dfdzw(5,1)= fw(6,1)+ fkyy*fw(4,1) +
                                 fkyx*fw(7,1)- dbdzovb*fw(5,1)
c dyp2'=
            dfdzw(6,1)= 2.0d0*fkyy*fw(5,1)+
                      2.0d0*fkxy*fw(9,1) - 2.0d0*dbdzovb*fw(6,1)
c dxy'
            dfdzw(7,1)=  fw(8,1)+fw(9,1)
c dxpy'=
            dfdzw(8,1)= fw(10,1)+fkxx*fw(7,1)+
                                 fkxy*fw(4,1)- dbdzovb*fw(8,1)
c dxyp'=
            dfdzw(9,1)=  fw(10,1)+fkyy*fw(7,1)+
```



```
                                 fkyx*fw(1,1)- dbdzovb*fw(9,1)
c dxpyp'=
            dfdzw(10,1)= fkxx*fw(9,1) + fkxy*fw(5,1) +
                         fkyy*fw(8,1) + fkyx*fw(2,1)
                         - 2.0d0*dbdzovb*fw(10,1)
c
      RETURN
      END
c
c
c
c*****************************************************************
c *** fodo routine ***********************************************
c*****************************************************************
      SUBROUTINE fodostep
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(nthet=1000)
c      DIMENSION dthetatab(nthet)
      common/blk2/fkbet2(566),fkbtx02,fkbty02,pervo,betao,
     .            pervpower
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab(566)
      COMMON/blk31/iprint,nprint,quadfld(566)
      IF (0.LE.jstep.and.jstep.LT.jfend) THEN
         dz=dzmag
          dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         fkbtx02=fkbet2(ilattno)
         fkbty02=-fkbet2(ilattno)
         jstep=jstep+1
         RETURN
      ENDIF
      IF (jstep.EQ.jdbeg)THEN
         ilattno=ilattno+1
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         if(iprint.eq.nprint)
     .      print *,"Quad no.=",ilattno," dtheta=",dtheta,
     .      "B Field=",quadfld(ilattno)
      ENDIF
      IF (jdbeg.LE.jstep.and.jstep.LT.jdend) THEN
         dz=dzmag
         fkbtx02=-fkbet2(ilattno)
         fkbty02=fkbet2(ilattno)
         jstep=jstep+1
         RETURN
      ENDIF
      IF (jstep.GE.jlatt) THEN
         dz=dzmag
         jstep=1
         ilattno=ilattno+1
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         fkbtx02=fkbet2(ilattno)
         fkbty02=-fkbet2(ilattno)
c         if(iprint.eq.nprint)
c     .      print *,"Quad no.=",ilattno," dtheta=",dtheta,
c     .      "B Field=",quadfld(ilattno)
         RETURN
      ENDIF
      dz=dzdrift
      fkbtx02=0.0d0
      fkbty02=0.0d0
      jstep=jstep+1
      RETURN
      END
c
```

```
c
c
      FUNCTION fkbeta02x(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2(566),fkbtx02,fkbty02,pervo,betao,
     .            pervpower
      fkbeta02x=-fkbtx02
      RETURN
      END
c
c
c
      FUNCTION fkbeta02y(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk1/dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp
      common/blk2/fkbet2(566),fkbtx02,fkbty02,pervo,betao,
     .            pervpower
      fkbeta02y=-fkbty02
      RETURN
      END
c
c
c
      SUBROUTINE betaval*(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      common/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
c betaconl=(energyf/energyo-1.0d0)/(nlaps*flength)
c halfbetaconl=.5*betaconl
      betaovbetao=DSQRT(1.0d0 + betaconl*z)
      dbdzovb=halfbetaconl/(1.0d0 + betaconl*z)
      return
      end
c*****************************************************************
      FUNCTION DEEEX2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk40/emitg2, emith2, beta
c
c  The full expression for dx2 in terms of the other 8 moments and the
c   emittances
c
c234567890123456789012345678901234567890123456789012345678901234567890123456789012
c
      DEEEX2=(4.*beta**4*dxpyp**2*dxxp**2 - 8.*beta**4*dxpyp**3*dxy +
     - 8.*beta**4*dxpy*dxpyp**2*dxyp +
     - 8.*beta**4*dxpyp*dxp2*dxxp*dxyp - 4.*beta**4*dxp2**2*dxyp**2 -
     - 8.*beta**4*dxp2*dxxp**2*dyp2 + 8.*beta**4*dxpyp*dxp2*dxy*dyp2 -
     - 8.*beta**4*dxpy*dxp2*dxyp*dyp2 + 4.*beta**4*dxpy**2*dyp2**2 -
     - 8.*beta**4*dxpy*dxpyp*dyp2*dyyp + 4.*beta**4*dxpyp**2*dyyp**2 +
     - 0.5*beta**2*dxpyp**2*emitg2 - 0.5*beta**2*dxp2*dyp2*emitg2 -
     - 0.015625*Sqrt(65536.00000000001*beta**8*dxpyp**4*dxxp**4 -
     - 262144.*beta**8*dxpyp**5*dxxp**2*dxy +
     - 262144.*beta**8*dxpyp**6*dxy**2 +
     - 262144.*beta**8*dxpy*dxpyp**4*dxxp**2*dxyp -
     - 262144.*beta**8*dxpyp**3*dxp2*dxxp**3*dxyp -
     - 524288.0000000001*beta**8*dxpy*dxpyp**5*dxy*dxyp +
     - 524288.0000000001*beta**8*dxpyp**4*dxp2*dxxp*dxy*dxyp +
     - 262144.*beta**8*dxpy**2*dxpyp**4*dxyp**2 -
     - 524288.0000000001*beta**8*dxpy*dxpyp**3*dxp2*dxxp*dxyp**2 +
     - 393216.*beta**8*dxpyp**2*dxp2**2*dxxp**2*dxyp**2 -
     - 262144.*beta**8*dxpyp**3*dxp2**2*dxy*dxyp**2 +
     - 262144.*beta**8*dxpy*dxpyp**2*dxp2**2*dxyp**3 -
     - 262144.*beta**8*dxpyp*dxp2**3*dxxp*dxyp**3 +
     - 65536.00000000001*beta**8*dxp2**4*dxyp**4 +
     - 262144.*beta**8*dxpyp**3*dxp2*dxxp**2*dxy*dyp2 -
     - 786432.0000000001*beta**8*dxpyp**4*dxp2*dxy**2*dyp2 -
     - 262144.*beta**8*dxpy*dxpyp**2*dxp2*dxxp**2*dxyp*dyp2 +
     - 1.572864e6*beta**8*dxpy*dxpyp**3*dxp2*dxy*dxyp*dyp2 -
     - 524288.0000000001*beta**8*dxpyp**2*dxp2**2*dxxp*dxy*dxyp*
     - dyp2 - 786432.0000000001*beta**8*dxpy**2*dxpyp**2*dxp2*
     - dxyp**2*dyp2 +
     - 524288.0000000001*beta**8*dxpy*dxpyp*dxp2**2*dxxp*dxyp**2*
     - dyp2 + 262144.*beta**8*dxpyp*dxp2**3*dxy*dxyp**2*dyp2 -
     - 262144.*beta**8*dxpy**2*dxpyp**2*dxp2**3*dxyp**3*dyp2 +
     - 131072.*beta**8*dxpy**2*dxpyp**2*dxxp**2*dyp2**2 -
     - 262144.*beta**8*dxpy**2*dxpyp**3*dxy*dyp2**2 -
     - 524288.0000000001*beta**8*dxpy*dxpyp**2*dxp2*dxxp*dxy*dyp2*
     - 262144.*beta**8*dxpy**3*dxpyp**2*dxyp*dyp2**2 +
     - 262144.*beta**8*dxpy*dxpyp**2*dxp2*dxxp*dxyp*dyp2**2 -
     - 1.048576e6*beta**8*dxpy*dxpyp*dxp2**2*dxy*dxyp*dyp2**2 +
     - 393216.*beta**8*dxpy**2*dxp2**2*dxyp**2*dyp2**2 -
     - 262144.*beta**8*dxpy**2*dxp2*dxxp**2*dyp2**3 +
     - 262144.*beta**8*dxpy**2*dxpyp*dxp2*dxy*dyp2**3 +
     - 524288.0000000001*beta**8*dxpy*dxp2**2*dxxp*dxy*dyp2**3 -
     - 262144.*beta**8*dxp2**3*dxy**2*dyp2**3 -
     - 262144.*beta**8*dxpy**3*dxp2*dxyp*dyp2**3 +
     - 65536.00000000001*beta**8*dxpy**4*dyp2**4 -
     - 262144.*beta**8*dxpy*dxpyp**3*dxxp**2*dxy*dyyp +
     - 524288.0000000001*beta**8*dxpy*dxpyp**4*dxy*dyp2*dyyp +
     - 524288.0000000001*beta**8*dxpyp**3*dxp2*dxxp*dxy*dyp2*dyyp -
     - 524288.0000000001*beta**8*dxpy**2*dxpyp**3*dxyp*dyp2*dyyp -
     - 524288.0000000001*beta**8*dxpyp**2*dxp2**2*dxy*dxyp*dyp2*
     - dyyp + 262144.*beta**8*dxpy*dxpyp*dxp2**2*dxyp**2*dyp2*
     - dyyp + 524288.0000000001*beta**8*dxpy*dxpyp*dxp2*dxxp**2*
     - dyp2**2*dyyp -
     - 524288.0000000001*beta**8*dxpy*dxpyp**2*dxp2*dxy*dyp2**2*
     - dyyp - 524288.0000000001*beta**8*dxpyp*dxp2**2*dxxp*dxy*
     - dyp2**2*dyyp +
     - 524288.0000000001*beta**8*dxpy**2*dxpyp*dxp2*dxyp*dyp2**2*
     - dyyp - 524288.0000000001*beta**8*dxpy*dxp2**2*dxxp*dxyp*
     - dyp2**2*dyyp +
     - 524288.0000000001*beta**8*dxp2**3*dxy*dxyp*dyp2**2*dyyp -
     - 262144.*beta**8*dxpy**3*dxpyp*dyp2**3*dyyp +
     - 131072.*beta**8*dxpyp**4*dxxp**2*dyyp**2 -
     - 262144.*beta**8*dxpyp**5*dxy*dyyp**2 +
     - 262144.*beta**8*dxpy*dxpyp**4*dxyp*dyyp**2 -
     - 262144.*beta**8*dxpyp**3*dxp2*dxxp*dxyp*dyyp**2 +
     - 131072.*beta**8*dxpyp**2*dxp2**2*dxyp**2*dyyp**2 -
     - 262144.*beta**8*dxpyp**2*dxp2*dxxp**2*dyp2*dyyp**2 +
     - 262144.*beta**8*dxpyp**3*dxp2*dxy*dyp2*dyyp**2 -
     - 262144.*beta**8*dxpy*dxpyp**2*dxp2*dxyp*dyp2*dyyp**2 +
     - 524288.0000000001*beta**8*dxpyp*dxp2**2*dxxp*dxyp*dyp2*
     - dyyp**2 - 262144.*beta**8*dxp2**3*dxyp**2*dyp2*dyyp**2 +
     - 393216.*beta**8*dxpy**2*dxpyp**2*dyp2**2*dyyp**2 -
     - 262144.*beta**8*dxpy*dxpyp**3*dyp2*dyyp**1 +
     - 65536.00000000001*beta**8*dxpyp**4*dyyp**4 +
     - 16384.*beta**6*dxpyp**4*dxxp**2*emitg2 -
     - 32768.*beta**6*dxpyp**5*dxy*emitg2 +
     - 32768.*beta**6*dxpy*dxpyp**4*dxyp*emitg2 -
     - 32768.*beta**6*dxpyp**3*dxp2*dxxp*dxyp*emitg2 +
     - 16384.*beta**6*dxpyp**2*dxp2**2*dxyp**2*emitg2 -
     - 16384.*beta**6*dxpyp**2*dxp2*dxxp**2*dyp2*emitg2 +
     - 65536.00000000001*beta**6*dxpyp**3*dxp2*dxy*dyp2*emitg2 -
     - 65536.00000000001*beta**6*dxpy*dxpyp**2*dxp2*dxyp*dyp2*
     - emitg2 + 32768.*beta**6*dxpyp*dxp2**2*dxxp*dxyp*dyp2*
     - emitg2 - 16384.*beta**6*dxp2**3*dxyp**2*dyp2*emitg2 +
     - 16384.*beta**6*dxpy**2*dxpyp**2*dyp2**2*emitg2 -
     - 32768.*beta**6*dxpyp*dxp2**2*dxy*dyp2**2*emitg2 +
     - 32768.*beta**6*dxpy*dxp2**2*dxyp*dyp2**2*emitg2 -
     - 16384.*beta**6*dxpy**2*dxp2*dyp2**3*emitg2 -
     - 32768.*beta**6*dxpy*dxpyp**3*dyp2*dyyp*emitg2 +
     - 32768.*beta**6*dxpy*dxpyp*dxp2*dyp2**2*dyyp*emitg2 +
     - 16384.*beta**6*dxpyp**4*dyyp**2*emitg2 -
     - 16384.*beta**6*dxpyp**2*dxp2*dyp2*dyyp**2*emitg2 +
     - 1024.*beta**4*dxpyp**4*emitg2**2 -
     - 2048.*beta**4*dxpyp**2*dxp2*dyp2*emitg2**2 +
     - 1024.*beta**4*dxp2**2*dyp2**2*emitg2**2 +
     - 1024.*beta**4*dxpyp**2*dxp2*dyp2*emith2**2 -
     - 1024.*beta**4*dxp2**2*dyp2**2*emith2**2))/
     - (beta**4*dxp2*(8.*dxpyp**2 - 8.*dxp2*dyp2))
c
      RETURN
      END
c
```

```
      FUNCTION DEEEY2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk40/emitg2, emith2, beta
c
c  The full expression for dy2 in terms of the other 8 moments and the
c   emittances
c
c
c23456789012345678901234567890123456789012345678901234567890123456789012
c
        DEEEY2=0.5*(-1.*beta**2*
     -   (256.*beta**2*dxpyp**2*dxxp**2 - 512.*beta**2*dxpyp**3*dxy +
     -    512.*beta**2*dxpy*dxpyp**2*dxyp -
     -          512.*beta**2*dxpyp*dxp2*dxxp*dxyp +
     -          256.*beta**2*dxp2**2*dxyp**2 +
     -          512.*beta**2*dxpyp*dxp2*dxy*dyp2 -
     -          512.*beta**2*dxpy*dxp2*dxyp*dyp2 -
     -          256.*beta**2*dxpy**2*dyp2**2 +
     -          512.*beta**2*dxpy*dxpyp*dyp2*dyyp +
     -          256.*beta**2*dxpyp**2*dyyp**2 -
     -   512.*beta**2*dxp2*dyp2*dyyp**2 + 32.*dxpyp**2*emitg2 -
     -          32.*dxp2*dyp2*emitg2) -
     -        1.*Sqrt(beta**4*(256.*beta**2*dxpyp**2*dxxp**2 -
     -                512.*beta**2*dxpyp**3*dxy +
     -                512.*beta**2*dxpy*dxpyp**2*dxyp -
     -                512.*beta**2*dxpyp*dxp2*dxxp*dxyp +
     -                256.*beta**2*dxp2**2*dxyp**2 +
     -                512.*beta**2*dxpyp*dxp2*dxy*dyp2 -
     -                512.*beta**2*dxpy*dxp2*dxyp*dyp2 -
     -                256.*beta**2*dxpy**2*dyp2**2 +
     -                512.*beta**2*dxpy*dxpyp*dyp2*dyyp +
     -                256.*beta**2*dxpyp**2*dyyp**2 -
     -   512.*beta**2*dxp2*dyp2*dyyp**2 + 32.*dxpyp**2*emitg2 -
     -                32.*dxp2*dyp2*emitg2)**2 -
     -     4.*beta**4*dyp2*(-256.*dxpyp**2 + 256.*dxp2*dyp2)*
     -            (-256.*beta**4*dxpyp**2*dxp2*dxy**2 +
     -                512.*beta**4*dxpy*dxpyp*dxp2*dxy*dxyp -
     -                256.*beta**4*dxpy**2*dxp2*dxyp**2 +
     -                256.*beta**4*dxpy**2*dxxp**2*dyp2 -
     -                512.*beta**4*dxpy**2*dxpyp*dxy*dyp2 -
     -                512.*beta**4*dxpy*dxp2*dxxp*dxy*dyp2 +
     -                256.*beta**4*dxp2**2*dxy**2*dyp2 +
     -                512.*beta**4*dxpy**3*dxyp*dyp2 -
     -                512.*beta**4*dxpy*dxpyp*dxxp**2*dyyp +
     -                1024.*beta**4*dxpy*dxpyp**2*dxy*dyyp +
     -                512.*beta**4*dxpyp*dxp2*dxxp*dxy*dyyp -
     -                1024.*beta**4*dxpy**2*dxpyp*dxyp*dyyp +
     -                512.*beta**4*dxpy*dxp2*dxxp*dxyp*dyyp -
     -                512.*beta**4*dxp2**2*dxy*dxyp*dyyp -
     -                512.*beta**4*dxpyp*dxp2*dxy*dyyp**2 +
     -                512.*beta**4*dxpy*dxp2*dxyp*dyyp**2 +
     -                256.*beta**4*dxpy**2*dyp2*dyyp**2 -
     -                512.*beta**4*dxpy*dxpyp*dyyp**3 +
     -                256.*beta**4*dxp2*dyyp**4 +
     -                32.*beta**2*dxpy**2*dyp2*emitg2 -
     -    64.00000000000001*beta**2*dxpy*dxpyp*dyyp*emitg2 +
     -  32.*beta**2*dxp2*dyyp**2*emitg2 + 1.*dxp2*emith2**2)))/
     -      (beta**4*dyp2*(-256.*dxpyp**2 + 256.*dxp2*dyp2))
c
      RETURN
      END
c*****************************************************************
c******  Generalized Emittance Calculations  ********************
c*****************************************************************
c
        FUNCTION EG(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
        IMPLICIT DOUBLE PRECISION (a-h,o-z)
        COMMON/blk40/emitg2, emith2, beta
c
        EG=16*beta**2*(dxpyp*dxy - dxpy*dxyp) +
     -   8.*beta**2*(-dxxp**2 + dxp2*dx2) + 8.*beta**2*
     -   (-dyyp**2 + dyp2*dy2)
c
```



```
        RETURN
        END
c
c*****************************************************************
        FUNCTION EH(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
        IMPLICIT DOUBLE PRECISION (a-h,o-z)
        COMMON/blk40/emitg2, emith2, beta
c
        EH=(256*beta**4*(dxpyp**2*dxy**2 - 2*dxpy*dxpyp*dxy*dxyp +
     -          dxpy**2*dxyp**2 + 2*dxpy*dxxp*dxy*dyp2 -
     -          dxp2*dxy**2*dyp2 -
     -          dxpy**2*dx2*dyp2 - 2*dxpyp*dxxp*dxy*dyyp -
     -          2*dxpy*dxxp*dxyp*dyyp + 2*dxp2*dxy*dxyp*dyyp +
     -          2*dxpy*dxpyp*dx2*dyyp + 2*dxpyp*dxxp*dxyp*dy2 -
     -          dxp2*dxyp**2*dy2 - dxpyp**2*dx2*dy2) +
     -      256*beta**4*(-dxxp**2 + dxp2*dx2)*(-dyyp**2 +
     -      dyp2*dy2))**0.5
c
        RETURN
        END
c*****************************************************************
c
      FUNCTION DEEX2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk40/emitg2, emith2, beta
c
c  The reduced expression for dx2 in terms of the other 8 moments and the
c   emittances
c
c
c23456789012345678901234567890123456789012345678901234567890123456789012
c
        DEEX2=(4.*beta**4*dxpyp**2*dxxp**2 - 8.*beta**4*dxpyp**3*dxy +
     -   8.*beta**4*dxpy*dxpyp**2*dxyp +
     -   8.*beta**4*dxpyp*dxp2*dxxp*dxyp - 4.*beta**4*dxp2**2*dxyp**2 -
     -   8.*beta**4*dxp2*dxxp**2*dyp2 + 8.*beta**4*dxpyp*dxp2*dxy*dyp2 -
     -   8.*beta**4*dxpy*dxp2*dxyp*dyp2 + 4.*beta**4*dxpy**2*dyp2**2 -
     -   8.*beta**4*dxpy*dxpyp*dyp2*dyyp + 4.*beta**4*dxpyp**2*dyyp**2 +
     -   0.5*beta**2*dxpyp**2*emitg2 - 0.5*beta**2*dxp2*dyp2*emitg2)/
     -   (beta**4*dxp2*(8.*dxpyp**2 - 8.*dxp2*dyp2))
c
      RETURN
      END
c*****************************************************************
      FUNCTION DEEY2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk40/emitg2, emith2, beta
c
c  The reduced expression for dy2 in terms of the other 8 moments and the
c   emittances
c
c
c23456789012345678901234567890123456789012345678901234567890123456789012
c
        DEEY2=0.5*(-1.*beta**2*
     -   (256.*beta**2*dxpyp**2*dxxp**2 - 512.*beta**2*dxpyp**3*dxy +
     -    512.*beta**2*dxpy*dxpyp**2*dxyp -
     -          512.*beta**2*dxpyp*dxp2*dxxp*dxyp +
     -          256.*beta**2*dxp2**2*dxyp**2 +
     -          512.*beta**2*dxpyp*dxp2*dxy*dyp2 -
     -          512.*beta**2*dxpy*dxp2*dxyp*dyp2 -
     -          256.*beta**2*dxpy**2*dyp2**2 +
     -          512.*beta**2*dxpy*dxpyp*dyp2*dyyp +
     -          256.*beta**2*dxpyp**2*dyyp**2 -
     -   512.*beta**2*dxp2*dyp2*dyyp**2 + 32.*dxpyp**2*emitg2 -
     -          32.*dxp2*dyp2*emitg2)/
     -      (beta**4*dyp2*(-256.*dxpyp**2 + 256.*dxp2*dyp2))
c
      RETURN
      END
```

E.3 "CORRECTOR"

"CORRECTOR" is a FORTRAN 77 code modified from J. Barnard's MOMENT code. It implements the SPSA algorithm to optimize the settings of the adjustable magnetic quadrupoles of the beam correction system based on a desired set of beam second moments. The code allows the user to determine the initial settings of the quadrupole field strengths and rotation angles. "CORRECTOR's" algorithm incorporates the reduced set of 8 second moment evolution equations by substituting in the approximate explicit solutions for Δx^2 and Δy^2 (Equations 6.11 and 6.16) wherever they occur in the moment evolution equations.

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c                                                                        c
c  'corrector.f': SPSA algorithm to find the optimum position and        c
c  magnetic fields for a 4 quad system to shape the transverse second c
c  moments to remove any effects of upstream arbitrary quad rotation  c
c  and/or off-design quad field strengths .  This is done with 4       c
c  quads at the usual HLP spacing with variable rotation angle and     c
c  magnetic field strength.  The reduced form of the constraint c
c  equations is used                                                   c
c                                                                        c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      PROGRAM moment
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
c  *** SETUP PROGRAM ***********************************************
      EXTERNAL dfdzsub
c  *** filename characters (Aa,etc.)*******************************
      PARAMETER (nprtmax=1,nthet=100000)
c  *** allocate space for arrays***********************************
      DIMENSION g0(10,nprtmax),g1(10,nprtmax),dfdzw0(8,nprtmax)
      DIMENSION dthetatab(nthet),good(10),y(2),kdelta(10)
      DIMENSION thetanew(10),gee(10),geeb(10),quadfldnew(10)
      COMMON/blk1/emitg2, emith2, beta
      COMMON/blk2/fkbet2(10),fkbtx02,fkbty02,pervo,betao,
     .            pervpower,quadfld(10),fkfodo(10)
      COMMON/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk4/ddx2(800),ddxxp(800),ddxp2(800),ddy2(800),ddyyp(800),
     .ddyp2(800),ddxy(800),ddxpy(800),ddxyp(800),ddxpyp(800),zee(800)
      COMMON/blk5/eemitnx(800),eemitny(800),aangmom(800),
     .eemitng(800),eemitnh(800)
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab
      COMMON/blk31/iprint,nprint
      COMMON/blk35/good
c
            ilattno=1
            ifodo=1
c
            OPEN(70, file='converge2.dat')
            OPEN(80, file='allout')
        print *,'started'
c *** Initialize constants  ***************************************
c***Numerical constants*****************************
            pi=3.141592654d0
            fpi=3.141592654d0
            halfpi=pi/2.d0
            fmhc2=938.d6
            fmec2=511.d3
            frpieps0=1.1127d-10
            curra=31.3d6
c***Physical data*********************************
           etaq=0.29d0
           flattic=38.1d0
           rpipe=3.149d0
           nhlp=4
           nlaps=1
           flength=nlaps*nhlp*flattic
           qcharge=1.0d0
           energyo=80.0d3
           energyf=80.0d3
           amu=39.9d0
           beta=DSQRT(2.0d0*energyo/(amu*fmhc2))
            gamma=DSQRT(1/(1-beta**2))
           betao=beta
           betaf=DSQRT(2.0d0*energyf/(amu*fmhc2))
           betaconl=(energyf/energyo-1.0d0)/flength
           halfbetaconl=.5d0*betaconl
           current=2.0d-3
            brho=25.798d0
            quadb0=0.25258d0
           nstepsperhlp=200
           z0=0
```



```
            pervalt=2.0d0*current*qcharge/(amu*beta**3*curra)
            pervo=.000357d0
            pervf=.000353d0
             (kbeta0*quadb0/(rpipe*brho)
c*****************************************************************
c*****************************************************************
c Obtained from straight6g.f with no rotation errors after hlp #566
c  at z=21537.55   file='aligned566.out'
c*****************************************************************
        good(1)=.5854810257981904
        good(2)=1.0166301804069 77E-02
        good(3)=1.771945125260419E-04
        good(4)=.1596119570577
        good(5)=-3.051328062721792E-03
        good(6)=6.077825066057941E-05
        good(7)= 0.0
        good(8)= 0.0
        good(9)= 0.0
        good(10)= 0.0
c***SPSA Parameters*********************************************
         maxit=400000
         kbiga=maxit/20
         smalla=.06
         smallb=.00333
         smallc=1.0d-6
         smalld=1.0d-6
         k=0
         ysqrmin=10
        angmommin=100
c*************************************************************
c       Get a divide by zero in Pervpower if betao=betaf so for
c       a drifting beam calculate betas based on 80keV and 81keV
c*************************************************************
         pervpower=-1.81444d+00
c***initialization parameters*************************
        ilattno=1
        ifodo=1
c***Initial quad rotation angles and quad strengths***********
            thetamax=1.0d+00*0.0174533d+00
            DO i=1,4
            dthetatab(i)=0.0d+00
            quadfld(i)=quadb0
            ENDDO
c***SPSA Setup*************************************************
        iseed=227
        kflag=0
 50     asubk=smalla/(kbiga+1+k)**.602
        bsubk=smallb/(kbiga+1+k)**.602
        csubk=smallc/(k+1)**.101
        dsubk=smalld/(k+1)**.101
        jstep=1
        DO i=1,8
          rdm=ran(iseed)
          IF (rdm .LT. 0.5) THEN
            kdelta(i)=-1
          ELSE
            kdelta(i)=1
          ENDIF
        ENDDO
        kounter=1
        Do 1000 kounter=1,2
         jstep=1
          IF (kounter .EQ.1) THEN
                DO i=1,4
                  j=i+4
                  dthetatab(i)=dthetatab(i)+csubk*kdelta(i)
                  quadfld(i)=quadfld(i)+dsubk*kdelta(j)
                ENDDO
          ELSE
                DO i=1,4
                  j=i+4
                  dthetatab(i)=dthetatab(i)-2.*csubk*kdelta(i)
                  quadfld(i)=quadfld(i)-2.*dsubk*kdelta(j)
```

```
                ENDDO
            ENDIF
c*****************************************************************
              thetasum=0.0d0
              theta2sum=0.0d0
              nlatt=nhlp*nlaps
              DO j=1,nlatt
                 thetasum=thetasum+ dthetatab(j)
                 theta2sum=theta2sum+ dthetatab(j)**2
              ENDDO
              thetaavg=thetasum/nlatt
              theta2avg=theta2sum/nlatt
              dthetarms=DSQRT(theta2avg-thetaavg**2)
              dthetarmstheory=2.*thetamax/3.
              llattno=1
              costwotheta=1.0d0
              sintwotheta=0.0d0
             dz0=flattic/float(nstepsperhlp)
             epsdyinit=0.0d-5
c*****************************************************
c  random rotations to max of 0.5 deg, for 562 HLP, calculated at
c  z=21412.2 cm
c*********Input to Quad 0 cm ************************
c** the moments after 562 hlp with
c   random rotation error up to 0.5 deg. z=21412.2
             dx2=.8117015616058228
             dxxp=1.442077282077043E-02
             dxp2=2.610347833560520E-04
             dy2=.2100116409152491
             dyyp=-5.210377682372131E-03
             dyp2=1.728112964400602E-04
             dxy=-5.480955980645722E-02
             dxpy=-2.423687546361204E-03
             dxyp=-4.475037712971694E-03
             dxpyp=-3.816892533347671E-05
             dalpha0=0.5*DATAN(2*dxy/(dx2-dy2))
c*****************************************************************
c *** Calculate the generalized emittances based on the above******
c*****************************************************************
c *** generalized emittance constraint equations
c
         emitg2=EG(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
         emith2=EH(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
c
c*****************************************************************
c *** Setup fodo stuff ***************************************
c *** Initial dx^2*k_beta^2(including space charge) ***************
             z=a
             dz=dz0
             IF (jfodo.EQ.1) THEN
c   jfend is the value of j at end of focusing magnet
c   focusing magnet has values of j between 0 and jfend
c   first drift                          jfend and jdbeg
c   defocusing magnet                    jdbeg and jdend
c   second drift                         jdend and jlatt
                jfend=AINT(etaq*flattic/dz0 +.5)
                IF (jfend.LT.1) jfend=1
                dzmag=etaq*flattic/jfend
                jdrift=AINT((1.0d0-etaq)*flattic/dz0+.5)
                IF (jdrift.LT.1)jdrift=1
                dzdrift=(1.0d0-etaq)*flattic/jdrift
                jdbeg=jfend+jdrift
                jdend=jdbeg+jfend
                jlatt=jdend+jdrift
                  nz=4*nstepsperhlp
c                nz=(3.+etaq)*nstepsperhlp+1

            DO j=1,4
             fkbet2(j)=quadfld(j)/(rpipe*brho)
             fkfodo(j)=fkbet2(j)**(.5)
            ENDDO
          ENDIF
c
```



```
c ***  initialize and write down beginning values ****************
                z=z0
                g0(1,1)=dx2
                  g0(2,1)=dxxp
                g0(3,1)=dxp2
                  g0(4,1)=dy2
                g0(5,1)=dyyp
                g0(6,1)=dyp2
                g0(7,1)=dxy
                g0(8,1)=dxpy
                g0(9,1)=dxyp
                g0(10,1)=dxpyp
c ****************************************************************
c ********************** MAIN Z-LOOP ******************************
c ****************************************************************
                     DO 2000 iz=1,nz
                        IF (jfodo.EQ.1)CALL fodostep
c *** Advance all particles ***************************************

                        CALL dfdzsub(z,g0,dfdzw0)
c
                        CALL fint(g0,g1,z,dz,dfdzsub,dfdzw0)
c
                           z=z+dz
                           dxxp=g1(2,1)
                           dxp2=g1(3,1)
                           dyyp=g1(5,1)
                           dyp2=g1(6,1)
                           dxy=g1(7,1)
                           dxpy=g1(8,1)
                           dxyp=g1(9,1)
                           dxpyp=g1(10,1)
c
                  dx2=DEEX2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
                  dy2=DEEY2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
c
                  g1(1,1)=dx2
                  g1(4,1)=dy2
c
                        DO 400 icoord=1,10
                           g0(icoord,1)=g1(icoord,1)
 400                    CONTINUE
c*The following puts everything in terms of mm and mr vice cm and rads
         zee(iz)=z
         ddx2(iz)=dx2*1e2
         ddxxp(iz)=dxxp*1e4
         ddxp2(iz)=dxp2*1e6
         ddy2(iz)=dy2*1e2
         ddyyp(iz)=dyyp*1e4
         ddyp2(iz)=dyp2*1e6
         ddxy(iz)=dxy*1e2
         ddxpy(iz)=dxpy*1e4
         ddxyp(iz)=dxyp*1e4
         ddxpyp(iz)=dxpyp*1e6
c
 2000  CONTINUE
c
         CALL lossfunc(g1,ysqr,yloss)
         y(kounter)=ysqr
        angmom=(dxyp-dxpy)*1e4
 1000  CONTINUE
c
c*****************************************************************
c        Compute the gradient approximation, calculate the new Quad
c        angles and check that they are significantly different from the
c        last iteration and that the maximum iterations allowed has not
c        been exceeded.  If not, iterate again starting at flag 50.
c*****************************************************************
c
         ydif=y(1)-ylast
         ysqrnum=y(1)-y(2)
         ylast=y(1)
         DO I=1,4
```

```
      j=i+4
        gee(i)=ysqrnum/(2*csubk*kdelta(i))
        geeb(i)=ysqrnum/(2*dsubk*kdelta(j))
        thetanew(i)=dthetatab(i)-(asubk*gee(i))
        quadfldnew(i)=quadfld(i)-(bsubk*geeb(i))
      ENDDO
c
      IF (ysqr .LT. ysqrmin) THEN
         OPEN(101,file='best1')
         OPEN(102,file='zcorr1.dat')
         ysqrmin=ysqr
        WRITE(101,*) 'Least Loss Function was ', ysqr
        WRITE(101,*) 'Angmom was ',angmom
        WRITE (101,*) 'Iteration Number was ',k1,'  k=',k
       WRITE(101,*) 'smalla=',smalla,'kbiga=',kbiga
       WRITE(101,*) 'smallb=',smallb,'  maxit=',maxit
       WRITE(101,*) 'smallc=',smallc,'smalld=',smalld
        DO i=1,4
        WRITE (101,*) 'Quad Field ',i,' was  ',quadfld(i)
        ENDDO
       WRITE (101,*) 'Quad 1 Angle was  ',dthetatab(1)
       WRITE (101,*) 'Quad 2 Angle was  ',dthetatab(2)
       WRITE (101,*) 'Quad 3 Angle was  ',dthetatab(3)
       WRITE (101,*) 'Quad 4 Angle was  ',dthetatab(4)
        WRITE (101,*)  'dx2= ',dx2,good(1)
        WRITE (101,*)  'dxxp= ',dxxp,good(2)
        WRITE (101,*)  'dxp2= ',dxp2,good(3)
        WRITE (101,*)  'dy2= ',dy2,good(4)
        WRITE (101,*)  'dyyp= ',dyyp,good(5)
        WRITE (101,*)  'dyp2= ',dyp2,good(6)
        WRITE (101,*)  'dxy= ',dxy,good(7)
        WRITE (101,*)  'dxpy= ',dxpy,good(8)
        WRITE (101,*)  'dxyp= ',dxyp,good(9)
        WRITE (101,*)  'dxpyp= ',dxpyp,good(10)
        WRITE (101,*)  '        '
        WRITE (101,*)  'beta=',beta
c
      DO i=1,800
      aangmom(i)=ddxyp(i)-ddxpy(i)
      eemitnx2=16.0d0*beta**2*(ddx2(i)*ddxp2(i)-ddxxp(i)**2)
      eemitnx(i)=SQRT(eemitnx2)
      eemitny2=16.0d0*beta**2*(ddy2(i)*ddyp2(i)-ddyyp(i)**2)
      eemitny(i)=SQRT(eemitny2)
      eemitng2=.5*(eemitnx2+eemitny2)+
     .        16.*beta**2*(ddxy(i)*ddxpyp(i)-ddxyp(i)*ddxpy(i))
      eemitng(i)=SQRT(eemitng2)
c 345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
      eemitnh(i)=(eemitnx2*eemitny2+
     .  (4.0d0*beta)**4*(ddxpyp(i)**2*ddxy(i)**2-
     .  2*ddxpy(i)*ddxpyp(i)*ddxy(i)*ddxyp(i)+
     .  ddxpy(i)**2*ddxyp(i)**2+2*ddxpy(i)*ddxxp(i)*ddxy(i)*ddyp2(i)-
     .  ddxp2(i)*ddxy(i)**2*ddyp2(i)-ddxpy(i)**2*ddx2(i)*ddyp2(i)-
     .  2*ddxpyp(i)*ddxxp(i)*ddxy(i)*ddyyp(i)-
     .  2*ddxpy(i)*ddxxp(i)*ddxyp(i)*ddyyp(i)+
     .  2*ddxp2(i)*ddxy(i)*ddxyp(i)*ddyyp(i)+
     .  2*ddxpy(i)*ddxpyp(i)*ddx2(i)*ddyyp(i)+
     .  2*ddxpyp(i)*ddxxp(i)*ddxyp(i)*ddy2(i)-
     .  ddxp2(i)*ddxyp(i)**2*ddy2(i)-
     .  ddxpyp(i)**2*ddx2(i)*ddy2(i)))**0.25
       WRITE(102,*) zee(i),ddx2(i),ddxxp(i),ddxp2(i),ddy2(i),ddyyp(i),
     .   ddyp2(i),ddxy(i),ddxpy(i),ddxyp(i),ddxpyp(i),aangmom(i),
     .   eemitnx(i),eemitny(i),eemitng(i),eemitnh(i),k
      ENDDO
      CLOSE(101)
      CLOSE(102)

      ENDIF
      IF (abs(angmom) .LT. abs(angmommin)) THEN
         OPEN(103,file='best2')
         OPEN(104,file='zcorr2.dat')
        angmommin=angmom
        WRITE(103,*) 'Least Angmom was ',angmom
        WRITE(103,*) 'Loss Function was ', ysqr
```



```
        WRITE (103,*) 'Iteration Number was ',k1,'  k=',k

       WRITE(103,*) 'smalla=',smalla,'kbiga=',kbiga
       WRITE(101,*) 'smallb=',smallb,'  maxit=',maxit
        DO i=1,4
        WRITE(103,*) 'Quad Field ',i,' was  ',quadfld(i)
        ENDDO
       WRITE(103,*) 'Quad 1 Angle was  ',dthetatab(1)
       WRITE(103,*) 'Quad 2 Angle was  ',dthetatab(2)
       WRITE(103,*) 'Quad 3 Angle was  ',dthetatab(3)
       WRITE(103,*) 'Quad 4 Angle was  ',dthetatab(4)
        WRITE(103,*) 'dx2= ',dx2,good(1)
        WRITE(103,*) 'dxxp= ',dxxp,good(2)
        WRITE(103,*) 'dxp2= ',dxp2,good(3)
        WRITE(103,*) 'dy2= ',dy2,good(4)
        WRITE(103,*) 'dyyp= ',dyyp,good(5)
        WRITE(103,*) 'dyp2= ',dyp2,good(6)
        WRITE(103,*) 'dxy= ',dxy,good(7)
        WRITE(103,*) 'dxpy= ',dxpy,good(8)
        WRITE(103,*) 'dxyp= ',dxyp,good(9)
        WRITE(103,*) 'dxpyp= ',dxpyp,good(10)
        WRITE(103,*) '        '
        WRITE(103,*) 'beta=',beta
c
      DO i=1,800
      aangmom(i)=ddxyp(i)-ddxpy(i)
      eemitnx2=16.0d0*beta**2*(ddx2(i)*ddxp2(i)-ddxxp(i)**2)
      eemitnx(i)=SQRT(eemitnx2)
      eemitny2=16.0d0*beta**2*(ddy2(i)*ddyp2(i)-ddyyp(i)**2)
      eemitny(i)=SQRT(eemitny2)
      eemitng2=.5*(eemitnx2+eemitny2)+
     .        16.*beta**2*(ddxy(i)*ddxpyp(i)-ddxyp(i)*ddxpy(i))
      eemitng(i)=SQRT(eemitng2)
c 345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012
      eemitnh(i)=(eemitnx2*eemitny2+
     .  (4.0d0*beta)**4*(ddxpyp(i)**2*ddxy(i)**2-
     .  2*ddxpy(i)*ddxpyp(i)*ddxy(i)*ddxyp(i)+
     .  ddxpy(i)**2*ddxyp(i)**2+2*ddxpy(i)*ddxxp(i)*ddxy(i)*ddyp2(i)-
     .  ddxp2(i)*ddxy(i)**2*ddyp2(i)-ddxpy(i)**2*ddx2(i)*ddyp2(i)-
     .  2*ddxpyp(i)*ddxxp(i)*ddxy(i)*ddyyp(i)-
     .  2*ddxpy(i)*ddxxp(i)*ddxyp(i)*ddyyp(i)+
     .  2*ddxp2(i)*ddxy(i)*ddxyp(i)*ddyyp(i)+
     .  2*ddxpy(i)*ddxpyp(i)*ddx2(i)*ddyyp(i)+
     .  2*ddxpyp(i)*ddxxp(i)*ddxyp(i)*ddy2(i)-
     .  ddxp2(i)*ddxyp(i)**2*ddy2(i)-
     .  ddxpyp(i)**2*ddx2(i)*ddy2(i)))**0.25
       WRITE(104,*) zee(i),ddx2(i),ddxxp(i),ddxp2(i),ddy2(i),ddyyp(i),
     .   ddyp2(i),ddxy(i),ddxpy(i),ddxyp(i),ddxpyp(i),aangmom(i),
     .   eemitnx(i),eemitny(i),eemitng(i),eemitnh(i),k
      ENDDO
      CLOSE(103)
      CLOSE(104)

      ENDIF
      one=dthetatab(1)*1000
      two=dthetatab(2)*1000
      three=dthetatab(3)*1000
      four=dthetatab(4)*1000
     angmom=(dxyp-dxpy)*1e4
c
      WRITE(70,*),k,one,two,three,four,quadfldnew(1),quadfldnew(2),
     .   quadfldnew(3),quadfldnew(4),ysqr,angmom
  111 FORMAT(2(1x,1pe11.4))
  112 FORMAT(5(1x,1pe11.4))
c
c*****************************************************************
c*****  Stopping Criteria  ***************************************
c*****************************************************************
      IF (k1 .GT. maxit) THEN
         PRINT *, 'SPSA exceeded maximum iterations limit'
      WRITE(80,*) 'SPSA exceeded maximum iterations limit of', k1
      ELSE
         sllmit=0.0000001
```

```
          IF (ysqr .GT. slimit) THEN
               DO i=1,4
                 dthetatab(i)=thetanew(i)
                  quadfld(i)=quadfldnew(i)
               ENDDO
               g=asubk*gea(1)
              PRINT *,k,'  ',ysqr,angmom
              k=k+1
               kl=kl+1
              kflag=0
              GO TO 50
          ELSE
          kflag=kflag+1
               IF (kflag .LT. 1) THEN
                 DO i=1,4
                 dthetatab(i)=thetanew(i)
                  quadfld(i)=quadfldnew(i)
                 ENDDO
                 k=k+1
                  kl=kl+1
                 PRINT *,'k=',k,'  angles=',dthetatab(1),dthetatab(2),
     .                    dthetatab(3),dthetatab(4),'  flag=',kflag
                 GO TO 50
               ENDIF
           ENDIF
c        ENDIF
        ENDIF

c*** Printing loop ***********************************************
c
        CALL prntsub(z)
        PRINT *, 'dx2=',dx2,'  dxxp=',dxxp,'  dxp2=',dxp2
        PRINT *, 'dy2=',dy2,'  dyyp=',dyyp,'  dyp2=',dyp2
        PRINT *,'dxy=',dxy,' dxpy=',dxpy,' dxyp=',dxyp,' dxpyp=',dxpyp
c
        DO i=1,4
           angle=thetanew(i)*(360/(2*pi))
          PRINT *, 'Correct angle for Quad',i, 'is',angle
          WRITE (80,*) 'Correct angle for Quad',i, 'is',angle
        ENDDO
        DO i=1,4
        PRINT *, 'Quad Field',i,'is',quadfld(i),'Tesla'
        WRITE (80,*) 'Quad Field',i,'is',quadfld(i),'Tesla'
        ENDDO
        WRITE (80,*)  'dx2=',dx2,'  dxxp=',dxxp,'  dxp2=',dxp2
        WRITE (80,*)  'dy2=',dy2,'  dyyp=',dyyp,'  dyp2=',dyp2
        WRITE (80,*)  'dxy=',dxy,' dxpy=',dxpy,' dxyp=',dxyp,
     .                'dxpyp=',dxpyp
        WRITE(80,*) 'Final angmom=',angmom
        WRITE(80,*) 'Final Loss Function=', ysqr
        WRITE(80,*) 'Desired moments were:'
        WRITE(80,*)      'dx2=0.25491'
        WRITE(80,*)    'dxxp=5.79943d-03'
        WRITE(80,*)    'dxp2=1.33493d-04'
        WRITE(80,*)    'dy2=.67851'
        WRITE(80,*)    'dyyp=-1.14665d-02'
        WRITE(80,*)    'dyp2=1.94344d-04'
        WRITE(80,*)    'dxy= 0, dxpy=0, dxyp=0, dxpyp=0'
        WRITE(80,*)    'smalla=',smalla, '     smallb=',smallb
        WRITE(80,*)    'smallc=',smallc, '     smalld=',smalld
c
c
        CLOSE(70)
      CALL EXIT(1)
      END
c*****************************************************************
c *** Least Square Calculation Routine ***************************
c*****************************************************************
        SUBROUTINE lossfunc(g2,ysqrt,yloss)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk35/good
      DIMENSION good(10),g2(10,1)
c
```



```
                          dx2=g2(1,1)
                        dxxp=g2(2,1)
                        dxp2=g2(3,1)
                          dy2=g2(4,1)
                        dyyp=g2(5,1)
                        dyp2=g2(6,1)
                        dxy=g2(7,1)
                        dxpy=g2(8,1)
                        dxyp=g2(9,1)
                        dxpyp=g2(10,1)
c
        ysqrt=DSQRT((dx2-good(1))**2+(dxxp-good(2))**2+
     .    (dxp2-good(3))**2+(dy2-good(4))**2+(dyyp-good(5))**2+
     .    (dyp2-good(6))**2+dxy**2+dxpy**2+dxyp**2+dxpyp**2)
c
        yloss=DSQRT((1e2*(dx2-good(1)))**2+(1e4*(dxxp-good(2)))**2+
     .    (1e6*(dxp2-good(3)))**2+(1e2*(dy2-good(4)))**2+
     .    (1e4*(dyyp-good(5)))**2+(1e6*(dyp2-good(6)))**2+
     .    (1e4*dxy)**2+(1e4*dxpy)**2+(1e4*dxyp)**2+(1e6*dxpyp)**2)
c
       RETURN
        END
c*****************************************************************
c*****************************************************************
c *** Printing routine *******************************************
c*****************************************************************
      SUBROUTINE prntsub(z)
      PARAMETER (nprtmax=1, nthet=100000)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      DIMENSION dthetatab(nthet)
      COMMON/blk1/emitg2, emith2, beta
      COMMON/blk2/fkbet2(10),fkbtx02,fkbty02,pervo,betao,
     .            pervpower,quadfld(10),fkfodo(10)
      COMMON/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .flattno,dthetatab
      xav=0.
      yav=0.
      xpav=0.
      ypav=0.
      call betavals(z)
      beta=betao*betaovbetao
      dalpha=.5*DATAN(2*dxy/(dx2-dy2))
      cosalpha=DCOS(dalpha)
      sinalpha=DSIN(dalpha)
      cosalpha2=cosalpha**2
      sinalpha2=sinalpha**2
      costwoalpha=DCOS(2.*dalpha)
      sintwoalpha=DSIN(2.*dalpha)
      transenergy=.5*(dxp2 + dyp2
     .     - fkbeta02x(z)*( (dx2-dy2)*costwotheta +
     .                      2.0d0*dxy*sintwotheta )
     .     - pervo*DLOG(
     .                  DSQRT(dx2*cosalpha2 +
     .                        dy2*sinalpha2 +
     .                        dxy*sintwoalpha ) +
     .                  DSQRT(dy2*cosalpha2 +
     .                        dx2*sinalpha2 -
     .                        dxy*sintwoalpha )  ) )
      dxb2=dx2*cosalpha2+dy2*sinalpha2+2*dxy*sinalpha*cosalpha
      dyb2=dy2*cosalpha2+dx2*sinalpha2-2*dxy*sinalpha*cosalpha
      abeam=2*DSQRT(dxb2)
      bbeam=2*DSQRT(dyb2)
      angmom=dxyp-dxpy
      emitnx2=16.0d0*beta**2*(dx2*dxp2-dxxp**2)
      emitnx=SQRT(emitnx2)
      emitny2=16.0d0*beta**2*(dy2*dyp2-dyyp**2)
      emitny=SQRT(emitny2)
      emitng2=.5*(emitnx2+emitny2)+
     .         16.*beta**2*(dxy*dxpyp-dxyp*dxpy)
      emitng=SQRT(emitng2)
```

```
      fnp=1.0d0
      dum=1.0d-8
      RETURN
      END
c
c*****************************************************************
c *** Integration routine ***************************************
c*****************************************************************
c *** This subroutine is adapted from the subroutine RK4 from  *******
c *** Numerical Recipes by Press et al. The main difference is the ***
c *** use of two dimensional arrays rather than linear arrays.  ******
c *** Input variables:                                          ******
c ***  Y(ieqn,it):   The 2-D array of dependent variables       ******
c ***   DYDX(ieqn,it): Initial derivatives calculated in calling *****
c ***   subroutine.                                              *****
c ***   ieqnmax,nprtmax: Maximum dimensions of the arrays        *****
c ***   X: The independent variable                              *****
c ***   H: The step size in X.                                   *****
c ***   DERIVS: The name of the subroutine with the derivatives. *****
c *** Output variables:                                          *****
c ***   YOUT(ieqn,it): The new values of Y at X+H                *****
c*****************************************************************
      SUBROUTINE fint(Y,YOUT,X,H,DERIVS,DYDX)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(ieqnlim=10,nprtmax=1,idir=8)
      EXTERNAL DERIVS
      DIMENSION Y(ieqnlim,nprtmax),DYDX(idir,nprtmax),
     .     YOUT(ieqnlim,nprtmax)
      DIMENSION YT(ieqnlim,nprtmax),DYT(idir,nprtmax),
     .   DYM(idir,nprtmax)
      ieqnmax=10
      itmax=nprtmax
      HH=H*0.5
      H6=H/6.
      XH=X+HH
      XF=X+H
c
      DO 110 it=1,itmax
         YT(2,it)=Y(2,it)+HH*DYDX(1,it)
         YT(3,it)=Y(3,it)+HH*DYDX(2,it)
         YT(5,it)=Y(5,it)+HH*DYDX(3,it)
         YT(6,it)=Y(6,it)+HH*DYDX(4,it)
         YT(7,it)=Y(7,it)+HH*DYDX(5,it)
         YT(8,it)=Y(8,it)+HH*DYDX(6,it)
         YT(9,it)=Y(9,it)+HH*DYDX(7,it)
         YT(10,it)=Y(10,it)+HH*DYDX(8,it)
 110  CONTINUE
c
      CALL DERIVS(XH,YT,DYT)
      DO 120 it=1,itmax
         YT(2,it)=Y(2,it)+HH*DYT(1,it)
         YT(3,it)=Y(3,it)+HH*DYT(2,it)
         YT(5,it)=Y(5,it)+HH*DYT(3,it)
         YT(6,it)=Y(6,it)+HH*DYT(4,it)
         YT(7,it)=Y(7,it)+HH*DYT(5,it)
         YT(8,it)=Y(8,it)+HH*DYT(6,it)
         YT(9,it)=Y(9,it)+HH*DYT(7,it)
         YT(10,it)=Y(10,it)+HH*DYT(8,it)
 120  CONTINUE
c
      CALL DERIVS(XH,YT,DYM)
      DO 130 it=1,itmax
            YT(2,it)=Y(2,it)+H*DYM(1,it)
            DYM(1,it)=DYT(1,it)+DYM(1,it)
            YT(3,it)=Y(3,it)+H*DYM(2,it)
            DYM(2,it)=DYT(2,it)+DYM(2,it)
            YT(5,it)=Y(5,it)+H*DYM(3,it)
            DYM(3,it)=DYT(3,it)+DYM(3,it)
            YT(6,it)=Y(6,it)+H*DYM(4,it)
            DYM(4,it)=DYT(4,it)+DYM(4,it)
            YT(7,it)=Y(7,it)+H*DYM(5,it)
            DYM(5,it)=DYT(5,it)+DYM(5,it)
            YT(8,it)=Y(8,it)+H*DYM(6,it)
```



```
            DYM(6,it)=DYT(6,it)+DYM(6,it)
            YT(9,it)=Y(9,it)+H*DYM(7,it)
            DYM(7,it)=DYT(7,it)+DYM(7,it)
            YT(10,it)=Y(10,it)+H*DYM(8,it)
            DYM(8,it)=DYT(8,it)+DYM(8,it)
 130  CONTINUE
c
      CALL DERIVS(XF,YT,DYT)
      DO 140 it=1,itmax
            YOUT(2,it)=Y(2,it)
     .           +H6*(DYDX(1,it)+DYT(1,it)+2.*DYM(1,it))

            YOUT(3,it)=Y(3,it)
     .           +H6*(DYDX(2,it)+DYT(2,it)+2.*DYM(2,it))
            YOUT(5,it)=Y(5,it)
     .           +H6*(DYDX(3,it)+DYT(3,it)+2.*DYM(3,it))

            YOUT(6,it)=Y(6,it)
     .           +H6*(DYDX(4,it)+DYT(4,it)+2.*DYM(4,it))
            YOUT(7,it)=Y(7,it)
     .           +H6*(DYDX(5,it)+DYT(5,it)+2.*DYM(5,it))

            YOUT(8,it)=Y(8,it)
     .           +H6*(DYDX(6,it)+DYT(6,it)+2.*DYM(6,it))
            YOUT(9,it)=Y(9,it)
     .           +H6*(DYDX(7,it)+DYT(7,it)+2.*DYM(7,it))

            YOUT(10,it)=Y(10,it)
     .           +H6*(DYDX(8,it)+DYT(8,it)+2.*DYM(8,it))
 140  CONTINUE
      RETURN
      END
c
c*****************************************************************
c *** Runge-kutta integration routine ***************************
c*****************************************************************

      SUBROUTINE dfdzsub(zw,fw,dfdzw)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(nprtmax=1,nthet=100000)
      DIMENSION fw(10,nprtmax),dfdzw(8,nprtmax)
      DIMENSION dthetatab(nthet)
      COMMON/blk1/emitg2, emith2, beta
      COMMON/blk2/fkbet2(10),fkbtx02,fkbty02,pervo,betao,
     .            pervpower,quadfld(10),fkfodo(10)
      COMMON/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab
c
            dxxp=fw(2,1)
            dxp2=fw(3,1)
             dyyp=fw(5,1)
            dyp2=fw(6,1)
            dxy=fw(7,1)
            dxpy=fw(8,1)
            dxyp=fw(9,1)
            dxpyp=fw(10,1)
c
            dx2=DEEX2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
            dy2=DEEY2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
c
            fw(1,1)=dx2
            fw(4,1)=dy2
c
            CALL betavals(zw)
            perv=parvo*betaovbetao**pervpower
            alpha=.5*DATAN(2*dxy/(dx2-dy2))
            cosalph=DCOS(alpha)
            sinalph=DSIN(alpha)
            cos2alpha=cosalph**2
            sin2alpha=sinalph**2
```

```
            sinalphcosalph=sinalph*cosalph
            dxb2=dx2*cos2alpha+dy2*sin2alpha+2*dxy*sinalphcosalph
            dyb2=dy2*cos2alpha+dx2*sin2alpha-2*dxy*sinalphcosalph
            dxbdyb=SQRT(dxb2*dyb2)
            fksxb=.5*perv/(dxb2+dxbdyb)
            fksyb=.5*perv/(dyb2+dxbdyb)
            fksxx=fksxb*cos2alpha+fksyb*sin2alpha
            fksxy=(fksxb-fksyb)*sinalphcosalph
            fksyx=fksxy
            fksyy=fksyb*cos2alpha+fksxb*sin2alpha
            fkqxx=fkbeta02x(zw)*costwotheta/betaovbetao
            fkqyy=fkbeta02y(zw)*costwotheta/betaovbetao
            fkqxy=fkbeta02x(zw)*sintwotheta/betaovbetao
            fkqyx=fkqxy
            fkxx=fkqxx+fksxx
            fkxy=fkqxy+fksxy
            fkyx=fkqyx+fksyx
            fkyy=fkqyy+fksyy
c
c
c dxxp'=
                  dfdzw(1,1)=fw(3,1) + fkxx*dx2 +
                                       fkxy*fw(7,1) - dbdzovb*fw(2,1)
c dxp2'=
                  dfdzw(2,1)= 2.0d0*fkxx*fw(2,1)+
                            2.0d0*fkxy*fw(8,1) - 2.0d0*dbdzovb*fw(3,1)
c dyyp'=
                  dfdzw(3,1)= fw(6,1)+ fkyy*dy2 +
                                       fkyx*fw(7,1)- dbdzovb*fw(5,1)
c dyp2'=
                  dfdzw(4,1)= 2.0d0*fkyy*fw(5,1)+
                            2.0d0*fkxy*fw(9,1) - 2.0d0*dbdzovb*fw(6,1)
c dxy'
                  dfdzw(5,1)=  fw(8,1)+fw(9,1)
c dxpy'=
                  dfdzw(6,1)= fw(10,1)+fkxx*fw(7,1)+
                                       fkxy*dy2- dbdzovb*fw(8,1)
c dxyp'=
                  dfdzw(7,1)=  fw(10,1)+fkyy*fw(7,1)+
                                       fkyx*dx2- dbdzovb*fw(9,1)
c dxpyp'=
                  dfdzw(8,1)= fkxx*fw(9,1) + fkxy*fw(5,1) +
                              fkyy*fw(8,1) + fkyx*fw(2,1)
                              - 2.0d0*dbdzovb*fw(10,1)
c
      RETURN
      END
c
c
c
c*****************************************************************
c *** fodo routine ***********************************************
c*****************************************************************
      SUBROUTINE fodostep
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      PARAMETER(nthet=100000)
      DIMENSION dthetatab(nthet)
      COMMON/blk2/fkbet2(10),fkbtx02,fkbty02,pervo,betao,
     .            pervpower,quadfld(10),fkfodo(10)
      COMMON/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
      COMMON/blk11/jstep,jphase,jfend,jdbeg,jdend,jlatt,dzmag,dzdrift
      COMMON/blk12/dz,dz0,dzpre
      COMMON/blk30/dtheta,costwotheta,sintwotheta,thetamax,thetarms,
     .ilattno,dthetatab
      COMMON/blk31/iprint,nprint
c
      IF (0.LE.jstep.and.jstep.LT.jfend) THEN
         dz=dzmag
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         fkbtx02=fkbet2(ilattno)
         fkbty02=-fkbet2(ilattno)
```



```
         jstep=jstep+1
         RETURN
      ENDIF
c
      IF (jstep.EQ.jdbeg)THEN
         ilattno=ilattno+1
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
      ENDIF
c
      IF (jdbeg.LE.jstep.and.jstep.LT.jdend) THEN
         dz=dzmag
         fkbtx02=-fkbet2(ilattno)
         fkbty02=fkbet2(ilattno)
         jstep=jstep+1
         RETURN
      ENDIF
c
      IF (jstep.GE.jlatt) THEN
         dz=dzmag
         jstep=1
         ilattno=ilattno+1
         dtheta=dthetatab(ilattno)
         costwotheta=DCOS(2.*dtheta)
         sintwotheta=DSIN(2.*dtheta)
         fkbtx02=fkbet2(ilattno)
         fkbty02=-fkbet2(ilattno)
         RETURN
      ENDIF
c
      dz=dzdrift
      fkbtx02=0.0d0
      fkbty02=0.0d0
      jstep=jstep+1
      RETURN
      END
c*****************************************************************
      FUNCTION fkbeta02x(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk1/emitg2, emith2, beta
      COMMON/blk2/fkbet2(10),fkbtx02,fkbty02,pervo,betao,
     .            pervpower,quadfld(10),fkfodo(10)
      fkbeta02x=-fkbtx02
      RETURN
      END
c*****************************************************************
      FUNCTION fkbeta02y(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk1/emitg2, emith2, beta
      COMMON/blk2/fkbet2(10),fkbtx02,fkbty02,pervo,betao,
     .            pervpower,quadfld(10),fkfodo(10)
      fkbeta02y=-fkbty02
      RETURN
      END
c*****************************************************************
      SUBROUTINE betavals(z)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk3/betaconl,halfbetaconl,betaovbetao,dbdzovb
c betaconl=(energyf/energyo-1.0d0)/(nlaps*flength)
c halfbetaconl=.5*betaconl
      betaovbetao=DSQRT(1.0d0 + betaconl*z)
      dbdzovb=halfbetaconl/(1.0d0 + betaconl*z)
      RETURN
      END
c*****************************************************************
c**********Approximated Constraint Equations**********************
c*****************************************************************
      FUNCTION DEEX2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk1/emitg2, emith2, beta
c
c23456789012345678901234567890123456789012345678901234567890123456789012
```

```
c
        DEEX2=(4.*beta**4*dxpyp**2*dxxp**2 - 8.*beta**4*dxpyp**3*dxy +
     -  8.*beta**4*dxpy*dxpyp**2*dxyp +
     -  8.*beta**4*dxpyp*dxp2*dxxp*dxyp - 4.*beta**4*dxp2**2*dxyp**2 -
     -  8.*beta**4*dxp2*dxxp**2*dyp2 + 8.*beta**4*dxpyp*dxp2*dxy*dyp2 -
     -  8.*beta**4*dxpy*dxp2*dxyp*dyp2 + 4.*beta**4*dxpy**2*dyp2**2 -
     -  8.*beta**4*dxpy*dxpyp*dyp2*dyyp + 4.*beta**4*dxpyp**2*dyyp**2 +
     -  0.5*beta**2*dxpyp**2*emitg2 - 0.5*beta**2*dxp2*dyp2*emitg2)/
     -  (beta**4*dxp2*(8.*dxpyp**2 - 8.*dxp2*dyp2))
c
      RETURN
      END
c
c*********************************************************************
      FUNCTION DEEY2(dxxp,dxp2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
      IMPLICIT DOUBLE PRECISION (a-h,o-z)
      COMMON/blk1/emitg2, emith2, beta
c
c23456789012345678901234567890123456789012345678901234567890123456789012
c
        DEEY2=0.5*(-1.*beta**2*
     -   (256.*beta**2*dxpyp**2*dxxp**2 - 512.*beta**2*dxpyp**3*dxy +
     -    512.*beta**2*dxpy*dxpyp**2*dxyp -
     -          512.*beta**2*dxpyp*dxp2*dxxp*dxyp +
     -          256.*beta**2*dxp2**2*dxyp**2 +
     -          512.*beta**2*dxpyp*dxp2*dxy*dyp2 -
     -          512.*beta**2*dxpy*dxp2*dxyp*dyp2 -
     -          256.*beta**2*dxpy**2*dyp2**2 +
     -          512.*beta**2*dxpy*dxpyp*dyp2*dyyp +
     -          256.*beta**2*dxpyp**2*dyyp**2 -
     -   512.*beta**2*dxp2*dyp2*dyyp**2 + 32.*dxpyp**2*emitg2 -
     -          32.*dxp2*dyp2*emitg2))/
     -     (beta**4*dyp2*(-256.*dxpyp**2 + 256.*dxp2*dyp2))
c
      RETURN
      END
c*********************************************************************
c******  Generalized Emittance Calculations  ***********************
c*********************************************************************
c
        FUNCTION EG(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
        IMPLICIT DOUBLE PRECISION (a-h,o-z)
        COMMON/blk1/emitg2, emith2, beta
c
        EG=16*beta**2*(dxpyp*dxy - dxpy*dxyp) +
     -   8.*beta**2*(-dxxp**2 + dxp2*dx2) + 8.*beta**2*
     -   (-dyyp**2 + dyp2*dy2)
c
        RETURN
        END
c
c*********************************************************************
        FUNCTION EH(dx2,dxxp,dxp2,dy2,dyyp,dyp2,dxy,dxpy,dxyp,dxpyp)
        IMPLICIT DOUBLE PRECISION (a-h,o-z)
        COMMON/blk1/emitg2, emith2, beta
c
        EH=(256*beta**4*(dxpyp**2*dxy**2 - 2*dxpy*dxpyp*dxy*dxyp +
     -        dxpy**2*dxyp**2 + 2*dxpy*dxxp*dxy*dyp2 -
     -        dxp2*dxy**2*dyp2 -
     -        dxpy**2*dx2*dyp2 - 2*dxpyp*dxxp*dxy*dyyp -
     -        2*dxpy*dxxp*dxyp*dyyp + 2*dxp2*dxy*dxyp*dyyp +
     -        2*dxpy*dxpyp*dx2*dyyp + 2*dxpyp*dxxp*dxyp*dy2 -
     -        dxp2*dxyp**2*dy2 - dxpyp**2*dx2*dy2) +
     -     256*beta**4*(-dxxp**2 + dxp2*dx2)*(-dyyp**2 +
     -     dyp2*dy2))**0.5
c
        RETURN
        END
c*********************************************************************
```

E.4 "PEPPERPOT"

"PEPPERPOT" is a code written in IDL. It was modified from A. Paul's IDL pepperpot image analysis code used with images of electron beams from the ETA-II accelerator. It was modified to analyze ion beams passed through a rectangular pattern pepperpot mask more suitable to the analysis of a beam produced by an alternating gradient focusing lattice. A modification was made to calculate the transverse cross-moments and display these correlations. The ability to fit a gaussian curve to beamlet spot cross-sections was also added.

```
; GBI.PRO
; ---------------------------------------------------------------------
;
; center (1..nbeam,6) == xpix,ypix,xmin,xmax,ymin,ymax
; bradius(0..nbeam-1) == radius of beamlet, center of beamlet=0
; jradius(0..nbeam-1) == radial position of beamlet on mask
; data   (1..nbeam,9) == peak,x1,x2,xabs,y1,y2,yabs,totalint,background
; phase  (1..nbeam,6) == xabs,xp1,xp2,yabs,yp1,yp2
;                        -3  / 0,0,xav,xpav,yav,ypav (centroid of phase space)
;                        -2  / r12,r34,xs,xps,ys,yps (phase space plot scales)
;                        -1  / s11,s12,s22,s33,s34,s44
; emitn  (1..nbeam)  == emitx,emity for each beamlet
; mask(*,3)  == x,y,radius / 1:nbeam are for beamlets, mask(0,2)=input radius
;                  mask(0,0) not used, mask(0,1)=mask rotaton angle
; plot_lets(1..nbeam) == beamlets_number_plotted , beamlet_ignor_flag
;                  / (nbeam+1),1=label_ellipse_flag
; prange(50) ==(0..3)=profile(xmin,xmax,ymin,ymax), (4)=nsave,(5..49)overplots
;
;---------------------------------------------------------------------
; Modified from Pepperpot.pro, A.C.Paul, 5/9/90
; Read *.RAS files with pepper pot images
; plot C)ontour, S)urface, B)eamlets, E)llipse, P)rofiles, J)ness
; Calculate beam brightness and phase space occupied by the beamletts.
;
;---------------------------------------------------------------------
;
;  1) analyze, 4352          / Take x,y cuts of beamlets, prints half size
;   ) adjust_centers, 3223
;  2) autoview, 712          / TV, surface, and contour plots of many files
;  3) data_input, 3724       / Interactive data input, change parameters
;  4) decode, 1339           / Input string ss, returns array of numbers
;  5) default_data, 4772     / Set initial parameters
;  6) do_levels, 283         / calculate emittance for levels 10,.90%
;  7) do_profiles, 4559      / Cuts in x and y
;  8) doplots, 5052          / plot beamlets,contour,surface,mask,window, ....
;  9) dump_image, 3459       / Prints actual IMAGE sub area in pixels, raw data
; 10) dumpfile, 567          / saves and restores parameters 1/4/91
;   ) fit_gauss, 4240
; 11) fix_data, 833          / Fix window, image and mask arrays
;   ) fix_beam, 963          / Fix beam centers, add or delete beamlet windows an
<d mask
; 12) fix_image, 1078        / Fixes the image array, subtract background, etc,
; 13) fix_mask, 893          / fix mask
;   ) gen_rectangular, 2794 / generates cartesian mask
; 14) gen_round, 2659        / generates round mask pattern
; 15) gen_mask, 3155         / generates mask hole pattern (x,y) at L=0, mask(*)
; 16) get_centers, 3289      / Finds beamlet center(*) and sub_window in pixels
; 17) get_beamlets, 1453     / Finds beamlet width at level in sub_window, data(*
<)
; 18) get_emittance, 1673    / Calculates emittance of beamlets, phase(*),emitn(*
<)
;   ) get_halo, 383          /
;   ) get_rebin, 1428
;   ) get_scale, 5024
;   ) get_sigma, 1569
; 19) function get_word, 269 / returns word from string
; 20) greetings, 234         / Print inital information on help
; 21) helpme, 137            / Online help package
; 22) listdata, 3886         / List beamlet centers, and data() array to screen
; 23) move_curser, 3836      / Move cursor on contour or window plot, set surface 1
; 24) movie, 1870            / Movie of cuts every 5 pixels in the x or y directio
<n
; 25) next_file, 3571        / Gets next sequential file'
; 26) function odd,n         / n=5 x=5.00 xh=2 x-2nh=1 / n=6 x=6.00 xh=3 x-2xh=0
; 27) output, 1214           / Writes file printname with beamlet data
; 28) pepperpot, 5274        / Main program
;   ) plot_all_beam, 2437
; 29) plot_6d_space, 2049    / Plots xy', yx', x'y', and xy of beamlets
; 30) plot_beamlet, 2478     / Plot surface plot of given beamlet
; 31) plot_brightness, 3643  / Plots beamlet brightness vs radius
; 32) plot_ellipse, 1984     / Plot phase ellipse on data plot from sigma matrix
; 33) plot_mask, 2586        / plot mask hole pattern on contour, center plot
; 34) plot_phase_space, 2218 / Plots x and y phase space
```



```
; 35) plot_tv_frame, 1158 / Plots 'TV' of image, overlay windows
; 36) plot_window, 3517   / Plots location and beamlet number on pixel grid
; 38) read_titles, 4979   / Reads the title lines for a sequence of files.
; 39) readdata, 3593      / Reads initial data, window,level,filenames etc.
; 40) readfile, 1363      / Reads image.RAS file
; 41) readme, 197         / Reads help file 'pepperpot.readme'
;     rotate_mask, 862
; 42) set_contour, 4684   / Finds center for contour plot from mouse position
; 43) set_path_name, 4962 / Sets directory path name for Save and Ras files'
; 44) set_surface, 4707   / Finds center for surface plot from mouse position
; 45) show_list, 4882     / List to screen plots that will be generated by '*'
; 46) smooth_data, 4736   / Use sliding box car smooth on image
;   ) summary, 4146
; 47) summaryr, 4058  / Plots Peak,Total beamlet intensity, x,y and radius
; 48) toggle_windows, 4935  / Shows and hides windows 0, 2 and 4
;   ) vary_mask, 3049
; 49) write_scales, 2205
; 50) zero_image, 3424   / Allows 'zeroing' of a bad area of the image
;
;---------------------------------------------------------------------
;
; save 'dumpfile' consists of
;     printf,1,filename
;     printf,1,title
;     printf,1,imin,imax,background,xdim,ydim
;     printf,1,xcmpix,ycmpix,drift,rhole,separation,thickness
;     printf,1,window,xwindow,ywindow,level
;     printf,1,current,energy,betagamma,restm,brightness,emitx,emity
;     printf,1,lpass,nholes,emit_scl(4),bright_scl(4)
;     printf,1,nbeam,nmask,typemask
;     printf,1,'DATA()'
;     printf,1,data(0:nbeam*2,*)              0..
;     printf,1,'CENTER()'
;     printf,1,center(0:nbeam*2,*)            0..
;     printf,1,'MASK()'
;     printf,1,nx,ny,junk,junk
;     printf,1,mask(1:nmask,*)                0..
;
;---------------------------------------------------------------------
;
pro helpme
common com1,filename,dumpname,printname,dataname,title
for k=1,9 do print,' '
print,' '
print,'< ************** GBI  pepperpot.pro (5/14/97) ************** >'
print,' '
print,'ETAJR LOS calibrations x,y(cm/pixel) are:'
print,'          1320(0.0157,0.0130), 1726(0.0272,0.0185), 1806(0.0150,0.0119)'
print,'a)nalyze  Do cuts in x and y of given beamlet'
print,'A)utoview Hardcopy of TV, Surface, and Contour plots, many files '
print,'b)eamlet  Plot given beamlet 1..max'
print,'B)        Regenerates beamlets, Ba)ckground subtraction'
print,'c)ontour  Plot contours of image, Enter ix1 ix2 iy1 iy2, cs auto scales'
print,'d)ata     Input E(MeV), current, mask, pixels, window, Image, and level'
print,'e)llipse  Plot x-xp and y-yp phase space, es input fix scales for e plot
<'
print,'          e6 plots (x-yp),(y-xp),(xp-yp),(xy), e26,e46 to windows 2 and
<4
print,'el        plot emittance vs level 10,.90%, ele 5,10,20...,90,95%'
print,'e<wids>   plot beamlets transformed to location of waist,image,distance'
print,'E)nhance  Scales image() array to 0..254 pixels'
print,'fix       c)enter(), m)ask(), i)mage() arrays'
print,'g)et_halo Remove beamlets from image, leaves background halo.'
print,'h)elp     'h' this online help, 'H' will read file pepperpot.readme'
print,'i)nit     Mask hole pattern, iw mask&window centers superimposed'
print,'          ic)rossover, is)ift by given x,y, iv)ary_mask'
print,'          ig)enerate click on holes, igx generate mask of nx*ny holes'
print,'          ir)generate round mask of nbeam holes, ik)rotate mask pattern'
print,'j)ness    Plot brightness, js input fix scales for all J plots'
print,'l)ist     List data, intensity of beamlets, x,y)centroid etc.'
print,'m)ove     Move cursor, left button returns x,y, right button returns'
print,'M)ovie    x,y profiles every x=5..xdim in steps of 5, M,Mx,My, Mxa adds'
print,'n)ext     File, new image'
```

```
print,'o)utput    Writes cent,bright,emit file ''+printname+'' 'O' forces new'
print,'p)rofile   Cut in image, e.g., x104 (xcut @ y104) y55 (ycut @ x55), pio'
print,'           po)verplot profiles, pr)overplot range profiles, kmin,max,step
+'
print,'P)ath      Set file directory path, e.g., '.ras files 'R355/' '
print,'           Ps set path for 'save' files'
print,'q)uit      Exits this program after allowing for 'saving' of files.'
print,'R)         ??? '
print,'s)urface   Surface plot of sub_window region. Enter --> ix1 ix2 iy1 iy2'
print,'S)mooth    Smooth image, if smooth<3 then regenerate centers etc'
print,'Sa         Adjust beamlet centers'
print,'t)V        Produce a TV image of frame, e2,i2,j2,t2,v2,or w2 ==> window 2
+'
print,'           ts scales image to 0..255,   e2,i4,j4,t4,v4,or w4 ==> window 4
+'
print,'T)itles    Reads title lines for a sequence of files'
print,'u)abort    abort in get_beamlet at beamlet=iabort'
print,'v)iew      Summary of all beamlets, radius, intensity, peak. vr vs radius
+'
print,'w)indows   Plot window of beamlets and centers with indices, wc)entmeters
+'
print,'           wn)no frames, just center and index number into window 2.'
print,'W          Toggles s)how and h)ide of windows 2 and 4,  Ws2,Ws4,Wh2,Wh4'
print,'x)ecute    runs beamlet and emittance procedures.'
print,'y)dump     sub region of image to ''+dumpname+''          WO)verplot c'
print,'z)ero      sub region of image '
print,'*)         Print plots now and return to program, ** purges all plots'
print,'           *l show to screen plot that will be generated by ''
print,'-----------------------------------------------------------------'
return
end
;
;-----------------------------------------------------------------------
;
pro readme
common com3,ssin,pview,ch2
;
  ss='' & ch=' ' & line=0
  openr,1,'pepperpot.readme'
    if ch2 ne ' ' then $
      begin
        lsx=strlen(ssin)-1
        schk=strmid(ssin,1,lsx)
        while not eof(1) do begin
          readf,1,ss
            if (strmid(ss,0,1) eq '@') then $
              if (strmid(ss,1,lsx) eq schk) then goto,listit
              if (strmid(ss,1,1) eq '@' ) then goto,exit
        end
        listit: readf,1,ss
          while (strmid(ss,0,1) ne '@') do $
            begin
              print,ss & readf,1,ss
              line=line+1
              if line gt 40 then begin
                line=0 & read,'Continue <,n> ',ch & if ch eq 'n' then goto,exit
              end
            end
        goto,exit
      end $
    else while not eof(1) do begin & wshow,0,0
      readf,1,ss & print,ss & line=line+1
      if line eq 43 then begin
        line=0 & read,'Continue <,n> ',ch & if (ch eq 'n') then goto,exit & end
    end
exit: close,1
return
end
;
;-----------------------------------------------------------------------
;
pro greetings,cerror
common com3,ssin,pview,ch2
```



```
common com15,version,comhead,comdata,moments
;
ss=strarr(12)
ll=strpos(comdata(0),'VERSION') & ll=strlen(comdata(0))
ss( 0)='.=========================================.'
ss( 1)='   Program to prosses *.RAS files and calculate from the TV image'
ss( 2)='   the beam emittance and brightness for 'PEPPER_POT' beamlets.'
ss( 3)='                                                                '
ss( 4)='   Greetings / pepperpot.pro                                    '
ss( 5)='          Enter Capital H for 'pepperpot.readme'                '
ss( 6)='          Enter Capital Hx for help on subject x =command character.'
ss( 7)='                any other entry starts program                  '
ss( 8)='                                                                '
ss( 9)='                 '+strmid(comdata(0),ll,ll-11)                  '
ss(10)='                                                                '
ss(11)='=========================================='
  yyx=0.9
  for k=0,11 do begin
    xyouts,0.12,yyx,ss(k),color=0,size=1.25,/norm & yyx=yyx-0.06
  end
  ss='' & read,'Your entry please <H)ELP, R)estart, LF> ==> ',ss
  ch=strmid(ss,0,1)
  if ch eq 'H' then begin
    if strlen(ss) gt 1 then ch2=strmid(ss,1,1) else ch2=' ' & readme
  end else if ch eq 'R' or ch eq 'r' then begin
    print,'RESTART from file'
    cerror='R'
  end else helpme
return
end
;
;-----------------------------------------------------------------------
;
function get_word,ss,nword
;
  i=strpos(ss,' ')
  if i lt 0 then read,' ',word $
  else begin
    while strmid(ss,i,1) eq ' ' do i=i+1
    word=strmid(ss,i,strlen(ss)-i)
  end
;
return,word
end
;
;-----------------------------------------------------------------------
;
pro do_levels,ntime
common com1,filename,dumpname,printname,dataname,title
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,che16d,chelacl
;
  if nmask eq 0 then begin & print,'Must generate mask,' & return & end
  if strmid(ssin,2,1) eq 's' then begin
    k1=0 & k2=10 & lev=10*indgen(11) & emt=fltarr(11)
    lev(0)=5 & lev(10)=95
  end else begin
    k1=1 & k2=9 & lev=10*indgen(10) & emt=fltarr(10)
  end
;
  emx=emt & emy=emt
  for k=k1,k2 do begin
    level=lev(k)
      get_beamlets,0
        get_emittance,' '
    xemav=0 & yemav=0
    for i=1,nbeam do begin
        xemav=xemav+1e6*emitn(i,0)
```

```
282          yemav=yemav+1e6*emitn(i,1)
283        end
284        xemav=xemav/nbeam & yemav=yemav/nbeam
285        ave=0.5*(xemav+yemav)
286        emx(k)=xemav & emy(k)=yemav & emt(k)=ave
287        if ntime eq 0 then $
288          print,'  level '+string(level,format='(i3)')+'  '+comdata(8)
289      end
290        plot,lev(k1:k2),emt(k1:k2),xrange=[0,100],psym=4,$
291          xtitle='Percent Level',ytitle='Beam Emittance (mm-mr)',$
292            title=title+' '+filename
293        oplot,lev(k1:k2),emt(k1:k2)
294        oplot,lev(k1:k2),emx(k1:k2),psym=4
295        oplot,lev(k1:k2),emy(k1:k2),psym=4
296        xx=fltarr(2) & yy=xx
297        for k=k1,k2 do begin
298          xx(0)=lev(k) & xx(1)=xx(0) & yy(0)=emx(k) & yy(1)=emy(k)
299          oplot,xx,yy
300        end
301        if emx(5) gt emy(5) then ix=5 else ix=-5
302        xyouts,lev(5)+ix,emx(5),'X'
303        xyouts,lev(5)-ix,emy(5),'Y'
304    ;
305        if ntime eq 0 then begin & xx=0.4 & yy=0.93 & end $
306                        else begin & xx=0.1 & yy=-0.03 & end
307        yy=yy-0.03 & xyouts,xx,yy,comdata(1),/norm
308        yy=yy-0.03 & xyouts,xx,yy,comdata(4),/norm
309        yy=yy-0.03 & xyouts,xx,yy,comdata(6)+comdata(7),/norm
310        yy=yy-0.03 & xyouts,xx,yy,comdata(9),/norm
311        yy=yy-0.03
312        if mask(0,1) ne 0 then xyouts,xx,yy,'Mask rotation '+comdata(19),/norm
313    ;
314      level=10
315        get_beamlets,0
316          get_emittance,' '
317    return
318    end
319    ;
320    ;-----------------------------------------------------------------------
321    ;
322    pro label_frames,ntime,pixsum,heysum,imsum,ssize
323    ;
324    common com1,filename,dumpname,printname,dataname,title
325    common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
326    common com5,plot_lets,center,maxb,nbeam
327    ;
328      itot=0 & for k=1,nbeam do itot=itot+data(k,7)
329      if ntime eq 6 then begin
330        print,'                  image        halo        pixsum     itotal(beam)>bg'
331        print,'               ---------    ---------    ---------    ---------'
332        print,'Totals: '+$
333                          '  '+string(imsum,format='(f10.0)')+$
334                          '  '+string(heysum,format='(f10.0)')+$
335                          '  '+string(pixsum,format='(f10.0)')+$
336                          '  '+string(itot,format='(f10.0)')+$
337                          '   im-ha='+string(imsum-heysum,format='(f10.0)')
338        return
339      end
340    ;
341      if ntime eq 1 then iy=-0.07 else iy=0.90
342      d1=imsum-heysum
343      d2=heysum-pixsum
344      xyouts,0.18,iy,filename,/norm
345      xyouts,0.52,iy,ssize,/norm
346      xyouts,0.18,iy-0.03,'Total pixels in image                  : '+$
347                          string(imsum,format='(f10.0)'),/norm
348      xyouts,0.18,iy-0.06,'Total pixels in halo                   : '+$
349                          string(heysum,format='(f10.0)')+'     '+$
350                          string(d1,format='(f10.0)'),/norm
351      xyouts,0.18,iy-0.09,'Total pixels in halo above background: '+$
352                          string(pixsum,format='(f10.0)')+'     '+$
353                          string(d2,format='(f10.0)'),/norm
354      xyouts,0.18,iy-0.12,'Total pixels in beamlets above backg : '+$
```



```
355                          string(itot,format='(f10.0)'),/norm
356    ;
357    return
358    end
359    ;
360    ;-----------------------------------------------------------------------
361    ;
362    pro get_halo,ntime
363    ;
364    common com1,filename,dumpname,printname,dataname,title
365    common com2,xdim,ydim,imswch,image,iobject
366    common com3,ssin,pview,ch2
367    common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
368    common com5,plot_lets,center,maxb,nbeam
369    common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
370    common com8,window,xwindow,ywindow,chwin2,level
371    ;
372      if strpos('24',ch2) lt 0 then is0=1 else begin  ; g2 g21 etc.
373        is0=0 & ch2=strmid(ssin,2,1)
374      end
375      ss='y'
376      halo=image
377    ;
378      for k=1,nbeam do begin
379        nx1=center(k,2) & nx2=center(k,3) & ny1=center(k,4) & ny2=center(k,5)
380        bg1=total(halo(nx1:nx2,ny1))/(nx2-nx1+1)
381        bg2=total(halo(nx1:nx2,ny2))/(nx2-nx1+1)
382        bg3=total(halo(nx1,ny1:ny2))/(ny2-ny1+1)
383        bg4=total(halo(nx2,ny1:ny2))/(ny2-ny1+1)
384        bg=0.25*(bg1+bg2+bg3+bg4)
385        halo(nx1:nx2,ny1:ny2)=bg
386        if ch2 eq '1' then $
387             print,'beamlet: '+string(k,format='(i3)')+'  bg1,2,3,4='+$
388                        ' '+string(bg1,format='(i3)')+' '+string(bg2,format='(i3)')+$
389                        ' '+string(bg3,format='(i3)')+' '+string(bg4,format='(i3)')+$
390                        '  average: '+string(bg,format='(i3)')
391    ;
392    ;     floor=fltarr(nx2-nx1+1,ny2-ny1+1)
393    ;     halo(nx1:nx2,ny1:ny2)=byte(floor)
394    ;
395      end
396      xx=indgen(512)
397      k=size(halo) & kxmax=k(1) & kymax=k(2)
398      isx=500 & isy=450
399      if kxmax lt isx then begin & isx=400 & isy=440 & end
400      halo=smooth(halo(0:isx-1,0:isy-1),5)
401      k=size(halo) & kxmax=k(1) & kymax=k(2)
402      nx1=min(center(1:nbeam,2))-20
403      nx2=max(center(1:nbeam,3))+20
404      ny1=min(center(1:nbeam,4))-20
405      ny2=max(center(1:nbeam,5))+20
406      ssize='{ '+string(nx1,format='(i3)')+':'+string(nx2,format='(i3)')+','+$
407                string(ny1,format='(i3)')+':'+string(ny2,format='(i3)')+' }'
408      if nx1 lt 0 then nx1=0 & if ny1 lt 0 then ny1=0
409      if nx2 gt kxmax than nx2=kxmax & if ny2 gt kymax then ny2=kymax
410    ;
411    ; SURFACE PLOT OF HALO
412    ; --------------------
413      iz=max(image)
414      ii=rebin(halo,isx/5,isy/5)
415      ax=nx2-nx1+1.0 & ay=ny2-ny1+1.0
416      py=0.5*(total(halo(nx1,ny1:ny2))+total(halo(nx2,ny1:ny2)))
417      px=0.5*(total(halo(nx1:nx2,ny1))+total(halo(nx1:nx2,ny2)))
418      p0=0.5*(px*ay+py*ax)
419      p1=min(halo(nx1:nx2,ny1:ny2))
420      p2=max(halo(nx1:nx2,ny1:ny2))
421      p3=total(halo(nx1:nx2,ny1:ny2))
422      p4=p3/(ax*ay) & p5=p0/(ax*ay)
423      my=1.2*p2
424      surface,ii
425      xyouts,0.1,0.90,filename,/norm
426      xyouts,0.1,0.87,$
427          'Halo min,max: '+string(p1,format='(i3)')+string(p2,format='(i4)')+$
```

```
428          '    total'+ssize+'='+string(p),format='(f10.0)'}+$
429          '     ave='+string(p4,format='(f5.1)'),/norm
430      xyouts,0.1,0.84,$
431          'Background'+ssize+' total='+string(p0,format='(f10.0)'}+$
432          '     ave='+string(p5,format='(f5.1)'),/norm
433      ;
434      ; CONTOUR PLOT OF HALO
435      ; --------------------
436        if ntime eq 0 then begin
437          ss=' ' & read,'Contour plot <y,n,q)uits,a)bort ',ss
438          if ss eq 'a' then abort
439          if ss eq 'q' then return
440        end
441        if ss ne 'n' then begin
442          contour,halo(nx1:nx2,ny1:ny2),nlevels=20,xstyle=1,ystyle=1
443          xyouts,0.1,0.90,filename,/norm
444          xyouts,0.1,0.87,$
445            'Halo min,max: '+string(p1,format='(i3)')+string(p2,format='(i4)'),/norm
446          if ntime eq 0 then wset,2
447          contour,image(nx1:nx2,ny1:ny2),nlevels=20,xstyle=1,ystyle=1
448        end
449        if ntime eq 0 then begin & wshow,2 & wshow,4 & end
450      ;
451      ; TV PLOT OF HALO
452      ; ---------------
453        if ntime eq 0 then begin
454          ss=' ' & read,'TV plot of halo <y,n,q)uits> ',ss
455          if ss eq 'q' then return
456        end
457        if ss ne 'n' then begin
458          if ntime eq 0 then wset,2
459            tvscl,halo(nx1:nx2,ny1:ny2)
460        end
461      ;
462      ; Vertical slices through halo
463      ; ---------------------------
464        if ntime eq 0 then begin
465          wset,0
466          ss=' ' & read,'Vertical slices <y,n,q)uits,a)bort ',ss
467          if ss eq 'a' then abort
468          if ss eq 'q' then return
469        end
470        hey=float(halo)
471        if ss ne 'n' then begin
472      ;   get average background
473          bg=fltarr(kxmax)
474          for k=5,ny1 do bg=bg+hey(*,k)
475          for k=ny2,kymax-5 do bg=bg+hey(*,k)
476          bg=byte(bg/(ny1+kymax-ny2))
477          pixsum=0
478          heysum=total(hey(nx1:nx2,ny1:ny2)) & imsum=total(image(nx1:nx2,ny1:ny2))
479          for k=ny1,ny2 do pixsum=pixsum+total(hey(nx1:nx2,k)-bg(nx1:nx2))
480          if ntime eq 0 then label_frames,6,pixsum,heysum,imsum,ssize
481      ;
482          if is0 eq 1 then begin
483          if ntime eq 0 then wset,0
484          plot,xx,halo(*,(ny1+ny2)/2),xstyle=1,xrange=[0,xdim],yrange=[0,my],/nodata
485            for k=ny1,ny2,5 do oplot,xx,halo(*,k)
486              label_frames,ntime,pixsum,heysum,imsum,ssize
487          end
488      ;
489          if ntime eq 0 then wset,2
490          plot,xx,halo(*,(ny1+ny2)/2),xstyle=1,xrange=[0,xdim],yrange=[0,my]
491            for k=5,ny1,5 do oplot,xx,halo(*,k)
492            for k=ny2,kymax-5,5 do oplot,xx,halo(*,k)
493              label_frames,ntime,pixsum,heysum,imsum,ssize
494            plots,[nx1,nx1],[0,0.5*my] & plots,[nx2,nx2],[0,0.5*my]
495      ;
496          if ntime eq 0 then wset,4
497          plot,xx,hey(*,(ny1+ny2)/2)-bg,xrange=[0,xdim],yrange=[0,my],xstyle=1
498            for k=ny1,ny2,5 do oplot,xx,hey(*,k)-bg
499              label_frames,ntime,pixsum,heysum,imsum,ssize
500            plots,[nx1,nx1],[0,0.5*my] & plots,[nx2,nx2],[0,0.5*my]
```



```
501        end
502      ;
503      ; Horizontal slices through halo
504      ; ---------------------------
505        if ntime eq 0 then begin
506          ss=' ' & read,'Horizontal slices <y,n,a)bort ',ss & if ss eq 'a' then abort
507        end
508        if ss ne 'n' then begin
509      ;   get average background
510          bg=fltarr(kymax)
511          for k=5,nx1 do bg=bg+hey(k,*)
512          for k=nx2,kxmax-5 do bg=bg+hey(k,*)
513          bg=byte(bg/(nx1+kxmax-nx2))
514          pixsum=0
515          heysum=total(hey(nx1:nx2,ny1:ny2)) & imsum=total(image(nx1:nx2,ny1:ny2))
516          for k=nx1,nx2 do pixsum=pixsum+total(hey(k,ny1:ny2)-bg(ny1:ny2))
517          if ntime eq 0 then label_frames,6,pixsum,heysum,imsum,ssize
518      ;
519          if is0 eq 1 then begin
520          if ntime eq 0 then wset,0
521            plot,xx,hey((nx1+nx2)/2,*),xstyle=1,xrange=[0,ydim],yrange=[0,my],/nodata
522              for k=nx1,nx2,5 do oplot,xx,hey(k,*)
523                label_frames,ntime,pixsum,heysum,imsum,ssize
524          end
525      ;
526          if ntime eq 0 then wset,2
527            plot,xx,hey((nx1+nx2)/2,*),xstyle=1,xrange=[0,ydim],yrange=[0,my]
528              for k=5,nx1,5 do oplot,xx,hey(k,*)
529              for k=nx2,kxmax-5,5 do oplot,xx,hey(k,*)
530                label_frames,ntime,pixsum,heysum,imsum,ssize
531            plots,[ny1,ny1],[0,0.5*my] & plots,[ny2,ny2],[0,0.5*my]
532      ;
533          if ntime eq 0 then wset,4
534            plot,xx,hey(*,(nx1+nx2)/2)-bg,xrange=[0,ydim],yrange=[0,my],xstyle=1
535              for k=nx1,nx2,5 do oplot,xx,hey(k,*)-bg
536                label_frames,ntime,pixsum,heysum,imsum,ssize
537            plots,[ny1,ny1],[0,0.5*my] & plots,[ny2,ny2],[0,0.5*my]
538        end
539      ;
540      if ntime eq 0 then wset,0
541      return
542      end
543      ;
544      ;-----------------------------------------------------------------------
545      ;
546      pro dumpfile,fname,csw
547      ;                          '--- s)ave,   r)ecall
548      common com1,filename,dumpname,printname,dataname,title
549      common com2,xdim,ydim,imswch,image,iobject
550      common com3,ssin,pview,ch2
551      common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
552      common com5,plot_lets,center,maxb,nbeam
553      common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
554      common com8,window,xwindow,ywindow,chwin2,level
555      common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
556      common com10,xcent,ycent,nxcent,nycent,imin,imax,background
557      common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
558      common com12,nb,type_slice,xslice,yslice,prange,chsave1,chsave2,chsave3
559      common com15,version,comhead,comdata,moments
560      common com16,xwcent,ywcent,xpix0,ypix0,pixsize
561      common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelsc1
562      common com20,scales
563      common com21,nmax,masklist
564      ;
565        if strlen(fname) eq 0 then begin
566          read,'Enter dump filename: q)uits ',fname
567          if strlen(fname) eq 0 then return
568          if fname eq 'q' then return
569        end
570        if (strpos(fname,'/') eq -1) and $
571          (strlen(comdata(16)) gt 1) then fname=comdata(16)+fname
572      ;
573        if csw eq 's' then begin & ss='' ; SAVE file fname for restart
```

```
    junk=0
    openw,1,fname, /variable
      printf,1,filename
      printf,1,comdata(17)
      ss=string(imin,format='(i5)')+string(imax,format='(i5)')+$
         string(background,format='(i5)')+$
            string(xdim,format='(i5)')+string(ydim,format='(i5)')
      printf,1,strcompress(ss)+$
        '                                    \ imin,imax,backg,xdim,ydim'
      ss=string(xcmpix)+string(ycmpix)+string(drift)+$
        string(rhole)+string(separation)+string(thickness)
      printf,1,strcompress(ss)+'     \ x,ycmpix,drift,rhole,sep'
      ss=string(window)+string(xwindow)+string(ywindow)+string(level)
      printf,1,strcompress(ss)+$
        '                                    \  window,xwin,ywin,level'
      ss=string(current)+string(energy)+string(betagamma)+$
        string(restm)+string(brightness)+string(emitx)+string(emity)
      printf,1,strcompress(ss)+'   \ Amps,MeV,betag,xyemit'
      ss=string(ipass(0),format='(f5.2)')+string(ipass(1),format='(f5.2)')
      for k=0,7 do ss=ss+string(scales(k),format='(f5.2)')
      printf,1,strcompress(ss)+'   \ ipass(),scales(8)'
      ss=string(nbeam)+string(nmask)+string(typemask)
      printf,1,strcompress(ss)+$
        '                                    \ nbeam,nmask,typemask'
      printf,1,'DATA()'
      printf,1,data(0:nbeam+2,*)
      printf,1,'CENTER()'
      printf,1,center(0:nbeam+2,*)
      printf,1,'MASK()'
      printf,1,masklist(0,0),masklist(0,1),junk,junk
      printf,1,mask(0:nbeam,*)
    close,1 & print,'File '+fname+' written' & return
  end else $
;
  if csw eq 'r' then begin & ss='' ; RESTART from saved file
    num=fltarr(20)
    dataname=fname
    openr,1,fname
      print,'-----------------------------------------------------------------'
      readf,1,filename & print,filename
      readf,1,title    & print,title
      readf,1,ss & decode,ss,num,n & print,ss
        imin=fix(num(0)) & imax=fix(num(1)) & background=num(2)
          xdim=fix(num(3)) & ydim=fix(num(4))
      readf,1,ss & decode,ss,num,n & print,ss
        xcmpix=num(0) & ycmpix=num(1) & drift=num(2)
          rhole=num(3) & separation=num(4) & thickness=num(5)
      readf,1,ss & decode,ss,num,n & print,ss
        window=fix(num(0))
          xwindow=fix(num(1)) & ywindow=fix(num(2)) & level=fix(num(3))
      readf,1,ss & decode,ss,num,n & print,ss
        current=num(0) & energy=num(1) & betagamma=num(2) & restm=num(3)
          brightness=num(4) & emitx=num(5) & emity=num(6)
      readf,1,ss & decode,ss,num,n & print,ss+'   n='+string(n)
        ipass(0)=num(0) & ipass(1)=num(1)
          if n ge 6 then begin
            for j=0,3 do scales(j)=num(j+2)
              if n eq 10 then for j=4,7 do scales(j)=num(j+2)
          end
      readf,1,ss & decode,ss,num,n & print,ss
        nbeam=fix(num(0)) & nmask=fix(num(1)) & typemask=fix(num(2))
;
      maxb=0
      plot_lets(*,1)=1
;
      readf,1,ss & print,ss ; DATA()
        line=fltarr(nbeam+3)
          for k=0,8 do begin & readf,1,line & data(0:nbeam+2,k)=line & end
      readf,1,ss & print,ss ; CENTER()
        line=intarr(nbeam+3)
          for k=0,5 do begin & readf,1,line & center(0:nbeam+2,k)=line & end
      readf,1,ss & print,ss ; MASK()
        readf,1,nx,ny,junk1,junk2
```



```
          line=fltarr(nbeam+1)
            for k=0,2 do begin & readf,1,line & mask(0:nbeam,k)=line & end
      close,1
      masklist(0,0)=nx & masklist(0,1)=ny
      for k=1,nmask do mask(k,2)=sqrt(mask(k,0)^2+mask(k,1)^2)
      for k=0,maxptype do ptype(k)=0
;
      comdata(0)=filename+' ('+dataname+')     '+version
      comdata(1)='['+string(xdim,format='(i3)')+','+string(ydim,format='(i3)')+']
+'
      comdata(2)='['+string(imin,format='(i3)')+',+$
                 string(imax,format='(i3)')+'] '+string(background,format='(i3)')
      comdata(3)='['+string(xcmpix,format='(f7.5)')+','+$
                 string(ycmpix,format='(f7.5)')+']'
      comdata(4)='['+string(energy,format='(f8.4)')+',+$
                 string(betagamma,format='(f8.4)')+']    '+$
                 string(restm,format='(f12.3)')
      comdata(5)='['+string(2*xwindow,format='(i3)')+',+$
                 string(2*ywindow,format='(i3)')+']'
      comdata(6)='['+string(rhole,format='(f8.4)')+',+$
                 string(separation,format='(f8.4)')+',+$
                 string(thickness,format='(f8.4)')+'] N'
      comdata(7)=string(drift,format='(f8.4)')
;     comdata(8),(11),(12), and comdata(13) will be set in get_emittance
      ave=0.5*(emitx+emity)
      comdata(8)='['+string(emitx,format='(f8.3)')+',+$
                 string(emity,format='(f8.3)')+']+$
                 '    ave:'+string(ave,format='(f8.3)')+$
                 '    norm:'+string(ave*betagamma,format='(f9.4)')
      sdj=0
      comdata(9)='['+string(current,format='(f8.2)')+'(+$
                 string(ipass(0),format='(f6.3)')+'/'+$
                 string(ipass(1),format='(i4)')+'),'+$
                 string(brightness,format='(e10.2)')+'  +-'+$
                 string(sdj,format='(e8.2)')+']'
      comdata(10)=string(level,format='(i4)')
      comdata(17)=title
      comdata(19)=string(mask(0,1),format='(f8.3)')
    end
;
return
end
;
;-----------------------------------------------------------------------
;
pro autoview
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com13,sub_range
common com15,version,comhead,comdata,moments
;
  scloc=sub_range.scloc
  nfiles=0 & maxfile=3 & lprfiles=0
  mode=' ' & ntime=0 & prsw=' ' & ltype=' ' & ftype=' '
  ch=' ' & ioerror=' ' & filelist='' & ss='' & filename=''
  pool='0123456789abcdefghijklmnopqrstuvwxyz'
  path=comdata(15) & fn='filelist.dat'
  read,'Enter directory path (LF for none)                    ==> ',path
  if (strlen(path) gt 0) and $                                          ==> ',path
     (strmid(path,strlen(path)-1,1) ne '/') then path=path+'/'
  read,'Enter F)ilenames, or L)ist_filename                  <f,l> ==> ',ftype
    if ftype eq 'f' then begin
      read,'Enter *.RAS' filename          ',filename
      read,'  How many sequential files    ',nfiles
      nn=strpos(filename,'.ras')
      root=strmid(filename,0,nn-2) & suff=strmid(filename,nn-2,2)
      nc1=strpos(pool,strmid(suff,0,1)) & nc2=strpos(pool,strmid(suff,1,1))
      if nfiles lt 1 then nfiles=1
    end else if ftype eq 'l' then begin
      read,'Enter file name LF<uses '+fn+'>    ==> ',ss
        if strlen(ss) eq 0 then filelist=fn else filelist=ss
```

```
      end else return
    read,'A)utomatic, M)anule mode                          <a,m> ==> ',ch
      if ch eq 'm' then auto=0 else auto=1
    read,'T)V_image,S)urface,C)ontour                      <t,s,c> ==> ',itype
    read,'Smooth image                                   <y,n,3..9> ==> ',ss
      ch=strmid(ss,0,1) & smooth=0
        if strpos('0123456789',ch) ge 0 then begin
          smooth=byte(ch)-byte('0')
          if strlen(ss) gt 1 then smooth=10*smooth+byte(strmid(ss,1,1))-byte('0')
        end else if ch eq 'y' then read,'Enter smoothing width <3...> ',smooth
    read,'Hard copy                                          <y,n> ==> ',prsw
    if (auto eq 1) and (prsw eq 'y') then $
      read,'Enter number of files to print before timeout <1..4>     ==> ',maxfile
    print,'------------------------------------------------------------'
;
nfile=0 & if ftype eq 'l' then openr,1,filelist
next: nfile=nfile+1
    if ftype eq 'l' then begin
      readf,1,ss & if ss eq 'end' then goto,myexit else filename=ss & end $
    else if (ftype eq 'f') and (nfile gt 1) then next_file
    restart=' ' & ioerror='A' & readfile,ioerror,restart
    if smooth ge 3 then image=smooth(image,smooth)
    fix_image,'r'
    slen=strlen(title) & if slen gt 30 then slen=30
    short=' '+strmid(title,0,slen-1)
    print,string(nfile,format='(i2)')+', '+filename+' '+short
    itype=strlowcase(itype)
;
plotit:
    if itype eq 't' then begin & tv,image & xyouts,0.35,1.00,filename,/norm
      if (ntime eq 1) then begin
        xyouts,0.10,-0.03,filename,/norm & xyouts,0.10,-0.06,title,/norm & end
      end else $
;
    if itype eq 's' then begin & get_rebin,simage,cx,cy
        surface,simage,xtitle='x',ytitle='y'
          xyouts,0.14,0.90,filename,/norm & xyouts,0.14,0.87,title,/norm
        simage=0
      end else $
;
    if itype eq 'c' then begin
      step=fix(0.05*(imax-background)) & levels=fix(background)+step*indgen(20)
      ss='Grid(x1:x2,y1:y2) '+sclocs+' Levels='+$
          string(levels(0),format='(i3)')+'..'+string(levels(19),format='(i3)')+$
            ' in steps of '+string(step,format='(i3)')
      contour,simage,cx,cy,xstyle=8,ystyle=8,levels=levels,$
        xmargin=[8,8],ymargin=[4,4],xtitle='x(cm)',ytitle='y(cm)',subtitle=ss
      axis,xaxis=1,xrange=[0,xdim-1],xstyle=1,xtitle='x(pixels)',ticklen=-0.02
      axis,yaxis=1,yrange=[0,ydim-1],ystyle=1,ytitle='y(pixels)',ticklen=-0.02
      xyouts,0.14,0.85,filename,/norm & xyouts,0.14,0.82,title,/norm
      simage=0
      end
;
    if smooth ge 3 then begin
      if ntime eq 0 then yyx=0.90 else yyx=-0.10
        xyouts,0.20,yyx,'Smoothing='+string(smooth,format='(i2)'),/norm
      end
    if (ntime eq 0) then begin
      if auto eq 1 then wait,10 else begin
        read,'Next plot <y,n> ',ch & if ch eq 'n' then goto,myexit & end
    end
    if (prsw eq 'y') then $
      if (ntime eq 0) then begin & ntime=1 & root='idl.ps'
        set_plot,'ps' & device,filename=root
          print,'Hold on , generating printed output '+root & goto,plotit
        end else begin & ntime=0
          ; device,/close
          spawn,'lpr '+root & lprfiles=lprfiles+1
    if lprfiles eq maxfile then $
      begin & time=0 & lastfile='' & chf=''
      message='Waiting 3 minutes for print "QUE" to empty'
        print,'                    '+message
      pause:  wait,180 & spawn,'lpq > temp.lpq' & time=time+3 & nlpq=-1
```



```
        openr,2,'temp.lpq' & readf,2,message & message=strcompress(message)
        while not eof(2) do begin & readf,2,lastfile & nlpq=nlpq+1 & end
        close,2
        ss=' / '+string(time,format='(i2)')+' minutes, '
        chf=strmid(lastfile,0,1) & if chf eq 'a' then nlpq=1
        ss=ss+string(nlpq,format='(i2)')+' files in QUE'
        print,'                    ==>'+message+ss
        if (nlpq le 1) then goto,continue
        if (message ne 'no entries') then goto,pause
continue: lprfiles=0
      end
    set_plot,'sun' & !p.color=0 & !p.background=16777215 & end else wait,15
    if (ftype eq 'l') or ((ftype eq 'f') and (nfile lt nfiles)) then goto,next
;
myexit: if ftype eq 'l' then close,1
return
end
;
;-----------------------------------------------------------------------------
;
pro fix_data
;
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com21,nmax,masklist
;
  ch=' ' & fixtype=' '
  print,'------------------------------------------------------------------'
 print,ch
  read,'Fix clenters, limage, mlask <c,i,m,qiuit> ',fixtype
  if strpos('cim',fixtype) lt 0 then return
;
  if fixtype eq 'i' then fix_image,'i' else $
  if fixtype eq 'c' then fix_beam,ch else $
  if fixtype eq 'm' then fix_mask,ch,0
  if ch eq 'q' then return
;
  ch='x'
  get_beamlets,0
  get_emittance,ch
;
return
end
;
;-----------------------------------------------------------------------------
;
pro rotate_mask,ch
;
common com1,filename,dumpname,printname,dataname,title
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com15,version,comhead,comdata,moments
;
  if ch eq 'f' then begin
    print,'Mask rotation of '+comdata(19)+' removed.'
    ang=-mask(0,1)
  end else begin ;  ch2 eq 'k'
    if nmask lt 1 then begin & print,'First generate the mask' & return & end
    print,'Enter change in mask rotation angle (deg) '+$
                                  string(mask(0,1),format='(f8.3)')+' ==> '
    ss=get_word(ssin) & ang=float(ss)
  end
;
  mask(0,1)=mask(0,1)+ang & comdata(19)=string(mask(0,1),format='(f8.3)')
    pi=!pi & ang=ang*pi/180.0 & sn=sin(ang) & cs=cos(ang)
    for k=1,nmask do begin
      x=mask(k,0) & y=mask(k,1) & mask(k,0)=x*cs-y*sn & mask(k,1)=x*sn+y*cs
    end
```

```
  if ch eq 'k' then get_emittance,' '
;
return
end
;
;-----------------------------------------------------------------------
;
pro fix_mask,ctype,newbeam
;
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,lmin,lmax,background
common com15,version,comhead,comdata,moments
common com21,nmax,masklist
;
  list='' & ch='s'
; must un-rotate mask before fixing.
  if mask(0,1) ne 0 then rotate_mask,'f'
  if typemask eq 2 then gen_round,ch $ ; <----- show rings and holes in use
    else gen_rectangular,ch
;
  num=intarr(nbeam)
  if newbeam eq 0 then begin
    print,'Fix mask'
    print,'   a) enter list of beamlets to be re-associated with mask:'
    print,'   b) enter range to be shift e.g. '23-32 -3' shifts 23-32 left 3:'
    print,'-----------------------------------------------------------------'
    read,list
    if strpos(list,'-') gt 0 then begin ; range of beamlets to shift left-right
      dolist=0
      m=strpos(list,'-') & blk=strpos(list,' ')
      ns=fix(strmid(list,0,m))
      n2=fix(strmid(list,m+1,blk-m))
      nshft=fix(get_word(list,1))
    end else begin
      dolist=1 & decode,list,num,nn
    end
  end else begin              ; call from fix_beam, add newbeam mask
    dolist=1 & nn=newbeam
    for k=0,nn-1 do num(k)=nbeam-k
  end
;
  xx=fltarr(5) & yy=xx & dx=0.1*separation & dy=dx
  xx(0)=-dx & xx(1)=dx & xx(2)=dx & xx(3)=-dx & xx(4)=-dx
  yy(0)= dy & yy(1)=dy & yy(2)=-dy & yy(3)=-dy & yy(4)=dy
;
  if dolist eq 1 then goto,dolist
;
  for k=n2,ns do begin
      oplot,mask(k,0)+xx,mask(k,1)+yy,color=16777214
      mask(k,0)=mask(k,0)+nshft*separation
      mask(k,2)=sqrt(mask(k,0)^2+mask(k,1)^2)
      oplot,mask(k,0)+xx,mask(k,1)+yy
      xyouts,mask(k,0)-dx/2,mask(k,1)-dy/2,string(k,format='(i3)')
  end
  return
;
dolist:
  for k=0,nn-1 do begin
    i=num(k)
    read,'Enter mask hole number for beamlet '+string(i,format='(i3)')+' -->',j
    oplot,mask(i,0)+xx,mask(i,1)+yy,color=16777214
    mask(i,0)=masklist(j,0)
    mask(i,1)=masklist(j,1)
    mask(i,2)=masklist(j,2)
    oplot,mask(i,0)+xx,mask(i,1)+yy
    xyouts,mask(i,0)-dx/2,mask(i,1)-dy/2,string(i,format='(i3)')
  end
;
  ctype='q'
return
```



```
end
;
;-----------------------------------------------------------------------
;
pro fix_beam,chi
;
common com2,xdim,ydim,lmswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,lmin,lmax,background
common com11,sub_range
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
common com21,nmax,masklist
;
  ch=' '
  if chi ne '1' then begin
    read,'A)dd, D)elete beamlets <a,d,q)uits> ',ch
    if strpos('ad',ch) lt 0 then begin chi='q' & return & end
    chi=ch
  end
;
  if chi eq 'a' or chi eq '1' then begin ;       add beamlets from contour plot
+
    chi=' ' & ch2='s' & ss=' ' & ptype(1)=1 & ntime=73
    if ch eq 'a' then $
          get_scale,'*' ; set contour scales to significant region of image
    doplots,chi,ss,ntime  ; plot contour of image
    ptype(1)=0            ; suppress contour plot on return to main routine
    tvcrs,150,150         ; move cursor into image window
    print,'Click LEFT mouse button on beamlet to add, RIGHT quits.'
;
    xi=0 & yi=0 & nadd=0
    scloc=sub_range.scloc
    ix0=fix(strmid(scloc,1,3)) & iy0=fix(strmid(scloc,9,3))
    while ierr ne 4 do begin
      cursor,x,y,/data
      if x ne xi and y ne yi then begin
        xi=x & yi=y & ix=ix0+fix(x/xcmplx) & iy=iy0+fix(y/ycmplx)
        nadd=nadd+1
        nbeam=nbeam+1
;
;       adjust ix,iy so that it is at the max value within this beamlet
        ixmin=ix-5 & ixmax=ix+5 & iymin=iy-5 & iymax=iy+5
        jmax=max(image(ixmin:ixmax,iymin:iymax))
        icc=where(image(ixmin:ixmax,iymin:iymax) eq jmax)
        ic=icc(0)
        nx=ixmax-ixmin+1
        iyc=ic/nx
        ixc=ic-iyc*nx
        iyc=iymin+iyc
        ixc=ixmin+ixc
;
        center(nbeam,0)=ixc & center(nbeam,1)=iyc
        center(nbeam,2)=ixc-xwindow & center(nbeam,3)=ixc+xwindow
        center(nbeam,4)=iyc-ywindow & center(nbeam,5)=iyc+ywindow
        plot_lets(nbeam,1)=1
        print,'Adding beamlet '+string(nbeam,format='(i3)')+$
          ' ixc,iyc='+string(ixc,format='(i3)')+','+string(iyc,format='(i3)')
;
        k=nbeam
        iobject(center(k,2):center(k,3),center(k,4):center(k,5))=0
        mask(nbeam,0)=mask(nmask,0)
        mask(nbeam,1)=mask(nmask,1)
        mask(nbeam,2)=mask(nmask,2)
        nmask=nmask+1
      end ; if x,y point
    end ; while
    adjust_centers
    ssin='t2x' & plot_tv_frame,0 & wset,0 & ssin=' '
```

```
    ii=strpos(comdata(i),' beamlets')-3 & il=strlen(comdata(i))
    comdata(i)=strmid(comdata(i),0,ii)+$
                string(nbeam,format='(i3)')+strmid(comdata(i),ii+3,il-ii)
    if chl eq 'i' then return
    fix_mask,'a',nadd
;
  end else begin ;                          delete beamlets
    ntime=0 & chw=' ' & ss=' '
    plot_window,chw,ntime
    print,'Enter list of beamlet numbers to delete.'
    read,': ',ss
    if strlen(ss) eq 0 or ss eq 'q' then return
    num=intarr(nbeam)
    decode,ss,num,nn
;                                           order list in decending numerical value
    nk=nn
    for k=0,nk-1 do $
      if num(k) lt 1 or num(k) gt nbeam then begin
        nn=nn-1 & for j=k,nn-1 do num(j)=num(j+1)
      end
;
    for k=0,nn-1 do $
      for j=k+1,nn-1 do $
        if num(j) lt num(k) then begin
          isn=num(k) & num(k)=num(j) & num(j)=isn
        end
;
    for k=0,nn-1 do begin
      ib=num(k)
      if ib lt nbeam then begin
        for j=ib,nbeam-1 do center(j,*)=center(j+1,*)
        for j=ib,nbeam-1 do mask(j,*)=mask(j+1,*)
      end
      print,'Deleting beamlet '+string(ib,format='(i3)')+'    ',nbeam,nmask
      nbeam=nbeam-1
      nmask=nmask-1
    end
    ii=strpos(comdata(i),' beamlets')-3 & il=strlen(comdata(i))
    comdata(i)=strmid(comdata(i),0,ii)+$
                string(nbeam,format='(i3)')+strmid(comdata(i),ii+3,il-ii)
  end
;
return
end
;
;-----------------------------------------------------------------------
;
pro fix_image,csw
;
common com2,xdim,ydim,lmswch,image,iobject
common com3,ssin,pview,ch2
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com15,version,comhead,comdata,moments
;
; csw='b' called from main 'Ba' and smooth_data for background subtraction
; csw='E' called from main 'E' for image enhansement
; csw='l' called from fix??_data for removeal of dead pixels
; csw='r' called from readfile to set imin,imax and background
; csw='a' alternate pixels are zero, set equal to ave of neighbors
;
  ch=' ' & ss=''
  if csw eq 'a' then begin
    evens=0 & odds=0
    for k=0,ydim-2,2 do evens=evens+image(xdim/2,k)
    for k=1,ydim-2,2 do odds=odds+image(xdim/2,k)
    print,'Sum of evens = ',evens,' pixels at x=',xdim/2
    print,'Sum of odds  = ',odds,' pixels at x=',xdim/2
    read,'Fix R)eturn,E)ven,O)dd vert.pixels (Chose small value) <e,o,r> ',ss
      if (ss eq 'r') or (strlen(ss) lt 1) then return
      if ss eq 'e' then k0=2 else k0=1
      for k=k0,ydim-2,2 do image(*,k)=(image(*,k-1)/2)+(image(*,k+1)/2)
    end else $
  if csw eq 'b' then begin
```



```
    if (ch2 eq 'a') and (min(image) gt 0) then $
      begin & image=image-min(image) & wset,2 & tv,image & wset,0 & end
        get_beamlets,0 & get_emittance,' '
    end else $
  if (csw eq 'e') or (csw eq 'E') then begin
    imin=min(image) & imax=max(image)
    if imax ge 254 then begin & print,'Image already expanded.' & return & end
      scale=254.0/(imax-imin)
        if csw eq 'E' then begin
        read,'This will scale image(*) by '+string(scale,format='(f6.3)')+$
          ' for range of 0..254 pixels, continue <y,n> ',ch
        if ch ne 'y' then return
        end
        image=byte(scale*(image-imin))
        if csw eq 'E' then begin & get_beamlets,0 & get_emittance,' ' & end
      wset,2 & tv,image & wset,0
    end else $
  if csw eq 'i' then begin & ch=' ' & imx=''
    print,'-----------------------------------------------------------------'
    print,'If image has dead pixels, then image(*)=0 and we can not subtract'
    print,'background, this fix sets image(*)=imin so that the "Ba" works'
    print,'You may also set the image floor to zero, i.e., image(*)<floor=0'
    print,'and image(*)>0=image(*)-floor'
    read,'Fix D)ead pixels, Z)ero_floor  <d,z,q)uits> ==> ',ch
    if ch eq 'q' then return
    read,'Enter level, LF uses ('+string(imin,format='(i3)')+') ==> ',imx
    if strlen(imx) lt 1 then imx=imin & imx=byte(fix(imx))
    if ch eq 'd' then begin
      index=where(image lt imx,count)
        if count gt 0 then image(index)=imx
    end
    if ch eq 'z' then begin
      index=where(image lt imx,count)
      print,'Number of locations to be zeroed is '+string(count)
      if count gt 0 then begin
        image(index)=0 & index=where(image gt 0)
          image(index)=image(index)-imx
      end else image=image-imx
    end
  end
;
  backx=total(image(*,0))/xdim+total(image(*,ydim-1))/xdim
  backy=total(image(0,*))/ydim+total(image(xdim-1,*))/ydim
  background=fix((backx+backy)/4.0)
  imin=min(image) & imax=max(image)
  comdata(2)='['+string(imin,format='(i3)')+','+$
              string(imax,format='(i3)')+'] '+string(background,format='(i3)')
  print,comhead(2)+comdata(2)
;
return
end
;
;-----------------------------------------------------------------------
;
pro plot_tv_frame,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,lmswch,image,iobject
common com3,ssin,pview,ch2
common com5,plot_lets,center,maxb,nbeam
common com8,window,xwindow,ywindow,chwin2,level
common com15,version,comhead,comdata,moments
;
  ch1=' '
  if ntime eq 0 then begin
    cc=0 & ch=strmid(ssin,1,1)
    if strlen(ssin) eq 3 then ch1=strmid(ssin,2,1)
    if strpos('024',ch) ge 0 then begin & cc=fix(ch)
      wset,cc & wshow,cc,1
    end
  end
;
  erase
```

```
  if ntime eq 0 then begin
    if ch2 eq 'o' then imswch='i' else $
    if ch2 eq 'h' then imswch='h' else imswch=' '
  end
;
  if imswch eq 'i' then tvscl,iobject,/device else $
  if imswch eq 'h' then begin
    halo=image
    for k=1,nbeam do begin
      nx1=center(k,2) & nx2=center(k,3) & ny1=center(k,4) & ny2=center(k,5)
      bg1=total(halo(nx1:nx2,ny1))/(nx2-nx1+1)
      bg2=total(halo(nx1:nx2,ny2))/(nx2-nx1+1)
      bg3=total(halo(nx1,ny1:ny2))/(ny2-ny1+1)
      bg4=total(halo(nx2,ny1:ny2))/(ny2-ny1+1)
      bg=0.25*(bg1+bg2+bg3+bg4)
      halo(nx1:nx2,ny1:ny2)=bg
    end
    tvscl,halo,/device
  end else tv,image,/device
;
  ss='' & n=strlen(comdata(1))-10
  if n gt 1 then ss='                    ('+strmid(comdata(1),10,n)+')'
  xyouts,0.1,0.95,filename,/norm
  xyouts,0.1,-0.07,filename+ss,/norm
  xyouts,0.1,-0.10,title,/norm
  if ch1 ne ' ' then $
    for k=1,nbeam do begin
      lx=center(k,2)-5
      ly=(center(k,4)+center(k,5))/2
      xyouts,lx,ly,string(k,format='(i2)'),color=16777215,/device
    end
;
return
end
;
;-----------------------------------------------------------------------------
;
pro output,have_done,chx
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
;
  ss='' & ch=' ' & maxl=45
  if have_done eq 't' then goto,printit
  openw,1,printname, /variable
  print,'Writing beam and window data'
  ss=strcompress(strtrim(title,2)) & len=strlen(ss)-1
  k=0 & km=72
  repeat begin
    if k+km gt len then km=len-k
      printf,1,strmid(ss,k,km)
    k=k+72
  endrep until k ge len
  printf,1,' '
  for k=0,14 do printf,1,string(k,format='(i2)')+' '+comhead(k)+comdata(k)
  k=19 & printf,1,string(k,format='(i2)')+' '+comhead(k)+comdata(k)
  printf,1,strcompress(filename)+' ==> '+strcompress(printname)
  k=0
  print,'  @CENTERS(*)'
  printf,1,'  @CENTERS(*)'
  printf,1,$
'    n xpix ypix    n xpix ypix     n xpix ypix    n xpix ypix    n xpix ypix'
  repeat begin & ss=''
      for n=1,5 do begin
        k=k+1
        if k gt nbeam then goto,done
          ss=ss+string(k,format='(i6)')+$
```



```
          string(center(k,0),format='(i4)')+string(center(k,1),format='(i4)')
      end
    done: printf,1,ss
  endrep until k ge nbeam
  printf,1,'Beamlet centers and x,y range (pixels) center(1..nbeam,0..5)'
  printf,1,'    k    xcen ycen xmin xmax ymin ymax'+$
  printf,1,'    k    xcen ycen xmin xmax ymin ymax'
  nbh=nbeam/2
  for k=1,nbh do begin
    if plot_lets(k,1) eq 1 then ksw=k else ksw=-k
    if plot_lets(k+nbh,1) eq 1 then jsw=k+nbh else jsw=-k-nbh
    printf,1,format='(i4,1x,3(2x,2i4),2x,i4,1x,3(2x,2i4))',$
          ksw,center(k,0),center(k,1),$
              center(k,2),center(k,3),center(k,4),center(k,5),$
          jsw,center(k+nbh,0),center(k+nbh,1),$
              center(k+nbh,2),center(k+nbh,3),center(k+nbh,4),center(k+nbh,5)
  end
; ---------
  print,'  @DATA(*,)'
  printf,1,'  @DATA(*,)'
  printf,1,' n  peak   xmin   xmax   xcent   ymin   ymax   ycent   '+$
            'total_int  backg'
  printf,1,'  pixels   (cm)   (cm)    (cm)    (cm)    (cm)    (cm)   '+$
            '  pixels  pixels'
  for k=1,nbeam do begin
    ss=string(data(k,0),format='(i5)')+' '
      for j=1,6 do ss=ss+string(data(k,j),format='(f7.3)')
        ss=ss+' '+string(data(k,7),format='(f10.0)')+$
            ' '+string(data(k,8),format='(f6.2)')
          printf,1,string(k,format='(i3)')+' '+ss
    end
  print,'  @PHASE(*,)'
  printf,1,'  @PHASE(*,) at location of image'
  printf,1,'  n   xg-ave    xpmin     xpmax     yg-ave    ypmin     ypmax'
  for k=1,nbeam do begin
    ss='' & for j=0,5 do ss=ss+string(phase(k,j),format='(f9.5)')
      printf,1,string(k,format='(i3)')+' '+ss
    end
    printf,1,'.................'
    ss='' & for j=0,1 do ss=ss+string(phase(maxdata-2,j),format='(f9.5)')
     for j=2,5 do ss=ss+string(phase(maxdata-3,j),format='(f9.5)')
      printf,1,string(maxdata-3,format='(i3)')+' '+ss
    ss='' & for j=0,5 do ss=ss+string(phase(maxdata-1,j),format='(f9.5)')
      printf,1,string(maxdata-1,format='(i3)')+' '+ss
;???
  print,'  @EMIT_BRIGHTNESS'+'  nspcj='+string(nspcj)
  printf,1,'  @EMIT_BRIGHTNESS'+'  nspcj='+string(nspcj)
  printf,1,$
    '  k   Emitx      Emity      Radius     Current  Brightness   SPC_Corrected'
  printf,1,$
    '       m-rad      m-rad       cm       Amperes  A/(m-rad)^2  A/(m-rad)^2'
  for k=1,nbeam do begin
    ss=''
      ss=string(k,format='(i3)')+$
         string(emitn(k,0),format='(e10.3)')+$
         string(emitn(k,1),format='(e10.3)')+$
         string(jradius(k-1),format='(f9.4)')+$
         string(ibeamn(k-1),format='(f9.4)')+$
         string(bjj(k-1,0),format='(e15.3)')+$
         string(bjj(k-1,1),format='(e15.3)')
      printf,1,ss
    end
  print,'  @MASK'
  printf,1,'  @MASK        nmask=',nmask,' typemask=',typemask
  printf,1,'  k        x(cm)       y(cm)'
  for k=1,nmask do $
    printf,1,string(k,format='(i4)')+$
       string(mask(k,0),format='(f9.4)')+string(mask(k,1),format='(f9.4)')
  printf,1,'------'
  print,'FILE '+printname+' written and locked.'
  close,1
  have_done='t'
printit:
```

```
  if chx eq 'E' then return
  openr,1,printname & lines=0
    while not eof(1) do $
      begin & readf,1,ss & print,ss
        if strpos(ss,'0') ge 0 then begin
          read,'Continue < ,n> ',ch
            if ch eq 'n' then goto,close & lines=0 & end
        lines=lines+1 & if lines ge maxl then begin & lines=0
          read,'Continue < ,n> ',ch & if ch eq 'n' then goto,close & end
      end
close: close,1
return
end
;
;-----------------------------------------------------------------------
;
pro decode,ss,num,n
; input string ss, return num(0,...)=numbers, n entries
;   ss='125 160 240 275 ' then num(125,160,240,275) and n=4
;
  ns=size(num) & kmax=ns(1) & temp=''
  n=0 & sn='' & nblk=0 & nstart=0 & numbers='0123456789+-.e'
  if strpos(ss,'\') lt 0 then temp=ss else temp=strmid(ss,0,strpos(ss,'\')-1)
  sslen=strlen(temp) & if sslen eq 0 then return
  temp=strcompress(strtrim(temp,2))+' '
    for k=1,kmax do begin
      if nstart ge sslen then return
      nblk=strpos(temp,' ',nstart)
        sn=strmid(temp,nstart,nblk-nstart) & nstart=nblk+1
          for j=0,strlen(sn)-1 do $
            if strpos(numbers,strmid(sn,j,1)) eq -1 then goto,skipit
             if strpos(sn,'.') ge 0 then num(n)=float(sn) else num(n)=fix(sn)
        n=n+1
      skipit:
    end
return
end
;
;-----------------------------------------------------------------------
;
pro readfile,ioerror,restart
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com15,version,comhead,comdata,moments
;
  if ioerror eq 'F' then begin & ss='' & kmax=0 & goto,skip & end
  path=comdata(15)
  on_ioerror,error
  openr,unit,path+filename,/GET_LUN
    hdr=assoc(unit,bytarr(512)) & h=hdr(0)
      xdim=h(4)+256*h(5)
      ydim=h(6)+256*h(7)
print,'xdim=',xdim,'ydim=',ydim
      npic=float(xdim)*float(ydim)
      block=fix(npic/512.0)
      nstart=fix(npic-512.0*block)
    aa=assoc(unit,bytarr(xdim,ydim),512.0) & image=aa(0)
    ;  tail=assoc(unit,bytarr(18+nstart),512.0*(1+block)) & t=tail(0)
  goto,done
  error: print,'IO ERROR ',!err_string & ioerror='t' & return
;
done: free_lun,unit
  title='' & ss=''
  kmax=256 & while h(kmax) eq 32 do kmax=kmax-1
  if kmax-11 lt 80 then for k=10,kmax do ss=ss+string(h(k)) else $
  begin
    for k=10,25 do ss=ss+string(h(k))          ; first 20 characters
    for k=kmax-57,kmax do ss=ss+string(h(k))   ; last 60 characters
  end
skip: title=strmid(ss,0,kmax)
  if ioerror eq 'A' then return ; called from autoview
;
  comdata(1)='['+string(xdim,format='(i3)')+','+string(ydim,format='(i3)')+']'
```



```
;
; even odd error fix
; ------------------
  imin=min(image) & imax=max(image) & scale=254.0/(imax-imin)
  image=byte(scale*(image-imin))
  for k=1,ydim-2,2 do image(*,k)=(image(*,k-1)/2)+(image(*,k+1)/2)
;
; boarder reflections
; -------------------
  for k=8,0,-1 do begin
    image(k,*)=image(k+1,*)
    image(xdim-k-1,*)=image(xdim-k-2,*)
    image(*,k)=image(*,k+1)
    image(*,ydim-k-1)=image(*,ydim-k-2)
  end
;
  fix_image,'r' ; sets imin,imax, and background
;
  comdata(17)=title
  print,'IMAGE'+comdata(1)
  read,'Smooth image with sliding *box car*                <y,n> ==> ',ss
  print,'-----------------------------------------------------------------'
  ch=' '
  if ss eq 'y' then smooth_data,ch
  wset,2 & tv,image & wset,0
  get_centers,restart
  tvscl,iobject
;
return
end
;
;-----------------------------------------------------------------------
;
pro get_rebin,new_image,cx,cy ; Finds index for rebin (512/8,480/8)(510/6,478/2
+)
; called from autoview and doplot
;
common com2,xdim,ydim,imswch,image,iobject
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
;
  nx=10 & ny=nx & nml=nx-1
  nx=9 & ny=nx & nml=nx-1
  xr=float(xdim) & yr=float(ydim)
  for k=1,nml do begin
    rem=xr-fix(xr/nx)*nx & if (rem eq 0) then goto,next else nx=nx-1 & end
next:
  for k=1,nml do begin
    rem=yr-fix(yr/ny)*ny & if (rem eq 0) then goto,done else ny=ny-1 & end
done:
  nbx=xdim/nx & nby=ydim/ny
  new_image=rebin(image,nbx,nby)
  cx=nx*xcmplx*indgen(xdim/nx)-xcmplx*xdim/2
  cy=ny*ycmplx*indgen(ydim/ny)-ycmplx*ydim/2
;
return
end
;
;-----------------------------------------------------------------------
;
pro get_beamlets,iabort
;
common com2,xdim,ydim,imswch,image,iobject
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,lpass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
;
ch=' '
  itotal=0 & xmin=0 & xmax=0 & ymin=0 & ymax=0 & pmax=0 & sum=0 & nd=0 & back=0
;
```

```
  for beamlet=1,nbeam do begin
    err=0
    nx=center(beamlet,0) & ny=center(beamlet,1)
    if (nx lt xwindow) or (ny lt ywindow) or $
       (xdim-nx lt xwindow) or (ydim-ny lt ywindow) then err=1
    if (err ne 0) then goto,bypass
;
      nx1=nx-xwindow & if nx1 lt 1 then nx1=1
      nx2=nx+xwindow & if nx2 ge xdim then nx2=xdim-1
      ny1=ny-ywindow & if ny1 lt 1 then ny1=1
      ny2=ny+ywindow & if ny2 ge ydim then ny2=ydim-1
      back=(total(image(nx1:nx2,ny1))+total(image(nx1:nx2,ny2)))/(nx2-nx1+1)+$
           (total(image(nx1,ny1:ny2))+total(image(nx2,ny1:ny2)))/(ny2-ny1+1)
      back=0.25*back
;
  ;   TEST 4/21/97 subtract tilted local background, then calculate cut
;
      linex=image(nx1:nx2,ny) & maxline=nx2-nx1 & xwinmax=maxline-1
      ixa=where(linex eq max(linex)) & ix=ixa(0)
      i1x=float(min(linex(0:ix)))
      i2x=float(min(linex(ix:maxline)))
      ix1a=where(linex(0:ix) eq i1x)
      ix2a=where(linex(ix:maxline) eq i2x)
      ix1=float(ix1a(0)) & ix2=ix+float(ix2a(0))
      if ix2 eq ix1 then xas=0 else xas=(i2x-i1x)/(ix2-ix1)
      xback=i1x+xas*(indgen(nx2-nx1))
      linex=linex-xback
      cutx=fix(0.01*level*max(linex)+0.5)
      for k=ix,1,-1 do if linex(k) lt cutx then goto,gotx1
    gotx1: if k>1 ge maxline then goto,bypass
      if k lt 0 then k=0
      if (linex(k+1)-linex(k) eq 0) then xmin=(k-ix)*xcmpix else $
      xmin=(k-ix+(cutx-linex(k))/(linex(k+1)-linex(k)))*xcmpix
      for k=ix,xwinmax do if linex(k) lt cutx then goto,gotx2
      k=k-1
    gotx2: if (linex(k-1)-linex(k) eq 0) then xmax=(k-ix)*xcmpix else $
      xmax=(k-ix-(cutx-linex(k))/(linex(k-1)-linex(k)))*xcmpix
;
      liney=image(nx,ny1:ny2) & maxline=ny2-ny1 & ywinmax=maxline-1
      iya=where(liney eq max(liney)) & iy=iya(0)
      i1y=float(min(liney(0:iy)))
      i2y=float(min(liney(iy:maxline)))
      iy1a=where(liney(0:iy) eq i1y)
      iy2a=where(liney(iy:maxline) eq i2y)
      iy1=float(iy1a(0)) & iy2=iy+float(iy2a(0))
      if iy1 eq iy2 then yas=0 else yas=(i2y-i1y)/(iy2-iy1)
      yback=i1y+yas*(indgen(ny2-ny1))
      liney=liney-yback
;
      cuty=fix(0.01*level*max(liney)+0.5)
      for k=iy,1,-1 do if liney(k) lt cuty then goto,goty1
    goty1: if k>1 ge maxline then goto,bypass
      if k lt 0 then k=0
      if (liney(k+1)-liney(k) eq 0) then ymin=(k-iy)*ycmpix else $
      ymin=(k-iy+(cuty-liney(k))/(liney(k+1)-liney(k)))*ycmpix
      for k=iy,ywinmax do if liney(k) lt cuty then goto,goty2
      k=k-1
    goty2: if (liney(k-1)-liney(k) eq 0) then ymax=(k-iy)*ycmpix else $
      ymax=(k-iy-(cuty-liney(k))/(liney(k-1)-liney(k)))*ycmpix
;
    peak=max(image(nx1:nx2,ny1:ny2)) & if peak gt pmax then pmax=peak
      itotal=total(image(nx1:nx2,ny1:ny2))-back*(nx2-nx1+1)*(ny2-ny1+1)
        sum=sum+itotal
;
bypass:
      nd=nd+1
      data(nd,0)=image(nx,ny)       ; peak amplitude
      data(nd,1)=xmin & data(nd,2)=xmax & data(nd,3)=nx*xcmpix
      data(nd,4)=ymin & data(nd,5)=ymax & data(nd,6)=ny*ycmpix
      data(nd,7)=itotal             ; total int above background for beamlett.
      data(nd,8)=back
      for j=1,6 do if data(nd,j) gt 50.0 then err=1
        if err eq 1 then begin & plot_lets(nd,1)=0
```



```
          data(nd,1)=0 & data(nd,2)=0 & data(nd,4)=0 & data(nd,5)=0 & end
      if beamlet eq iabort then abort ; TEST debug ACP
    end
    data(0,0)=pmax & data(0,7)=sum
;
  xave=total(data(1:nbeam,3))/nbeam
  yave=total(data(1:nbeam,6))/nbeam
;
for k=1,nbeam do $
     bradius(k-1)=0.25*sqrt((data(k,2)-data(k,1))^2+(data(k,5)-data(k,4))^2)
;
return
end
;
;-----------------------------------------------------------------------
;
pro get_sigma,xx1,xx2,yy1,yy2
;
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com11,lpass,bradius,xave,yave,ibeamn,jradius,bjj
common com14,s11,s12,s22,s33,s34,s44,xcn,ycn,xpcn,ypcn,r12,r34,lx,ly,ld,ldx,ldy
common com15,version,comhead,comdata,moments
;
  xx1=fltarr(nbeam+1) & xx2=xx1 & yy1=xx1 & yy2=xx1
  count=0 & xcn=0 & ycn=0 & xpcn=0 & ypcn=0 & xemitav=0 & yemitav=0
  avgbeamlet=0
  for k=1,nbeam do if plot_lets(k,1) eq 1 then begin
    count=count+1
    xemitav=xemitav+emitn(k,0)
    yemitav=yemitav+emitn(k,1)
  end
  beamave=data(0,7)/nbeam
  xemitav=1e6*xemitav/(nbeam*beamave)
  yemitav=1e6*yemitav/(nbeam*beamave)
;-----------------------------------------------------------------
  for k=1,nbeam do if plot_lets(k,1) eq 1 then begin
    xcn=xcn+mask(k,0)*data(k,7)/beamave
    xpcn=xpcn+0.5*(phase(k,1)+phase(k,2))*data(k,7)/beamave
    ycn=ycn+mask(k,1)*data(k,7)/beamave
    ypcn=ypcn+0.5*(phase(k,4)+phase(k,5))*data(k,7)/beamave
;-----------------------------------------------------------------
  end
;
  if count eq 0 then return
  ni=1.0/count
  xc=xcn*ni & yc=ycn*ni & xpc=xpcn*ni & ypc=ypcn*ni
;
print,'xc,xpc  ',xc,xpc
print,'yc,ypc  ',yc,ypc
;
  s11=0 & s22=0 & s33=0 & s44=0 & s12=0 & s34=0
  s13=0 & s23=0 & s14=0 & s24=0
  a=0 & b=0 & ap=0 & bp=0 & x11=0 & x22=0 & y33=0 & y44=0
;
  for k=1,nbeam do if plot_lets(k,1) eq 1 then begin
     x=0.01*(mask(k,0)-xc) & xp1=phase(k,1)-xpc & xp2=phase(k,2)-xpc
     y=0.01*(mask(k,1)-yc) & yp1=phase(k,4)-ypc & yp2=phase(k,5)-ypc
;
    x1=x+ldx*xp1      & x2=x+ldx*xp2   & xx1(k)=x1 & xx2(k)=x2
    y1=y+ldy*yp1      & y2=y+ldy*yp2   & yy1(k)=y1 & yy2(k)=y2
;-----------------------------------------------------------------
      s11=s11+(x1^2+x2^2)*data(k,7)/beamave
      s12=s12+(x1*xp1+x2*xp2)*data(k,7)/beamave
      s22=s22+(xp1^2+xp2^2)*data(k,7)/beamave
      s33=s33+(y1^2+y2^2)*data(k,7)/beamave
      s34=s34+(y1*yp1+y2*yp2)*data(k,7)/beamave
      s44=s44+(yp1^2+yp2^2)*data(k,7)/beamave
;  cross moments
      s13=s13+(x1*y1+x2*y2)*data(k,7)/beamave
      s23=s23+(xp1*y1+xp2*y2)*data(k,7)/beamave
      s14=s14+(x1*yp1+x2*yp2)*data(k,7)/beamave
```

```
    s24=s24+(xp1*yp1+xp2*yp2)*data(k,7)/beamave
;-----------------------------------------------------------
;
  end
;
xcn=xcn*ni & ycn=ycn*ni
xpcn=1000.0*xpcn*ni & ypcn=1000.0*ypcn*ni
;xxpave=xxpave*ni & yypave=yypave*ni
;

;       .-- meters^2 .-- rad^2     .-- meter*rad        .-- mm-mr
  s11=1.0e6*s11*ni & s22=1.0e6*s22*ni & s12=1.0e6*s12*ni
  emitx=sqrt(s11*s22-(s12)^2)
  s33=1.0e6*s33*ni & s44=1.0e6*s44*ni & s34=1.0e6*s34*ni
  emity=sqrt(s33*s44-(s34)^2)
  s13=1.0e6*s13*ni & s23=1.0e6*s23*ni & s14=1.0e6*s14*ni & s24=1.0e6*s24*ni
print,'s11,s12,s22',s11,s12,s22
print,'s33,s34,s44',s33,s34,s44
;
  a=2*sqrt(1.0e-6*s11/2) & b=2*sqrt(1.0e-6*s33/2)
  ap=2.0e-6*s12/2/(sqrt(1.0e-6*s11/2)) & bp=2.0e-6*s34/2/(sqrt(1.0e-6*s33/2))
  tx=0.5*atan(2*s12/(s11-s22))*180/3.14159
  ty=0.5*atan(2*s34/(s33-s44))*180/3.14159
  print,'a=',a,'  b=',b,'  ap=',ap,'  bp=',bp
  print,'tx=',tx,'  ty=',ty
moments(0)=s11/2 & moments(1)=s12/2 & moments(2)=s22/2
moments(3)=s33/2 & moments(4)=s34/2 & moments(5)=s44/2
moments(6)=s13/2 & moments(7)=s23/2 & moments(8)=s14/2 & moments(9)=s24/2
print,'x2,xxp,xp2',moments(0),moments(1),moments(3)
print,'y2,yyp,yp2',moments(3),moments(4),moments(5)
print,'xy,xpy,xyp,xpyp',moments(6),moments(7),moments(8),moments(9)
;-----------------------------------------------------------------------
+-
;  moments are divided by a factor of 4 to get the 'expectation values' from
;  the sigma matrix values (emittance ellipse coefficients).  The sigma values
;  differ from the 'expectation values' because they are summed in pairs to
;  accomodate the array indices but only divided by the number of beamlets whic
+h
;  is one-half the number of data points summed giving 1 factor of 2.  The othe
+r
;  factor of 2 comes from the sigma value calculation with the 2/N in front
;  instead of 1/N which gives the other factor of 2.
;-----------------------------------------------------------------------
+-
print,'emitx,emity',emitx,emity
print,'angmom=',betagamma*(moments(8)-moments(7))
;
nex=betagamma*emitx & ney=betagamma*emity
print,'norm x emit=',nex,'   norm y emit=',ney
eng=sqrt(0.5*nex^2+0.5*ney^2+16*betagamma^2*(s13*s24-s14*s23))
;
enh=sqrt(256*betagamma^4*(s24^2*s13^2 - 2*s23*s24*s13*s14 + s23^2*s14^2 + 2*s23
+*s12*s13*s44 - s22*s13^2*s44 - s23^2*s11*s44 - 2*s24*s12*s13*s34 - 2*s23*s12*s14
+*s34 + 2*s22*s13*s14*s34 + 2*s23*s24*s11*s34 + 2*s24*s12*s14*s33 - s22*s14^2*s33
+ - s24^2*s11*s33) + 256*betagamma^4*(-s12^2 + s22*s11)*(-s34^2 + s44*s33))
;
print,'eng',eng
print,'enh',enh
  if s11*s22 gt 0 then r12=s12/sqrt(s11*s22) else r12=0
  if s33*s44 gt 0 then r34=s34/sqrt(s33*s44) else r34=0
;
; Distance to waist: lx=-s12/s22  and ly=-s34/s44
;
  s11=1.0e-6*s11 & s22=1.0e-6*s22 & s12=1.0e-6*s12
  emitx=1.0e-6*emitx
  s33=1.0e-6*s33 & s44=1.0e-6*s44 & s34=1.0e-6*s34
  emity=1.0e-6*emity
return
end
;
;-----------------------------------------------------------------------
;
pro get_emittance,chsw
```



```
;
common com2,xdim,ydim,imswch,image,iobject
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com14,s11,s12,s22,s33,s34,s44,xcn,ycn,xpcn,ypcn,r12,r34,lx,ly,ld,ldx,ldy
common com15,version,comhead,comdata,moments
common com17,chbav,have_done,maxptype,ptype,chel1lpse,chel6d,chelscl
;                    0     1     2     3     4     5     6     7     8
; data (    ,*)=pixel,xmin ,xmax ,xave,ymin ,ymax ,yave, sum, backg
; phase(    ,*)=XABS,xpmin,xpmax,YABS,ypmin,ypmax
; phase(m-1,*)=s11, s12, s22, s33, s34, s44    / m=maxdata=400
; phase(m-2,0)=r12, r34, xscl, xpscl, yscl, ypscl
; phase(m-3,*)=0,0, xave, xpave, yave, ypave / center of phase space
; phase(m-4,*)=xmin, xmax, xpmin, xpmax, 0,0  / fix scales of brightness plots
; phase(m-5,*)=xmin, xmax, xpmin, xpmax , 0,0 / fix scales of emittance plots
;
  if data(0,0) gt 0 and chsw eq 'n' then goto,regenerate
  if data(0,0) eq 0 then get_beamlets,0
;
  nemit=0
  zero=0.000001
  phase=fltarr(maxdata,6) ; phase(maxdata-1,6)=s11,s12,s22,s33,s34,s44
  if rhole eq 0 then $
    begin & phase(*,*)=0 & emitn(*,*)=0 & goto,regenerate & end
  if drift gt 0 then dcmi=1/(100*drift) else dcmi=0
;
openw,13,'hemitx.dat'
openw,14,'hemity.dat'
  for k=1,nbeam do begin
    rx=0.5*(data(k,2)-data(k,1))
      if (rx lt rhole) then rp=0 else rp=dcmi*(rx-rhole)
      emitn(k,0)=0.01*mask(k,0)*rp
      printf,13,mask(k,0),1e6*emitn(k,0)
;       emitn(k,0)=0.01*rhole*rp
      phase(k,0)=data(k,3)-xave
      rp0=(phase(k,0)-mask(k,0))*dcmi
      phase(k,1)=rp0-rp & phase(k,2)=rp0+rp
    ry=0.5*(data(k,5)-data(k,4))
      if (ry lt rhole) then rp=0 else rp=dcmi*(ry-rhole)
;       emitn(k,1)=0.01*rhole*rp
      emitn(k,1)=0.01*mask(k,1)*rp
      printf,14,mask(k,1),1e6*emitn(k,0)
      phase(k,3)=data(k,6)-yave
      rp0=(phase(k,3)-mask(k,1))*dcmi
      phase(k,4)=rp0-rp & phase(k,5)=rp0+rp
  end
close,13
close,14
;
emitn(0,2)=nbeam
;
regenerate: pi=!pi  ;pi=3.14159
;
; at mask, ldx=ldy=0
  ldx=0 & ldy=0 & get_sigma,xx1,xx2,yy1,yy2
  lx=-s12/s22
  ly=-s34/s44
  jradius=fltarr(nbeam) & ibeamn=jradius & bjj=fltarr(nbeam,2)
  phase(maxdata-1,0)=s11 & phase(maxdata-1,1)=s12 & phase(maxdata-1,2)=s22
  phase(maxdata-1,3)=s33 & phase(maxdata-1,4)=s34 & phase(maxdata-1,5)=s44
  phase(maxdata-2,0)=r12 & phase(maxdata-2,1)=r34
  phase(maxdata-3,2)=xcn & phase(maxdata-3,3)=xpcn
  phase(maxdata-3,4)=ycn & phase(maxdata-3,5)=ypcn

  ngood=0 & theta=0
  for k=1,nbeam do $
    if (phase(k,1) ne 0) and (phase(k,2) ne 0) and $
       (phase(k,1) ne 0) and (phase(k,2) ne 0) then begin
```

```
          theta=theta+0.25*(phase(k,2)-phase(k,1)+phase(k,5)-phase(k,4))
            ngood=ngood+1
        end
  if ngood gt 0 then theta=1000.0*theta/ngood
  for k=1,nbeam do jradius(k-1)=mask(k,2)+rhole
  if mask(0,2) gt 0 then radius=mask(0,2) else radius=max(jradius)
;
  ave=0.5*(emitx+emity)
  comdata(8)='['+string(emitx,format='(f8.3)')+',',+$
                 string(emity,format='(f8.3)')+']'+$
                 '    ave:'+string(ave,format='(f8.3)')+$
                 '    norm:'+string(ave*betagamma,format='(f9.4)')
  comdata(11)='['+string(100.0*sqrt(s11),format='(f7.4)')+',',+$
                 string(r12,format='(f7.4)')+',',+$
                 string(1000.0*sqrt(s22),format='(f7.2)')+']    ',+$
                 string(1x,format='(f7.3)')+' m'
  comdata(12)='['+string(100.0*sqrt(s33),format='(f7.4)')+',',+$
                 string(r34,format='(f7.4)')+',',+$
                 string(1000.0*sqrt(s44),format='(f7.2)')+']    ',+$
                 string(1y,format='(f7.3)')+' m'
;
; ------------------------
;
  if emitx eq 0 or emity eq 0 then return
  jx=2.0*current/(pi*1e-6*emitx*betagamma)^2
  jy=2.0*current/(pi*1e-6*emity*betagamma)^2
;print,'jx,jy',jx,jy
  jxy=1e12
;  jxy=sqrt(jx*jy)
  if radius*theta eq 0 then jwb=0 else $
    jwb=current/(pi*1e-5*radius*theta*betagamma)^2
  comdata(13)='['+string(jx,format='(e10.2)')+',',+$
                 string(jy,format='(e10.2)')+',',+$
                 string(jxy,format='(e10.2)')+',',+$
                 string(jwb,format='(e10.2)')+']'
;
; calculate brightness  j=in/(pi emit betagamm)^2
;   in=(IBEAM*nbeamlets*hole^2/RBEAM^2)*sum(pix n)/sum(all pix in image)
;
;                 .---- then we have inputted the measured passed current
;                /
  if ipass(1) gt 0 then nbpass=ipass(1) else nbpass=nbeam
  radius=50.0*(sqrt(s11)+sqrt(s33))
  amps=current*nbpass*(rhole/radius)^2 & imask=amps
;
; ----------------------------------------
; TEST 4/28/97 ACP
if chsw ne 'x' then goto,skipit
; print,'TEST:get_emittance  current='+string(current,format='(f7.1)')+$
;        '  nbpass='+string(nbpass,format='(i3)')+$
;        '  r(cm)='+string(radius,format='(f8.3)')+$
;        '  amps='+string(amps,format='(f8.3)')
  ipave=0
    for k=1,nbeam do ipave=ipave+data(k,0) & ipave=ipave/nbeam ; ave peak
    ; incident whole beam pixels = pi*r^2*ipave
    flux=!pi*radius^2*ipave     ; pixels
    iave=current/flux
  iapx=0
    for k=1,nbeam do iapx=iapx+data(k,7) & iapx=iapx/nbeam
  iave=0
  iapb=0
  itaa=0
; print,'     Average incident flux '+string(ipave,format='(f8.1)')
; print,'     Average impingent flux on mask(A/p) '+string(iave,format='(f8.2)'
+)
; print,'     Average pixels per beamlet '+string(iapx,format='(f8.2)')
; print,'     Average current per beamlet '+string(iapb,format='(f8.2)')
; print,'     Total passed current        '+string(itaa,format='(f8.2)')
skipit;
; END TEST 4/28/97 ACP
; ----------------------------------------
;
  if (ipass(0) gt 0) and (ipass(1) gt 0) then begin
```



```
    amps=ipass(0)*nbeam/ipass(1) & imask=ipass(0) & end
;
  amps=amps/data(0,7) ; data(0,7)=sum of all pixcels -- all beamlets total
  as=0
  as=2/!pi & cutoff=0.02*rhole/thickness
  for k=1,nbeam do begin
     rp=sqrt((phase(k,2)-phase(k,1))^2+(phase(k,5)-phase(k,4))^2)
     if rp lt cutoff then f=1.0 else $
     begin        ;     reff^2 = rhole^2 * (1-(2t/pi) -(2/pi)sin(t)cos(t))
       theta1=thickness*tan(rp)/(2.0*rhole)
         f=1.0-as*theta1-as*sin(theta1)*cos(theta1)
     end
     ibeamn(k-1)=amps*f*data(k,7)        ; current carried by this beamlet
     as=as+ibeamn(k-1)
  end
;       .--- diff between imask and as is non waist angle correction effect.
;      /
  if ipass(0) gt 0 then begin ; scale current to measured value
    as=imask/as
      for k=1,nbeam do ibeamn(k-1)=as*ibeamn(k-1)
  end else imask=as
  if mask(0,2) gt 0 then $
    comdata(14)='['+string(mask(0,2),format='(f8.3)') else $
    comdata(14)='['+string(radius,format='(f8.3)')
    comdata(14)=comdata(14)+',',+string(theta,format='(f7.3)')+',',+$
                                  string(imask,format='(f7.3)')+']'
;
  const=(betagamma*pi)^2
  correction=(0.02*pi*rhole)^2/(17000*betagamma)
  for k=1,nbeam do begin
    if (emitn(k,0) eq 0) or (emitn(k,1) eq 0) then jn=0 else $
      jn=ibeamn(k-1)/(const*emitn(k,0)*emitn(k,1))
     bjj(k-1,0)=jn & rb=bradius(k-1)
      if rb gt 0 then jn=jn*(1-(rhole/rb)^2)/(1-jn*correction*alog(rb/rhole))
     jradius(k-1)=mask(k,2)+rhole
     bjj(k-1,1)=jn
  end
  brightness=0 & nb=0
    for k=0,nbeam-1 do $
      if (bjj(k,nspcj) gt 0) then $
        if plot_lets(k,1) eq 1 then $
          begin & nb=nb+1 & brightness=brightness+bjj(k,nspcj) & end
  if nb gt 0 then brightness=brightness/nb
  sdj=0
    for k=0,nbeam-1 do $
      if (bjj(k,nspcj) gt 0) then if plot_lets(k,1) eq 1 then $
        sdj=sdj+(bjj(k,nspcj)-brightness)^2
          sdj=sqrt(sdj)/(nb-1)
  comdata(9)='['+string(current,format='(f8.2)')+'('+$
                 string(imask,format='(f6.3)')+'/'+$
                 string(nbpass,format='(i4)')+'),'+$
                 string(brightness,format='(e10.2)')+'  +-'+$
                 string(sdj,format='(e8.2)')+']'
return
end
;
;-----------------------------------------------------------------------------
;
pro movie,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com15,version,comhead,comdata,moments
common com18,kmovie,nslice,slices
;
  add=0 & kmin=5 & kmax=0 & kstep=5 & ss='' & smw=0 & num=intarr(4)
  if ntime eq 0 then begin
    itimex=1
    if (strlen(ssin) eq 3) then if (strmid(ssin,2,1) eq 'a') then add=1
    if ch2 eq 'a' then add=1
    print,'Enter kmin,kmax,kstep,smooth (LF uses 5,480,5,0) ' & read,ss
```

```
    if strlen(ss) gt 1 then begin & decode,ss,num,n
      kmin=fix(num(0)) & kmax=fix(num(1)) & kstep=fix(num(2))
      if n ge 4 then smw=fix(num(3))
      if n ge 5 then ltimex=fix(num(4))
    end
    kmovie(0)=kmin & kmovie(1)=kmax & kmovie(2)=kstep & kmovie(3)=smw
    if ch2 eq 'x' then kmovie(4)=1 else kmovie(4)=3
  end else begin
    kmin=kmovie(0) & kmax=kmovie(1) & kstep=kmovie(2) & smw=kmovie(3)
    if kmovie(4) eq 1 then ch2='x' else ch2='y'
  end
;
  heading=filename
  n=strlen(comdata(1))-10
  if n gt 1 then heading=heading+'  '+strmid(comdata(1),10,n)
  smhead='Cut smoothing ='+string(smw,format='(i2)')+$
                          '  Horizontal line = background level'
  if ch2 eq 'x' then begin ; horizontal slices
    if kmax eq 0 then kmax=5*fix(0.2*ydim)-5
    if smw ge 3 then line=smooth(image(*,kmin),smw) else line=image(*,kmin)
    linex=line & xaxis=indgen(xdim+1)
    plot,xaxis(5:xdim-5),line(5:xdim-5),color=0,$
      xrange=[0,xdim],yrange=[0,imax+10],xstyle=1,ystyle=1,$
        xtitle='X(pixels)',ytitle='Intensity',title=heading,/nodata
    xyouts,0.15,0.90,smhead,color=0,/norm
    plots,[0,xdim-1],[background,background],color=0
;
    if ntime eq 1 then $
      for k=kmin,kmax,kstep do begin
        ss='Y cut '+string(k,format='(i3)')
        xyouts,0.15,0.87,ss,color=0,/norm
        if smw ge 3 then line=smooth(image(*,k),smw) else line=image(*,k)
        plot,xaxis(5:xdim-5),line(5:xdim-5),color=0,$
          xrange=[0,xdim],yrange=[0,imax+10],xstyle=1,ystyle=1,$
            xtitle='X(pixels)',ytitle='Intensity',title=heading
        xyouts,0.15,0.90,smhead,color=0,/norm
        plots,[0,xdim-1],[background,background]
        linex=line
      end $
    else $
      for k=kmin,kmax,kstep do begin & ss='Y cut '+string(k,format='(i3)')
        xyouts,400,background+6,ss,color=0
        if smw ge 3 then line=smooth(image(*,k),smw) else line=image(*,k)
          oplot,xaxis(5:xdim-5),line(5:xdim-5),color=0
            if add eq 0 then begin
              oplot,xaxis(5:xdim-5),linex(5:xdim-5),color=16777215
              oplot,xaxis(5:xdim-5),line(5:xdim-5),color=0
              plots,[0,xdim-1],[background,background],color=0
              linex=line
            end
          wait,ltimex
        xyouts,400,background+6,ss,color=16777215
      end
  end $
;
  else begin ; verticle slices
    if kmax eq 0 then kmax=5*fix(0.2*xdim)-5
    if smw ge 3 then line=smooth(image(kmin,*),smw) else line=image(kmin,*)
    linex=line & xaxis=indgen(ydim+1)
    plot,xaxis(5:ydim-5),line(5:ydim-5),color=0,$
      xrange=[0,ydim],yrange=[0,imax+10],xstyle=1,ystyle=1,$
        xtitle='Y(pixels)',ytitle='Intensity',title=heading,/nodata
    xyouts,0.15,0.90,smhead,color=0,/norm
    plots,[0,ydim-1],[background,background],color=0
;
    if ntime eq 1 then $
      for k=kmin,kmax,kstep do begin
        ss='X cut '+string(k,format='(i3)')
        xyouts,0.15,0.87,ss,color=0,/norm
        if smw ge 3 then line=smooth(image(k,*),smw) else line=image(k,*)
        plot,xaxis(5:ydim-5),line(5:ydim-5),color=0,$
          xrange=[0,ydim],yrange=[0,imax+10],xstyle=1,ystyle=1,$
            xtitle='Y(pixels)',ytitle='Intensity',title=heading
```



```
        xyouts,0.15,0.90,smhead,color=0,/norm
        plots,[0,ydim-1],[background,background]
        linex=line
      end $
    else $
      for k=kmin,kmax,kstep do begin & ss='X cut '+string(k,format='(i3)')
        xyouts,400,background+6,ss,color=0
        if smw ge 3 then line=smooth(image(k,*),smw) else line=image(k,*)
          oplot,xaxis(5:ydim-5),line(5:ydim-5),color=0
            if add eq 0 then begin
              oplot,xaxis(5:ydim-5),linex(5:ydim-5),color=16777215
              oplot,xaxis(5:ydim-5),line(5:ydim-5),color=0
              plots,[0,ydim-1],[background,background],color=0
              linex=line
            end
          wait,ltimex
        xyouts,400,background+6,ss,color=16777215
      end
  end
  xyouts,400,background+6,ss,color=0
;
return
end
;
;-----------------------------------------------------------------------
;
pro plot_ellipse,plane,ellipse
;
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com14,s11,s12,s22,s33,s34,s44,xcn,ycn,xpcn,ypcn,r12,r34,lx,ly,ld,ldx,ldy
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
;
  if plane eq 'x' then goto,xmask else $
  if plane eq 'y' then goto,ymask else $
  if plane eq '1' then goto,xwaist else $
  if plane eq '3' then goto,ywaist else $
  if plane eq 'c' then goto,ximage else $
  if plane eq 'd' then goto,yimage else return
;
xwaist:
    t=0 & a=1.0e-12*emitx^2/s22 & b=s22
    xc=xcn & xcp=xpcn
    goto,generate
ywaist:
    t=0 & a=1.0e-12*emity^2/s44 & b=s44
    xc=ycn & xcp=ypcn
    goto,generate
;
ximage:
    xc=xcn & xcp=xpcn
    goto,xplane
yimage:
    xc=ycn & xcp=ypcn
    goto,yplane
;
xmask:
    s11=phase(maxdata-1,0) & s12=phase(maxdata-1,1) & s22=phase(maxdata-1,2)
    xc=phase(maxdata-3,2) & xcp=phase(maxdata-3,3)
xplane:
    t=0.5*atan(2*s12/(s11-s22))
    a=cos(t)^2*s11+2*cos(t)*sin(t)*s12+sin(t)^2*s22
    b=sin(t)^2*s11-2*cos(t)*sin(t)*s12+cos(t)^2*s22
    goto,generate
ymask:
    s33=phase(maxdata-1,3) & s34=phase(maxdata-1,4) & s44=phase(maxdata-1,5)
    xc=phase(maxdata-3,4) & xcp=phase(maxdata-3,5)
yplane:
    t=0.5*atan(2*s34/(s33-s44))
    a=cos(t)^2*s33+2*cos(t)*sin(t)*s34+sin(t)^2*s44
    b=sin(t)^2*s33-2*cos(t)*sin(t)*s34+cos(t)^2*s44
;
```

```
generate:
  a=2*sqrt(a) & b=2*sqrt(b)
  ac=a*cos(t) & as=a*sin(t) & bc=b*cos(t) & bs=b*sin(t)
  npoints=16 & ellipse=fltarr(2,npoints)
  t=0 & dt=2*!pi/(npoints-1)
    for k=0,npoints-1 do $
      begin
        x=cos(t) & y=sin(t)
          xx=ac*x-bs*y & yy=as*x+bc*y & xx=100*xx+xc & yy=1000*yy+xcp
            ellipse(0,k)=xx & ellipse(1,k)=yy
        t=t+dt
      end
;
return
end
;
;-----------------------------------------------------------------------
;
pro plot_6d_space,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com14,s11,s12,s22,s33,s34,s44,xcn,ycn,xpcn,ypcn,r12,r34,lx,ly,ld,ldx,ldy
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
;
  ch=' ' & !p.multi=[0,2]
;
  if nmask eq 0 then begin & print,'Must generate mask.' & return & end
;  if ntime eq 0 then read,'Plot beamlet lables <y,n> ',chel6d
;
  tlts=strarr(4) & axis=strarr(8) & as=fltarr(6,2)
  xymin=fltarr(2) & xymax=xymin & xpmin=fltarr(2) & xpmax=xpmin
  tlts[0]=' X-Yp ('+filename+')' & tlts[1]='Y-Xp ('+filename+')'
  tlts[2]='Xp-Yp ('+filename+')' & tlts[3]=' X-Y ('+filename+')'
  axis[0]=' x[mm]' & axis[1]='yp[mr]' & axis[2]='y[mm]' & axis[3]='xp[mr]'
  axis[4]='xp[mr]' & axis[5]='yp[mr]' & axis[6]='x[mm]' & axis[7]=' y[mm]'
  as[0,0]=0 & as[1,0]=0 & as[2,0]=0 & as[3,0]=0 & as[4,0]=0 & as[5,0]=0
;
;  phase(*,0)=xabs, phase(*,1)=xp1, phase(*,2)=xp2
;  phase(*,3)=yabs, phase(*,4)=yp1, phase(*,5)=yp2
;
   rad=atan(1)/4.5 ; 10 degrees
     circle=fltarr(37,2)
       circle(*,0)=sin(indgen(37)*rad) & circle(*,1)=cos(indgen(37)*rad)
   index=intarr(4,6)
   index(0,0)=0 & index(0,1)=0 & index(0,2)=4 & index(0,3)=5
   index(1,0)=3 & index(1,1)=3 & index(1,2)=1 & index(1,3)=2
   index(2,0)=1 & index(2,1)=2 & index(2,2)=4 & index(2,3)=5
   index(3,0)=0 & index(3,1)=0 & index(3,2)=3 & index(3,3)=3
;
   ps=1000.0
   index(0,4)=1.0  & index(0,5)=ps
   index(1,4)=1.0  & index(1,5)=ps
   index(2,4)=ps   & index(2,5)=ps
   index(3,4)=1.0  & index(3,5)=1.0
;
;  Calculate the cross correlation sums xy, x-yp, xp-y, and xp-yp
;   xy=0
;   avebeamlet=0
;   for k=1,nbeam do begin
;       avebeamlet=avebeamlet+data(k,7)/nbeam
;   end
;   for k=1,nbeam do if plot_lets(k,1) eq 1 then $
;       xy=xy+(mask(k,0)*mask(k,1))*data(k,7)/
;   xyp=0
;   for k=1,nbeam do $
;     if plot_lets(k,1) eq 1 then $
;   xyp=xyp+(mask(k,0)*phase(k,4)+mask(k,0)*phase(k,5))
```



```
;   xpy=0
;   for k=1,nbeam do $
;     if plot_lets(k,1) eq 1 then $
;   xpy=xpy+(mask(k,1)*phase(k,1)+mask(k,1)*phase(k,2))
;   xpyp=0
;   for k=1,nbeam do $
;     if plot_lets(k,1) eq 1 then $
;  xpyp=xpyp+(phase(k,1)*phase(k,4)+phase(k,2)*phase(k,5))
;  xy=100*xy/nbeam & xyp=1e4*xyp/nbeam
;  xpy=1e4*xpy/nbeam & xpyp=1e6*xpyp/nbeam
;  angmom=xpy-xyp
;  print,'xy,xpy,xyp,xpyp,angmom'
;  print,xy,xpy,xyp,xpyp,angmom
;  moments(6)=xy & moments(7)=xpy
;  moments(8)=xyp & moments(9)=xpyp
  sxyp=sqrt(s11/4*s44/4)      ; mm*mr
  sxpy=sqrt(s33/4*s22/4)      ; mm*mr
  sxpyp=sqrt(s22/4*s44/4)     ; mr^2
  sxy=sqrt(s11/4*s33/4)       ; mm^2
print,'  '
print,'sxy,sxpy,sxyp,sxpyp'
print,sxy,sxpy,sxyp,sxpyp
print,'  '
  sslb=strarr(4,4)
  sslb(0,0)='sxyp,xyp,xyp/sxyp'
  sslb(0,1)='<x*yp>='+string(moments(8),format='(f8.2)')+' mm-mr'
  sslb(0,2)='sum ='+string(sxyp,format='(e10.2)')
  sslb(0,3)='S/x*yp='+string(moments(8)/sxyp,format='(e10.2)')
  sslb(1,0)='sxpy,xpy,xpy/sxpy'
  sslb(1,1)='<y*xp>='+string(moments(7),format='(f8.2)')+' mm-mr'
  sslb(1,2)='sum ='+string(sxpy,format='(e10.2)')  .
  sslb(1,3)='S/y*xp='+string(moments(7)/sxpy,format='(e10.2)')
  sslb(2,0)='sxpyp,xpyp,xpyp/sxpyp'
  sslb(2,1)='<xp*yp>='+string(moments(9),format='(f8.2)')+' mr^2'
  sslb(2,2)='sum  ='+string(sxpyp,format='(e10.2)')
  sslb(2,3)='S/xpyp='+string(moments(9)/sxpyp,format='(e10.2)')
  sslb(3,0)='sxy,xy,xy/sxy'
  sslb(3,1)='<x*y>='+string(moments(6),format='(f8.2)')+' mm^2'
  sslb(3,2)='sum ='+string(sxy,format='(e10.2)')
  sslb(3,3)='S/xy='+string(moments(6)/sxy,format='(e10.2)')
;
  for plane=0,3 do begin
    ix=2*plane & iy=ix+1 & expnd=1.5
    ix0=index(plane,0) & ix1=index(plane,1) & xscl=index(plane,4)
    ix2=index(plane,2) & ix3=index(plane,3) & pscl=index(plane,5)
    xymin(0)=xscl*min(phase(where(plot_lets[1:nbeam,1] gt 0),ix0))
    xymax(0)=xscl*max(phase(where(plot_lets[1:nbeam,1] gt 0),ix0))
    xymin(1)=xscl*min(phase(where(plot_lets[1:nbeam,1] gt 0),ix1))
    xymax(1)=xscl*max(phase(where(plot_lets[1:nbeam,1] gt 0),ix1))
;    xymin(0)=xscl*min(mask(where(plot_lets[1:nbeam,1] gt 0),ix0))
;    xymax(0)=xscl*max(mask(where(plot_lets[1:nbeam,1] gt 0),ix0))
;    xymin(1)=xscl*min(mask(where(plot_lets[1:nbeam,1] gt 0),ix1))
;    xymax(1)=xscl*max(mask(where(plot_lets[1:nbeam,1] gt 0),ix1))
    xpmin(0)=pscl*min(phase(where(plot_lets[1:nbeam,1] gt 0),ix2))
    xpmax(0)=pscl*max(phase(where(plot_lets[1:nbeam,1] gt 0),ix2))
    xpmin(1)=pscl*min(phase(where(plot_lets[1:nbeam,1] gt 0),ix3))
    xpmax(1)=pscl*max(phase(where(plot_lets[1:nbeam,1] gt 0),ix3))
    xminp=expnd*min(xpmin) & xmaxp=expnd*max(xpmax)
    xmin =expnd*min(xymin) & xmax =expnd*max(xymax)
    as[0,0]=xmin & as[5,0]=xmax & as[2,1]=xmaxp & as[3,1]=xminp
;
    plot,as[*,0],as[*,1],$
      xrange=[xmin,xmax],yrange=[xminp,xmaxp],xstyle=1,ystyle=1,$
        xtitle=axis[ix],ytitle=axis[iy],title=tlts[plane],/nodata
      plots,[0,0],[xminp,xmaxp] & plots,[xmin,xmax],[0,0]
    if plane eq 0 or plane eq 2 then lxl=0.1 else lxl=0.6
      write_scales,lxl
    if plane eq 2 then begin ; Xp-Yp plot circles
    for k=1,nbeam do $
      if (plot_lets(k,1) eq 1) then begin
;        xxax=0.5*xscl*(mask(k,ix0)+mask(k,ix1))
         xxax=0.5*xscl*(phase(k,ix0)+phase(k,ix1))
         xps=0.5*xscl*(phase(k,lx1)-phase(k,ix0))
```

```
;         yyax=0.5*xscl*(mask(k,ix2)+mask(k,ix3))
          yyax=0.5*xscl*(phase(k,ix2)+phase(k,ix3))
          yps=0.5*xscl*(phase(k,ix3)-phase(k,ix2))
          oplot,circle(*,0)*xps+xxax,circle(*,1)*yps+yyax
          plots,[xxax,xxax],[yyax-0.5,yyax+0.5]
          plots,[xxax-0.5,xxax+0.5],[yyax,yyax]
        end
      end else $
      for k=1,nbeam do $
        if (plot_lets(k,1) eq 1) then begin
          xxax=[xscl*phase(k,ix0),xscl*phase(k,ix1)]
          yyax=[pscl*phase(k,ix2),pscl*phase(k,ix3)]
          plots,xxax,yyax & oplot,xxax,yyax,psym=6
        end
      if chel6d eq 'y' then begin
        for k=1,nbeam do if plot_lets(k,1) eq 1 then $
          xyouts,xscl*phase(k,ix0),pscl*phase(k,ix2),string(k,format='(i3)')
      end
      if ntime eq 1 then iyl=-0.07 else iyl=0.95
      if (plane eq 0) or (plane eq 2) then begin
        if mask(0,1) ne 0 then begin
          iyl=iyl-0.025
            xyouts,0.18,iyl,'Mask rotation '+comdata(19)+' deg.',/norm
          end
      end
;   for j=1,3 do begin & iyl=iyl-0.022
;       xyouts,ixl+0.02,iyl,sslb(plane,j),/norm & end
;       if (plane eq 1) and (ntime eq 0) then begin
;        read,'Continue <y,n> ',ch & if ch eq 'n' then goto,myexit & end
;   end
        iyl=iyl-0.022
        xyouts,ixl+0.02,iyl,sslb(plane,1),/norm
       if (plane eq 1) and (ntime eq 0) then begin
        read,'Continue <y,n> ',ch & if ch eq 'n' then goto,myexit & end
    end
;
myexit: !p.multi=0
return
end
;
;-----------------------------------------------------------------------
;
pro write_scales,xps
;
  ss=string(!x.crange(0),format='(f7.2)')+$
      string(!x.crange(1),format='(f7.2)')+$
      string(!y.crange(0),format='(f7.2)')+$
      string(!y.crange(1),format='(f7.2)')
  xyouts,xps,0.0,strcompress(ss),/norm
;
return
end
;
;-----------------------------------------------------------------------
;
pro plot_phase_space,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com14,s11,s12,s22,s33,s34,s44,xcn,ycn,xpcn,ypcn,r12,r34,lx,ly,ld,ldx,ldy
common com15,version,comhead,comdata,moments
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
common com20,scales
;
  if max(plot_lets(1:nbeam,1)) lt 1 then begin
    print,'All beamlets have been elliminated. See bleamlet option' & return
```



```
  end
  if nmask eq 0 then begin & print,'Must generate mask.' & return & end
  if (nbeam eq 0) then begin & print,'RAW MODE' & return & end
  if ch2 eq ' ' than begin
          xplane='x' & yplane='y' & ldx=0 & ldy=0 & sloc='mask' & end $
  else if ch2 eq 'w' then begin
          xplane='1' & yplane='3' & ldx=lx & ldy=ly & sloc='waist' & end $
  else if ch2 eq 'm' then begin
          xplane='x' & yplane='y' & ldx=0 & ldy=0 & sloc='mask' & end $
  else if ch2 eq 'i' then begin
          xplane='c' & yplane='d' & ldx=drift & ldy=drift
          sloc='TV image' & end $
  else if ch2 eq 'd' then begin
          xplane='c' & yplane='d' & ldx=ld & ldy=ld
          sloc='L='+string(ld,format='(f7.3)')+' m' & end $
  else return
;
  ss='' & ssl=' ' & ssxx='' & ssy=''
  flag=0 & xacep1=-100 & xacep2=100 & yacep1=-100 & yacep2=100
  label=plot_lets(nbeam+1,1)
;
  tits=strarr(6)
  tits(0)='x(cm)' & tits(1)='xp(mr)' & tits(2)='X-Xp ('+filename+')'
  tits(3)='y(cm)' & tits(4)='yp(mr)' & tits(5)='Y-Yp ('+filename+')'
  xxax=fltarr(2) & yyax=xxax
;
  get_sigma,xx1,xx2,yy1,yy2
  xx1=100.0*xx1 & xx2=100.0*xx2 & yy1=100.0*yy1 & yy2=100.0*yy2
  p=1000.0
  as=fltarr(6,2) & xxax=fltarr(2) & yyax=fltarr(2)
  if chelscl eq 's' then begin
    fixscale: m=maxdata-2
      xmin=phase(m,2) & xmax=phase(m,3) & xminp=phase(m,4) & xmaxp=phase(m,5)
      ymin=xmin & ymax=xmax & yminp=xminp & ymaxp=xmaxp
    goto,twoplots
  end
;
  q=1.5
  xmina=fltarr(2) & xmaxa=xmina & ymina=xmina & ymaxa=xmina
  xpmin=fltarr(2) & xpmax=xpmin & ypmin=xpmin & ypmax=xpmin
  xmina(0)=min(xx1(where(plot_lets(1:nbeam,1) gt 0)))
  xmaxa(0)=max(xx1(where(plot_lets(1:nbeam,1) gt 0)))
  xmina(1)=min(xx2(where(plot_lets(1:nbeam,1) gt 0)))
  xmaxa(1)=max(xx2(where(plot_lets(1:nbeam,1) gt 0)))
  xpmin(0)=p*min(phase(where(plot_lets(1:nbeam,1) gt 0),1))
  xpmax(0)=p*max(phase(where(plot_lets(1:nbeam,1) gt 0),1))
  xpmin(1)=p*min(phase(where(plot_lets(1:nbeam,1) gt 0),2))
  xpmax(1)=p*max(phase(where(plot_lets(1:nbeam,1) gt 0),2))
  xmin=q*min(xmina) & xmax=q*max(xmaxa)
     xminp=q*min(xpmin) & xmaxp=q*max(xpmax)
;
  ymina(0)=min(yy1(where(plot_lets(1:nbeam,1) gt 0)))
  ymaxa(0)=max(yy1(where(plot_lets(1:nbeam,1) gt 0)))
  ymina(1)=min(yy2(where(plot_lets(1:nbeam,1) gt 0)))
  ymaxa(1)=max(yy2(where(plot_lets(1:nbeam,1) gt 0)))
  ypmin(0)=p*min(phase(where(plot_lets(1:nbeam,1) gt 0),4))
  ypmax(0)=p*max(phase(where(plot_lets(1:nbeam,1) gt 0),4))
  ypmin(1)=p*min(phase(where(plot_lets(1:nbeam,1) gt 0),5))
  ypmax(1)=p*max(phase(where(plot_lets(1:nbeam,1) gt 0),5))
  ymin=q*min(ymina) & ymax=q*max(ymaxa)
     yminp=q*min(ypmin) & ymaxp=q*max(ypmax)
;
  if scales(0) ne 0 then begin ; use fixed scales
    xmin=scales(0) & xmax=scales(1) & xminp=scales(2) & xmaxp=scales(3)
    ymin=xmin & ymax=xmax & yminp=xminp & ymaxp=xmaxp
  end
;
twoplots:
; x-xp phase space
  !p.multi=[0,2]
  dxlab=0.04*xmax & dylab=0.04*ymax
  plane='x'
  as(0,0)=xmin & as(1,0)=0 & as(2,0)=0 & as(3,0)=0 & as(4,0)=0 & as(5,0)=xmax
```

```
  as(0,1)=0 & as(1,1)=0 & as(2,1)=xmaxp & as(3,1)=xminp & as(4,1)=0 & as(5,1)=0
  plot,as(*,0),as(*,1),xrange=[xmin,xmax],yrange=[xminp,xmaxp],$
    xstyle=1,ystyle=1,xtitle=tits(0),ytitle=tits(1),title=tits(2),/nodata
;  write_scales,0.10
  for k=1,nbeam do $
    if ((phase(k,3) gt yacep1) and (phase(k,3) lt yacep2)) and $
      (plot_lets(k,1) eq 1) then begin
        xxax=[xx1(k),xx2(k)] & yyax=[p*phase(k,1),p*phase(k,2)]
          plots,xxax,yyax & oplot,xxax,yyax,psym=6
        if flag eq 1 and label eq 2 then $
          xyouts,xx1(k)+dxlab,p*phase(k,1),string(k,format='(i3)')
      end
  if ntime eq 1 then begin
    ss=strmid(comdata(8),1,8)+' mm-mr'
;    xyouts,0.15,0.90,ss,/norm
;    ss=comhead(10)+comdata(10) & xyouts,0.15,0.87,ss,/norm
  end
  plots,[0,0],[xminp,xmaxp] & plots,[xmin,xmax],[0,0]
  if (label eq 2) and (flag eq 0) then $
    for k=1,nbeam do if (plot_lets(k,1) eq 1) then $
        xyouts,xx1(k)+dxlab,p*phase(k,1),string(k,format='(i3)')
  if chellipse eq 'y' then begin
    plot_ellipse,xplane,ellipse & oplot,ellipse(0,*),ellipse(1,*) & end
;
; y-yp phase space
  plane='y'
  as(0,0)=ymin & as(1,0)=0 & as(2,0)=0 & as(3,0)=0 & as(4,0)=0 & as(5,0)=ymax
  as(0,1)=0 & as(1,1)=0 & as(2,1)=ymaxp & as(3,1)=yminp & as(4,1)=0 & as(5,1)=0
  plot,as(*,0),as(*,1),xrange=[ymin,ymax],yrange=[yminp,ymaxp],$
    xstyle=1,ystyle=1,xtitle=tits(3),ytitle=tits(4),title=tits(5),/nodata
;  write_scales,0.60
  for k=1,nbeam do $
    if ((phase(k,0) gt xacep1) and (phase(k,0) lt xacep2)) and $
      (plot_lets(k,1) eq 1) then begin
        xxax=[yy1(k),yy2(k)] & yyax=[p*phase(k,4),p*phase(k,5)]
          plots,xxax,yyax & oplot,xxax,yyax,psym=6
        if (flag eq 1) and (label eq 2) then $
          xyouts,yy1(k)+dylab,p*phase(k,4),string(k,format='(i3)')
      end
  plots,[0,0],[yminp,ymaxp] & plots,[ymin,ymax],[0,0]
  if ntime eq 1 then begin
    ss=strmid(comdata(8),10,8)+' mm-mr'
;    xyouts,0.65,0.90,ss,/norm
  end
  if (label eq 2) and (flag eq 0) then $
    for k=1,nbeam do if (plot_lets(k,1) eq 1) then $
        xyouts,yy1(k)+dylab,p*phase(k,4),string(k,format='(i3)')
  if chellipse eq 'y' then begin
    plot_ellipse,yplane,ellipse & oplot,ellipse(0,*),ellipse(1,*) & end
;
; plot labels
; -----------
  if ntime eq 0 then $
    begin & yy=0.96 & dy=0.023 & end else begin & yy=-0.05 & dy=0.030 & end
  ss='rms edge x emittance '+string(1.0e6*emitx,format='(f8.2)')+' mm-mr'
      yy=yy-dy & xyouts,0.12,yy,ss,/norm
      ss='rms edge y emittance '+string(1.0e6*emity,format='(f8.2)')+' mm-mr'
      xyouts,0.62,yy,ss,/norm
  ss='norm. x emittance '+string(1.0e6*betagamma*emitx,format='(f8.3)')+' mm-mr
'
      yy=yy-dy & xyouts,0.12,yy,ss,/norm
      ss='norm. y emittance '+string(1.0e6*betagamma*emity,format='(f8.3)')+' m
m-mr'
      xyouts,0.62,yy,ss,/norm
;  ss='x,r12,xp='+string(100*sqrt(s11),format='(f7.3)')+$
;       string(r12,format='(f7.4)')+string(1000*sqrt(s22),format='(f7.3)')
;       yy=yy-dy & xyouts,0.12,yy,ss,/norm          ; x beam
;       ss='y,r34,yp='+string(100*sqrt(s33),format='(f7.3)')+$
;          string(r34,format='(f7.4)')+string(1000*sqrt(s44),format='(f7.3)')
;       xyouts,0.62,yy,ss,/norm                     ; y beam
  ss=comhead(10)+comdata(10)+' at '+sloc
      yy=yy-dy & xyouts,0.12,yy,ss,/norm          ; level
    n2=strlen(comdata(1))
```



```
      if n2 gt 10 then ss=strmid(comdata(1),10,n2-10)
        xyouts,0.62,yy,ss,/norm                     ; nbeam smoothing
;
  if mask(0,1) ne 0 then begin
    yy=yy-0.03 & xyouts,0.65,yy,'Mask '+comdata(19)+' deg.',/norm & end
; ------------
;
  if ntime ne 0 then goto,myexit
  if ssi eq 'l' then goto,myexit
  ssi=''
    read,'Plot at given X(cm),Y(cm)slice, S)cale, E)llipse, L)abel, N)olabel '+
+$
                                                                  '<LF exit> ',ss
+1
  if strlen(ssi) eq 0 then goto,myexit
;
    if ssi eq 'e' then begin
        plot_ellipse,xplane,ellipse
        plot,ellipse(0,*),ellipse(1,*),$
          xrange=[xmin,xmax],yrange=[xminp,xmaxp],$
            xstyle=1,ystyle=1,xtitle=tits(0),ytitle=tits(1),title=tits(2)
        plots,[xmin,xmax],[0,0] & plots,[0,0],[xminp,xmaxp]
        plot_ellipse,yplane,ellipse
        plot,ellipse(0,*),ellipse(1,*),$
          xrange=[ymin,ymax],yrange=[yminp,ymaxp],$
            xstyle=1,ystyle=1,xtitle=tits(0),ytitle=tits(1),title=tits(2)
        plots,[ymin,ymax],[0,0] & plots,[0,0],[yminp,ymaxp]
        cc=' ' & read,'Plot ellipse on "data phase space" <y,n> ',cc
        if strlen(cc) gt 0 then begin
          if cc eq 'y' then chellipse='y' else chellipse='n' & end
            goto,myexit
      end $
    else if ssi eq 's' then begin
      read,'Enter phase space scales rmax(cm),rpmax(mr) LF(autoscale) ==> ',ss
      if strlen(ss) eq 0 then begin & chelscl=' ' & goto,myexit & end
        num=fltarr(2) & decode,ss,num,n & x=1.0*num(0) & xp=1.0*num(1)
          m=maxdata-2
            phase(m,2)=-x & phase(m,3)=x & phase(m,4)=-xp & phase(m,5)=xp
              chelscl='s'
      goto,fixscale
      end $
    else if ssi eq 'n' then begin
      label=0 & plot_lets(nbeam+1,1)=label & end $
    else if ssi eq 'l' then begin
      label=2 & plot_lets(nbeam+1,1)=label & goto,twoplots
    end
;
    if strpos('abcdefghijklmnopqrstuvwxyz',ssi) ge 0 then goto,myexit
    nx=strpos(ssi,' ',0) & sx=strmid(ssi,0,nx) & x=float(sx)
    ny=strlen(ssi)-nx & sy=strmid(ssi,nx+1,ny) & y=float(sy)
    flag=1 & deltax=xcmpix*xwindow & deltay=ycmpix*ywindow
    xacep1=x-deltax & xacep2=x+deltax & yacep1=y-deltay & yacep2=y+deltay
    ssx='X='+string(x,format='(f8.3)')+' cm'
    ssy='Y='+string(y,format='(f8.3)')+' cm'
  goto,twoplots
;
myexit: !p.multi=0
return
end
;
;-----------------------------------------------------------------------
;
pro plot_all_beam,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com8,window,xwindow,ywindow,chwin2,level
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
;
```

```
  xx=fltarr(4) & yy=xx
  xx(0)=0.07 & xx(1)=0.32 & xx(2)=0.55 & xx(3)=0.82
  yy(0)=0.96 & yy(1)=0.70 & yy(2)=0.46 & yy(3)=0.20
  !p.multi=[0,4,4]
  ip=0
  for k=1,nbeam do begin
    nx1=center(k,2) & nx2=center(k,3) & ny1=center(k,4) & ny2=center(k,5)
    iy=center(k,1)
    plot,image(nx1:nx2,iy),yrange=[0,256],xstyle=1,ystyle=1
    for j=ny1,ny2 do oplot,image(nx1:nx2,j)
    xyouts,xx(ip mod 4),yy(ip/4),'#'+string(k,format='(i3)'),/norm
    if ip eq 0 then xyouts,0.1,0.98,filename,/norm
    ip=ip+1
    if k mod 16 eq 0 then begin
      ip=0
        if ntime eq 0 then begin
          cc=' ' & read,'Continue <y,n,a)bort> ',cc
          if cc eq 'n' then goto,byebye
          if cc eq 'a' then abort
        end
    end
  end
;
byebye:
!p.multi=0
return
end
;
;-----------------------------------------------------------------------
;
pro plot_beamlet,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com8,window,xwindow,ywindow,chwin2,level
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
;
  nbm=0 & ss='' & ss_window='  Window='+comdata(5)
  num=intarr(nbeam)
  if ntime eq 0 then $
  begin
    if ch2 eq 'n' then $
      begin & decode,ssin,num,n & nb=num(0) & goto,nextb & end
    read,'Which beamlet, <1..'+string(nbeam,format='(i3)')+$
                '> D)elete P)urge R)estore M)ultiple <LF next one > ',ss
      if strlen(ss) eq 0 then nb=plot_lets(maxb,0)+1 else $
      if (ss eq 'd') then begin
        read,'Enter beamlets to be "ignored" A)ll ==> ',ss
          if (ss eq 'a') then begin & plot_lets(*,1)=0 & goto,myexit & end
          decode,ss,num,n
            for k=0,n-1 do begin & kb=num(k)
              if (kb gt 0) and (kb le nbeam) then plot_lets(kb,1)=0
            end
          goto,myexit
        end else $
      if (ss eq 'p') then begin
        print,'Enter beamlets to be PERMANENTLY DESTROYED'
          read,'==> ',ss
          decode,ss,num,n
            for k=0,n-1 do begin & kb=num(k)
              if (kb gt 0) and (kb lt nbeam) then begin
                data(kb:nbeam-1,*)=data(kb+1:nbeam,*)
                mask(kb:nbeam-1,*)=mask(kb+1:nbeam,*)
                center(kb:nbeam-1,*)=center(kb+1:nbeam,*)
                plot_lets(kb:nbeam-1,*)=plot_lets(kb+1:nbeam,*)
                nmask=nmask-1 & nbeam=nbeam-1
              end else if kb ge nbeam then begin
                nbeam=nbeam-1 & nmask=nmask-1 & end
            end
```



```
            for k=nbeam+1,nbeam+2 do begin
              for j=0,8 do  data(k,j)=0
                for j=0,1 do center(k,j)=0
            end
          goto,myexit
        end else $
      if (ss eq 'r') then begin
        read,'Enter beamlets to be "restored" A)ll ==> ',ss
          if (ss eq 'a') then begin & plot_lets(*,1)=1 & goto,myexit & end
          decode,ss,num,n
            for k=0,n-1 do begin & kb=num(k)
              if (kb gt 0) and (kb le nbeam) then plot_lets(kb,1)=1
              end
          goto,myexit
        end else $
      if (ss eq 'm') then begin
          read,'Enter list of beamlet ==> ',ss
            decode,ss,plot_lets,nbm
              if nbm gt 2 then !p.multi=[0,2,nbm/2]
          np=1 & goto,next
        end $
      else nb=fix(ss)
nextb:
          if (nb lt 1) or (nb gt nbeam) then begin
            print,'Sorry, not that many beamlets' & goto,myexit & end
          have=0 & for k=1,maxb do if nb eq plot_lets(k,0) then $
            begin & have=1 & np=k & end
          if have eq 0 then $
            begin & maxb=maxb+1 & plot_lets(maxb,0)=nb & np=maxb & end
  end else begin & np=1 & nb=0 & end
;
next:
  ch=' '
  nb=plot_lets(np,0)
  if ntime eq 1 then print,'                beamlet '+string(nb,format='(i3)')
  if (ss eq 'm') and (np mod 2 eq 0) then read,'Continue <y,n> ',ch
  if ch eq 'n' then return
  nx1=center(nb,0)-xwindow  & if nx1 lt 0 then nx1=0
  nx2=nx1+xwindow+xwindow   & if nx2 ge xdim then nx2=xdim-1
  ny1=center(nb,1)-ywindow  & if ny1 lt 0 then ny1=0
  ny2=ny1+ywindow+ywindow   & if ny2 ge ydim then ny2=ydim-1
  surface,image(nx1:nx2,ny1:ny2)
    xyouts,0.2,0.87,title,/norm
      xyouts,0.2,0.84,'Beamlet '+string(nb,format='(i3)')+'  [('+$
        string(center(nb,0),format='(i3)')+','+$
        string(center(nb,1),format='(i3)')+')'+ss_window+'   '+$
        string(nx1,format='(i3)')+'-'+string(nx2,format='(i3)')+','+$
        string(ny1,format='(i3)')+'-'+string(ny2,format='(i3)'),/norm
      xyouts,0.2,0.81,'Peak ='+string(data(nb,0),format='(i3)')+$
        '  Sum='+string(data(nb,7),format='(e10.3)')+$
        '                '+filename,/norm
      n2=strlen(comdata(1))-10
        if n2 gt 1 then xyouts,0.2,0.78,strmid(comdata(1),10,n2),/norm
  if ntime eq 1 then begin
    np=np+1 & if np le maxb then goto,next & maxb=0 & end
  if nbm gt 0 then begin & nbm=nbm-1 & np=np+1 & goto,next & end
;
!p.multi=0
return
  myexit: ptype(4)=0 & !p.multi=0
return
end
;
;-----------------------------------------------------------------------
;
pro plot_mask,ss
;
common com1,filename,dumpname,printname,dataname,title
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
common com15,version,comhead,comdata,moments
```

```
;
; pview=w)indow, c)ontour, s)urface, p)rofile, e)mit, b)eamlet, B)rightness
; ig, igxy, igr generate mask
;
  if nbeam eq 0 then return
  if strpos('gsrc',ch2) ge 0 then $
    begin & ch=ch2
      if strlen(ss) ge 3 then ch=strmid(ss,2,1)
        gen_mask,ch ;  get_emittance,' '
    end
;
  if (nmask gt nbeam) or (nmask eq 0) then begin
    print,'Must generate mask' & return & end
;
  if ch2 eq 'w' then begin
    ch2='c' & chw=' ' & ntime=0
      plot_window,chw,ntime
        oplot,mask(1:nmask,0),mask(1:nmask,1),psym=4
  end else begin & s=2*separation
    plot,mask(1:nmask,0),mask(1:nmask,1),psym=4,xstyle=1,ystyle=1,$
      xrange=[min(mask(1:nmask,0))-s,max(mask(1:nmask,0))+s],$
      yrange=[min(mask(1:nmask,1))-s,max(mask(1:nmask,1))+s],$
        xtitle='x(cm)',ytitle='y(cm)',title='MASK'+' for '+filename
  end
  dx=8*xcmpix & dy=2*ycmpix
  for k=1,nmask do xyouts,mask(k,0)-dx,mask(k,1)+dy,string(k,format='(i3)')
  if mask(0,1) ne 0 then xyouts,0.20,0.90,'Mask rotation '+comdata(19),/norm
;
return
end
;
;-----------------------------------------------------------------------
;
function odd,n     ; n=5  x=5.00 xh=2  x-2nh=1  /  n=6  x=6.00 xh=3  x-2xh=0
  x=1.0*n
    xh=fix(x/2)
      if abs(x-2.0*xh) gt 0.5 then odd=1 else odd=0
return,odd
end
;
;
;-----------------------------------------------------------------------
;
function atan9,xc,yc      ; ang=atan(x,y) where tan(ang)=y/x
;
  abx=abs(xc)
  if abx lt 1.e-10 then begin
    if yc eq 0 then return,0
    if yc lt 0 then return,1.5*!pi $
                else return,0.5*!pi
  end
  val=atan(abs(yc)/abx)
  if xc lt 0 then begin
    if yc eq 0 then return,!pi
    if yc lt 0 then return,val+!pi $
                else return,!pi-val
  end else begin
    if yc eq 0 then return,0
    if yc lt 0 then return,2*!pi-val $
                else return,val
  end
end
;
;-----------------------------------------------------------------------
;
pro gen_round,ctype
;
common com1,filename,dumpname,printname,dataname,title
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com15,version,comhead,comdata,moments
common com21,nmax,masklist
```



```
;
  if typemask eq 1 then begin
    print,'typemask cartesian.'
      ch=' ' & read,'    Do you want to generate a round mask <y,n,q)uit>',ch
    if ch ne 'y' then return
  end
;
  typemask=2
  xc=0 & yc=0 & rr=0
  nask=1 & masklist(nask,0)=xc & masklist(nask,1)=yc
  maxr=5 & dth=(itarr(maxr+1) & nth=intarr(maxr+1) & nrsum=nth
  pi=4.0*atan(1.0) & dt=0.25*pi
  dth(0)=1.0
  dth(1)=dt & dth(2)=0.5*dt & dth(3)=0.5*dt & dth(4)=0.5*dt & dth(5)=0.25*dt
  nth(0)=0 & nth(1)=8 & nth(2)=16 & nth(3)=16 & nth(4)=16 & nth(5)=32
  nrsum(0)=1 & for kr=1,maxr do nrsum(kr)=nrsum(kr-1)+nth(kr)
  for kr=1,maxr do begin
    rr=rr+separation & theta=0 & dt=dth(kr) & tmax=nth(kr)
      for kt=1,tmax do begin & nask=nask+1
        masklist(nask,2)=rr
        masklist(nask,0)=rr*cos(theta) & masklist(nask,1)=rr*sin(theta)
        theta=theta+dt
      end
  end
  plot,masklist(1:nask,0),masklist(1:nask,1),psym=4,$
                                xtitle='x(cm)',ytitle='y(cm)',title='ROUND MASK'
  dx=0.4*separation
  for k=1,nask do xyouts,masklist(k,0)-dx,masklist(k,1),string(k,format='(i2)')
  xx=fltarr(2) & yy=xx & kl=1
  for kr=1,maxr do begin
    kl=kl+nth(kr-1)
    for j=1,nth(kr)-1 do begin
      xx(0)=masklist(kl+j,0) & xx(1)=masklist(kl+j+1,0)
      yy(0)=masklist(kl+j,1) & yy(1)=masklist(kl+j+1,1)
      oplot,xx,yy
    end
    xx(0)=xx(1) & yy(0)=yy(1) & xx(1)=masklist(kl+1,0) & yy(1)=masklist(kl+1,1)
    oplot,xx,yy
  end
;
; find order of beam images to mask layout
; ----------------------------------------
  if ctype eq 's' then goto,doshow
  ch=' '
  read,'A)utogen, N)umericalList, S)how, <q)uit> ',ch
  if ch eq 'Q' or ch eq 'q' then begin & ctype='q' & return & end
  if ch eq 'A' or ch eq 'a' then goto,autogen
  if ch eq 'N' or ch eq 'n' then goto,numlist
  if ch eq 'S' or ch eq 's' then goto,doshow
  return
;
; ----------------------------------------
numlist:
  print,'Input for each beamlet the 'mask' number printed.'
  print,'The beamlet number is shown for the w)indow or t)vx data.'
  print,'If you make an error, enter ???HELP and the new value.'
  print,'                                                          Good Luck'
  print,'----------------------------------------------------------------'
  nmask=0; css='a'
  xx=fltarr(5) & yy=xx & dx=0.3*separation & dy=dx
  xx(0)=-dx & xx(1)=dx & xx(2)=dx & xx(3)=-dx & xx(4)=-dx
  yy(0)= dy & yy(1)=dy & yy(2)=-dy & yy(3)=-dy & yy(4)=dy
    for k=1,nbeam do begin
      read,'beamlet '+string(k,format='(i3,$)')+' --> ',css
        ki=fix(css)
        nmask=nmask+1
        mask(nmask,0)=masklist(ki,0)
        mask(nmask,1)=masklist(ki,1)
      oplot,xx+masklist(ki,0),yy+masklist(ki,1)
    end
  return
;
; ----------------------------------------
```

```
doshow:
  xx=fltarr(5) & yy=xx & dx=0.3*separation & dy=dx
  xx(0)=-dx & xx(1)= dx & xx(2)= dx & xx(3)=-dx & xx(4)=-dx
  yy(0)= dy & yy(1)= dy & yy(2)=-dy & yy(3)=-dy & yy(4)= dy
  for k=1,nbeam do begin
    oplot,xx+mask(k,0),yy+mask(k,1)
    xyouts,mask(k,0)-0.9*dx,mask(k,1)-0.9*dy,string(k,format='(I3)')
  end
  xyouts,0.1,0.98,filename+'  {'+dataname+'}',/norm
  if mask(0,1) ne 0 then xyouts,0.70,0.98,'Mask rotation '+comdata(19),/norm
  ctype='q'
  return
;
autogen:
; ---------------------------------------
  mdex=intarr(nbeam)
  ordr=fltarr(nbeam+1) & ordx=ordr & ordy=ordr & ordt=ordr
  ick=' ' & read,'Which beamlet is the center beamlet (q)uits) --> ',ick
  if ick eq 'q' then return
  ick=fix(ick) & icx=center(ick,0) & icy=center(ick,1)
  for k=1,nbeam do begin
    x=xcmplx*(center(k,0)-icx) & y=ycmplx*(center(k,1)-icy)
    ordx(k)=x & ordy(k)=y & ordr(k)=sqrt(x*x+y*y) & ordt(k)=atan9(x,y)
  end
  ordr(0)=999.0                    ; 0th point is not used, make big
  i0=where(ordr eq min(ordr))      ; i0 is center where r approx 0
  ordr(i0)=999.0                   ; make a big number
  dr=min(ordr)                     ; smallest value is the first radial step
  for k=1,nbeam do begin
    ir=fix(ordr(k)/dr)             ; radial index counter
    if ir gt maxr then begin & is=0 & ir=0 & end else is=ir-1
    it=fix(0.5+(ordt(k)/dth(ir)))  ; theta index counter
    if it eq nth(ir) then it=0     ; 360 degrees is also 0
    ii=nrsum(is)+it+1              ; mask index value
    mdex(k-1)=ii
  end
  mdex(i0-1)=1                     ; r=0 center is first mask point
;
  if typemask eq 2 then begin
    nmask=0
    for k=0,nbeam-1 do begin
      nmask=nmask+1
        idx=mdex(k)
        mask(nmask,0)=masklist(idx,0)
        mask(nmask,1)=masklist(idx,1)
        mask(nmask,2)=masklist(idx,2)
    end
    !err=0
  end
return
end
;
;-----------------------------------------------------------------------
;
pro gen_rectangular,ctype
; ctype=s(show)
common com1,filename,dumpname,printname,dataname,title
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmplx,ycmplx,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com15,version,comhead,comdata,moments
common com21,nmax,masklist
;
  if typemask eq 2 then begin
    print,'typemask is for round mask.'
      ch=' ' & read,'Do you want to generate a rectangular <y,n,q)uits> ',ch
    if ch ne 'y' then return
  end
;
  typemask=1
  if ctype eq 's' then begin
```



```
    nx=masklist(0,0) & ny=masklist(0,1)
  end else begin
    if masklist(0,0) eq 0 then begin
      num=fltarr(4)
      ss=' ' & read,'Enter nx,ny for rectangular mask q)uits ==> ',ss
      if strlen(ss) eq 0 or ss eq 'q' then return
        decode,ss,num,n
        nx=fix(num(0)) & ny=fix(num(1))
        masklist(0,0)=nx & masklist(0,1)=ny
    end else begin & nx=masklist(0,0) & ny=masklist(0,1) & end
  end
;
  nask=0
  yci=-fix(ny/2)*separation & yc=yci
  xci=-fix(nx/2)*separation & xc=xci
  for ky=1,ny do begin
    for kx=1,nx do begin
      nask=nask+1 & masklist(nask,0)=xc & masklist(nask,1)=yc
        xc=xc+separation
    end
    xc=xci & yc=yc+separation
  end
  plot,masklist(1:nask,0),masklist(1:nask,1),psym=4,xstyle=1,ystyle=1,$
                          xrange=[xci-separation,xci+(nx+1)*separation],$
                          yrange=[yci-separation,yci+(ny+1)*separation],$
                          xtitle='x(cm)',ytitle='y(cm)',title='RECTANGULAR MASK'
  dx=0.4*separation & dy=0.18*separation
  for k=1,nask do $
    xyouts,masklist(k,0)-dx,masklist(k,1)+dy,string(k,format='(I3)')
  if ctype eq 's' then goto,doshowx
;
; ---------------------------------------
  ch=' '
  read,'A)utogen, N)umericalList, S)how, <q)uit> ',ch
  if ch eq 'Q' or ch eq 'q' then begin & ctype='q' & return & end
  if ch eq 'A' or ch eq 'a' then goto,autogenx
  if ch eq 'N' or ch eq 'n' then goto,numlistx
  if ch eq 'S' or ch eq 's' then goto,doshowx
  return
;
; ---------------------------------------
autogenx:
  ss=' ' & read,'Which beamlet is center <1..'+string(nbeam)+'> q)uits ',ss
  if ss eq 'q' then return
  center(nbeam+1,0)=0 & center(nbeam+1,1)=0
  icn=fix(ss) & icx=center(icn,0) & icy=center(icn,1)
print,'icx,icy',icx,icy
  ixline=fix((nx/4)*separation/xcmplx)          ; calc big number advance y
  idx=abs(center(icn+1,0)-center(icn-1,0))/2    ; approx horizontal spacing
  idy=idx                                       ; approx vertical spacing
print,'idx,idy',idx,idy
;
; generate list of beamlets at x=0. Number of such beamlets is number of
;                                  .--------------   horizontal lines in mask.
;                                 /
  nxline=intarr(nbeam+1) & ido=0
dododo:
  nnl=0
  for k=1,nbeam do begin
    if abs(center(k,0)-icx) lt idx then begin
      nnl=nnl+1 & nxline(nnl)=k
    end
  end
; eliminate doublicate entries in nxline (tv image can be slightly rotated)
  k=1
  while k lt nnl do begin
    i=nxline(k) & j=nxline(k+1)
    if abs(center(i,1)-center(j,1)) lt idy/2 then begin
      if abs(center(j,0)-icx) lt abs(center(i,0)-icx) then $ ; save k+1 in k
        begin & ie=i & nxline(k)=nxline(k+1) & end else ie=j
          for n=k+1,nnl-1 do nxline(n)=nxline(n+1) & nnl=nnl-1 ; eliminate
    end else k=k+1
  end
```

```
;
  ido=ido+1
  if ido eq 2 then goto,from_dodo
;
  nxcl=1 & while nxline(nxcl) ne icn do nxcl=nxcl+1
  print,'Central beamlet is in line '+string(nxcl,format='(i2)')
  print,'Mask illuminated by '+string(nnl,format='(i2)')+$
        ' horizontal lines of beamlets.'
  print,format='("Central line beamlets: ",12i4)',nxline(1:nnl)
;
; Order beamlets left to right, bottom to top
; ----------------------------------------
  idy=abs(center(nxline(nxcl+1),1)-center(nxline(nxcl-1),1))/4
  isline=intarr(nnl+1) & nsline=isline ; line start and # of beamlets per line
  for k=1,nnl do begin
    i1=nxline(k) & if i1 eq 1 then i1=2
    ir=nxline(k)-1 & if ir eq 0 then ir=1
      while abs(center(i1,0)-center(i1-1,0)) lt ixline and $
            abs(center(i1,1)-center(i1-1,1)) lt idy            do i1=i1+1
      while abs(center(ir+1,0)-center(ir,0)) lt ixline and $
            abs(center(ir+1,1)-center(ir,1)) lt idy            do ir=ir-1
      nbs=i1-ir-1
      if i1 lt ir then ibs=i1-1 else ibs=ir+1
      if k eq 1 and nxline(1) eq 1 and i1 eq 2 then begin & ibs=1 & nbs=1 & end
      isline(k)=ibs & nsline(k)=nbs
    print,'Line'+string(k,format='(i2)')+$
          ' starts with beamlet '+string(ibs,format='(i2)')+$
          ' and has '+string(nbs,format='(i2)')+' beamlets.',i1,ir
  end
; ----------------------------------------
  print,'Total beamlets: ',fix(total(nsline))
  print,'Now re-ordering beamlets .....'
; ----------------------------------------
  for ki=1,nnl do begin                  ; for each line of mask
    is=isline(ki) & nb=nsline(ki) & nm=is+nb-1
    for j=is,nm do $
      for k=j+1,nm do $
        if center(k,0) lt center(j,0) then begin ; swap j,k
          ixs=center(k,0) & iys=center(k,1)
          center(k,0)=center(j,0) & center(j,0)=ixs
          center(k,1)=center(j,1) & center(j,1)=iys
        end
  end
; ----------------------------------------
  goto,dododo ; find new beamlet numbers for horizontally central beamlets
;
from_dodo:
  icn=nxline(nxcl)
  print,'Central beamlet new index:',icn
  print,format='("Line centers:",16i4)',nxline(1:nnl)
  for k=1,nbeam do begin
    center(k,2)=center(k,0)-xwindow & center(k,3)=center(k,0)+xwindow
    center(k,4)=center(k,1)-ywindow & center(k,5)=center(k,1)+ywindow
  end
  ssin='t2x' & plot_tv_frame,0 & wset,0
;
; look for missing beamlets
  nobeam=intarr(nbeam+1) & nob=0 & ixx=fix(1.6*idx)
print,'ixx',ixx
  for ki=1,nnl do $
    for k=isline(ki),isline(ki)+nsline(ki)-2 do $
      if center(k+1,0)-center(k,0) gt ixx then begin
dummmy=center(k+1,0)-center(k,0) & print,'dummmy',dummmy,'  k',k
        nob=nob+1 & nobeam(k)=1 & end
  if nob gt 0 then begin
    print,format='("Beamlets preceeding missing beamlets:",20i4)',$
      where(nobeam gt 0)
  end
  print,'-----------------------------------------------------------------'
;
;                                  .--- calc the mask index for the center hole
  imask=nx*fix(ny/2)+fix(nx/2)+1
  mask(0,0)=0 & mask(0,1)=0 & mask(0,2)=0  ; zero out rotation angle
```



```
;
; for center of mask upward, advance y
; layout the mask id numbers for the beamlets. Start at central beamlet
;
  for ki=nxcl,nnl do begin        ;          from center of beam upward.
    k=nxline(ki) & nmask=imask    ; .------- from center of lines to left
    while abs(center(k+1,0)-center(k,0)) lt ixline and $
          abs(center(k+1,1)-center(k,1)) lt idy do begin
      mask(k,0)=masklist(nmask,0) & mask(k,1)=masklist(nmask,1)
      k=k-1 & nmask=nmask-1
      if nobeam(k) gt 0 then nmask=nmask-1
    end
    k=nxline(ki)+1 & nmask=imask+1 ; .------- from center of lines to right
    while abs(center(k-1,0)-center(k,0)) lt ixline and $
          abs(center(k-1,1)-center(k,1)) lt idy do begin
      if nobeam(k-1) gt 0 then nmask=nmask+1
      mask(k,0)=masklist(nmask,0) & mask(k,1)=masklist(nmask,1)
      k=k+1 & nmask=nmask+1
    end
    imask=imask+nx
  end
;
; for center of mask downward, decrementing y
;                                  .---- mask index for line below center line
  imask=(nx-1)*fix(ny/2)
  for ki=nxcl-1,1,-1 do begin
    k=nxline(ki) & nmask=imask
;                                  .----- from center of lines to left
    while abs(center(k+1,0)-center(k,0)) lt ixline and $
          abs(center(k+1,1)-center(k,1)) lt idy do begin
      if nobeam(k) gt 0 then nmask=nmask-1
      mask(k,0)=masklist(nmask,0) & mask(k,1)=masklist(nmask,1)
      k=k-1 & nmask=nmask-1
    end
    k=nxline(ki)+1 & nmask=imask+1
;                                  .----- from center of lines to right
    while abs(center(k-1,0)-center(k,0)) lt ixline and $
          abs(center(k-1,1)-center(k,1)) lt idy do begin
      if nobeam(k-1) gt 0 then nmask=nmask+1
      mask(k,0)=masklist(nmask,0) & mask(k,1)=masklist(nmask,1)
      k=k+1 & nmask=nmask+1
    end
    imask=imask-nx
  end
;
  nmask=nbeam
  for k=1,nmask do mask(k,2)=sqrt(mask(k,0)^2+mask(k,1)^2)
  get_beamlets,0 & get_emittance,'x'
  goto,doshowx
;
; ----------------------------------------
numlistx:
  print,'Input for each beamlet the "mask" number printed.'
  print,'The beamlet number is shown for the w)indow or t)vx data.'
  print,'If you make an error, enter ???HELP and the new value.'
  print,'                                                          Good Luck'
  print,'-----------------------------------------------------------------'
  nmask=0; css='a'
  xx=fltarr(5) & yy=xx & dx=0.3*separation & dy=dx
  xx(0)=-dx & xx(1)=dx & xx(2)=dx & xx(3)=-dx & xx(4)=-dx
  yy(0)= dy & yy(1)=dy & yy(2)=-dy & yy(3)=-dy & yy(4)=dy
    for k=1,nbeam do begin
      read,'beamlet '+string(k,format='(i3,$)')+' --> ',css
        ki=fix(css)
        nmask=nmask+1
        mask(nmask,0)=masklist(ki,0)
        mask(nmask,1)=masklist(ki,1)
        mask(nmask,2)=sqrt(mask(nmask,0)^2+mask(nmask,1)^2)
      oplot,xx+masklist(ki,0),yy+masklist(ki,1)
    end
return
;
; ----------------------------------------
```

```
doshowx:
  xx=fltarr(5) & yy=xx & dx=0.4*separation & dy=dx
  xx(0)=-dx & xx(1)= dx & xx(2)= dx & xx(3)=-dx & xx(4)=-dx
  yy(0)= dy & yy(1)= dy & yy(2)=-dy & yy(3)=-dy & yy(4)= dy
  for k=1,nbeam do begin
    oplot,xx+mask(k,0),yy+mask(k,1)
    xyouts,mask(k,0)-0.9*dx,mask(k,1)-0.9*dy,string(k,format='(i3)')
  end
  xyouts,0.1,0.98,filename+'  ('+dataname+')',/norm
  if mask(0,1) ne 0 then xyouts,0.70,0.98,'Mask rotation '+comdata(19),/norm
  ctype='q'
  return
;
end
;
;-----------------------------------------------------------------------
;
pro vary_mask,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com15,version,comhead,comdata,moments
common com21,nmax,masklist
;
  if nmask lt 1 then begin & print,'First generate the mask' & return & end
  if ntime eq 0 then begin
  read,'Enter minimum, maximum, and step (mask rotation angle deg) ',amn,amx,da
    n=nmax+nmax & masklist(n,0)=amn & masklist(n,1)=amx & masklist(n,2)=da
  end else begin
    n=nmax+nmax & amn=masklist(n,0) & amx=masklist(n,1) & da=masklist(n,2)
  end
;
  xyn=fltarr(1+fix((amx-amn)/da),4)  ; array to save emittance vs angle
  msave=mask & usave=mask
  aa=mask(0,1)                                  ; inital mask rotation angle
  if aa ne 0 then begin                         ; remove inital mask rotation
    aa=-aa*!pi/180.0 & sn=sin(aa) & cs=cos(aa)
      for k=1,nmask do begin
        x=msave(k,0) & y=msave(k,1)
          usave(k,0)=x*cs-y*sn & usave(k,1)=x*sn+y*cs
      end
  end
  ang=amn
  plot,[amn-da,amx+da],[2*1e6*betagamma*emitx,2*1e6*betagamma*emity],/nodata,$
    title='Normalized Emittance vs Mask Rotation Angle',$
      xtitle='Angle(deg)',ytitle='N. Emit(mm-mr)'
  xyouts,0.12,0.98,filename,/norm
;
  ian=0
  while ang le amx do begin
    aa=ang*!pi/180.0 & sn=sin(aa) & cs=cos(aa)
    for k=1,nmask do begin
      x=usave(k,0) & y=usave(k,1) & mask(k,0)=x*cs-y*sn & mask(k,1)=x*sn+y*cs
    end
    get_emittance,' ' & xyn(ian,0)=ang & xyn(ian,1)=1e6*betagamma*emitx & xyn(i
+an,2)=1e6*betagamma*emity
      xyouts,ang,1e6*betagamma*emitx,'x'
        xyouts,ang,1e6*betagamma*emity,'y'
    ian=ian+1 & ang=ang+da
  end
  oplot,xyn(*,0),xyn(*,1) & oplot,xyn(*,0),xyn(*,2)
;
; locate where emitx=emity and where RMS emittance is minimum
  for k=0,ian-1 do begin
    rms=sqrt(xyn(k,1)^2+xyn(k,2)^2)/sqrt(2) & xyn(k,3)=rms
  end
  oplot,xyn(*,0),xyn(*,3)
  xyouts,xyn(ian-1,0),xyn(ian-1,3),'RMS'
;
  if xyn(0,1) gt xyn(0,2) then begin & ibig=1 & ilit=2 & end $
                                 else begin & ibig=2 & ilit=1 & end
```



```
  kmax=size(xyn) & ikmax=kmax(1)-1
  k=1 & while xyn(k,ibig) gt xyn(k,ilit) and k lt ikmax do k=k+1
  if k lt ikmax then begin ; emitx and emity crossed
    print,'xy-transfere index: ',k,xyn(k,ibig),xyn(k,ilit)
;
;   Linear interpolation
    am=xyn(k-1,0)+(xyn(k-1,1)-xyn(k-1,2))*(xyn(k,0)-xyn(k-1,0))/$
                                  (xyn(k,2)-xyn(k-1,2))-(xyn(k,1)-xyn(k-1,1))
    exy=xyn(k-1,1)+(xyn(k,1)-xyn(k-1,1))*(am-xyn(k-1,0))/(xyn(k,0)-xyn(k-1,0))
;
;   How use three point Lagrangian interpolation
    ax=xyn(k-1,1)-2*xyn(k,1)+xyn(k+1,1)
    ay=xyn(k-1,2)-2*xyn(k,2)+xyn(k+1,2)
    bx=xyn(k+1,1)-xyn(k-1,1)
    by=xyn(k+1,2)-xyn(k-1,2)
    a=(ay-ax)/2 & b=(by-bx)/2 & c=xyn(k,2)-xyn(k,1)
    r=sqrt(b^2-4*a*c) & an=(-b-r)/(2*a) & ap=(-b+r)/(2*a)
    am=xyn(k,0)+an*da & exy=xyn(k,1)+bx*an/2+ax*an^2/2
    ss='Emx='+string(exy,format='(f8.3)')+$
        '    Angle(Emx=Emy):'+string(am,format='(f8.3)')
    xyouts,0.18,0.92,ss,/norm
;
;   have three points, y1=emit(ir-1), y2=emit(ir), y3=emit(ir+1)
;   use Lagrangian interpolation to find location of minimum and the value.
;
;           ( x-x2)( x-x3)       ( x-x1)( x-x3)       ( x-x1)( x-x2)
;   y(x) = -------------- y1 + -------------- y2 + -------------- y3
;           (x1-x2)(x1-x3)       (x2-x1)(x2-x3)       (x3-x1)(x3-x2)
;
;   x=x2+d   a=d/h  h=x2-x1  -1.0<= d <=1.0
;   y(x) = y2 + 1/2(y3-y1)*a + 1/2(y1-2y2+y3)*a^2
;   minimum is located at: dy/dx=0
;       x = x2 + 1/2(y1-y3)(x2-x1)/(y1-2y2+y3)
;       y = y2 - 1/8(y1-y3)^2/(y1-2y2+y3)
;
    ir=where(xyn(*,3) eq min(xyn(*,3)))
    ay=xyn(ir-1,3)-2*xyn(ir,3)+xyn(ir+1,3)
    ar=xyn(ir,0)+0.5*da*(xyn(ir-1,3)-xyn(ir+1,3))/ay
    rms=xyn(ir,3)-0.125*(xyn(ir-1,3)-xyn(ir+1,3))^2/ay
    ss='RMS='+string(rms,format='(f8.3)')+$
        '    Angle(RMS=min):'+string(ar,format='(f8.3)')
    xyouts,0.18,0.89,ss,/norm
  end
;
  mask=msave
  get_emittance,' '
return
end
;
;-----------------------------------------------------------------------
;
pro gen_mask,ctype
;
; ctype='a' automatic, 'g'=generate, 's'=shift, 'c'=center about zero
; ctype='r' generate round mask, ctype='g' generate cartesian mask.
;
common com3,ssln,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com15,version,comhead,comdata,moments
;
  if nbeam eq 0 then return
  if strpos('cgr',ctype) ge 0 then begin
        ssln='t2x' & plot_tv_frame,0 & wset,0 & end
;
  if ctype eq 'c' then goto,finishup else $
  if ctype eq 'g' then begin ; generate nx by ny rectangular mask
      gen_rectangular,ctype
        if ctype eq 'q' then return
        ctype=' '
```

```
3186          goto,finishup
3187        end else $
3188    if ctype eq 'r' then begin ; generate round mask of nbeams holes
3189          gen_round,ctype
3190          print,'Number of beamlets =',nbeam,'  number of mask holes =',nmask
3191          if ctype eq 'q' then return
3192          goto,finishup
3193    end
3194  ;
3195    if ctype eq 's' then begin ; shift mask
3196      read,'Enter mask x_shift(cm) and y_shift(cm) <x,y> ==> ',xc,yc
3197        for k=1,nmask do $
3198          begin & mask(k,0)=mask(k,0)+xc & mask(k,1)=mask(k,1)+yc & end
3199            goto,finishup
3200    end
3201  ;
3202  finishup: ch=' '
3203    comdata(19)='0'
3204    if nmask eq 0 then return
3205  ;
3206    mask(0,0)=0 & mask(0,1)=0 ; (0,0) not used, (0,1)=mask rotation angle
3207    if ctype eq 'c' then ch='y' else if ctype ne 'r' then $
3208      read,'Center mask about zero                    <y,n> ==> ',ch
3209    if ch eq 'y' then begin
3210      xc=total(mask(1:nmask,0))/nmask & yc=total(mask(1:nmask,1))/nmask
3211      for j=1,nmask do mask(j,2)=sqrt((mask(j,0)-xc)^2+(mask(j,1)-yc)^2)
3212      for j=1,nmask do $
3213        begin & mask(j,0)=mask(j,0)-xc & mask(j,1)=mask(j,1)-yc & end
3214    end
3215  ;
3216    read,'Has image been through a cross over          <y,n> ==> ',ch
3217    nc=strpos(comdata(6),']')+2
3218    if nc gt strlen(comdata(6)) then cross='N' else $
3219      cross=strmid(comdata(6),strpos(comdata(6),']')+2,1) ; =N or C
3220    if cross eq ' ' then cross='N'
3221    if ((ch eq 'y') and (cross eq 'N')) then cross='C'
3222    if ((ch eq 'n') and (cross eq 'C')) then begin & ch='y' & cross='N' & end
3223    if ch eq 'y' then for j=1,nmask do $
3224      begin & mask(j,0)=-mask(j,0) & mask(j,1)=-mask(j,1) & end
3225    if cross eq 'C' then mesg='rossover' else mesg='o_Crossover'
3226    comdata(6)=strmid(comdata(6),0,strpos(comdata(6),']')+1)+' '+cross+mesg
3227  return
3228  end
3229  ;
3230  ;-----------------------------------------------------------------
3231  ;
3232  pro adjust_centers
3233  common com2,xdim,ydim,imswch,image,iobject
3234  common com5,plot_lets,center,maxb,nbeam
3235  common com8,window,xwindow,ywindow,chwin2,level
3236  ;
3237  ; ch='y'
3238  ; read,'Adjust beamlet centers <a)uto,m)anule,q)uit> ',ch  ; TEST
3239  ; if ch eq 'a' then goto,dodoa
3240  ; if ch eq 'm' then goto,dodom
3241  ; return
3242  ;
3243  dodoa:
3244    for k=1,nbeam do begin
3245      nx=center(k,0) & nx1=nx-xwindow & nx2=nx+xwindow
3246      ny=center(k,1) & ny1=ny-ywindow & ny2=ny+ywindow
3247      jmax=max(image(nx1:nx2,ny1:ny2))
3248      icc=where(image(nx1:nx2,ny1:ny2) eq jmax)
3249      ic=icc(0)
3250      iy=ic/(2*xwindow+1)
3251      ix=ic-iy*(2*xwindow+1)
3252      idx=ix-xwindow
3253      idy=iy-ywindow
3254      center(k,0)=center(k,0)+idx
3255      center(k,1)=center(k,1)+idy
3256    end
3257  return
3258
```



```
3259  ;
3260  dodom:
3261    levels=30+12*indgen(20)
3262  again:
3263      ss=''
3264      read,'Enter beamlet number <1..nbeam,a)bort,q)uits> ',ss
3265      if ss eq 'a' then abort
3266      if ss eq 'q' then return
3267      k=int(ss)
3268      if k lt 1 or k gt nbeam then goto,again
3269  ;
3270      nx=center(k,0) & nx1=nx-xwindow & nx2=nx+xwindow
3271      ny=center(k,1) & ny1=ny-ywindow & ny2=ny+ywindow
3272      jmin=min(image(nx1:nx2,ny1:ny2))
3273      jmax=max(image(nx1:nx2,ny1:ny2))
3274  ;
3275      contour,image(nx1:nx2,ny1:ny2),levels=levels
3276      plots,[xwindow-5,xwindow+5],[ywindow,ywindow]
3277      plots,[xwindow,xwindow],[ywindow-5,ywindow+5]
3278  ;
3279      icc=where(image(nx1:nx2,ny1:ny2) eq jmax)
3280      ic=icc(0)
3281      iy=ic/(2*xwindow+1)
3282      ix=ic-iy*(2*xwindow+1)
3283      iz=size(icc)
3284      print,'size,ic,ix,iy=',iz(1),ic,ix,iy
3285      idx=ix-xwindow
3286      idy=iy-ywindow
3287  ;
3288      xc=xwindow+idx & yc=ywindow+idy
3289      plots,[xc-2,xc+2],[yc,yc]
3290      plots,[xc,xc],[yc-2,yc+2]
3291  ;
3292      goto,again
3293  ;
3294  end
3295  ;
3296  ;-----------------------------------------------------------------
3297  ;
3298  pro get_centers,cerror
3299  ;
3300  common com2,xdim,ydim,imswch,image,iobject
3301  common com3,asin,pview,ch2
3302  common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
3303  common com5,plot_lets,center,maxb,nbeam
3304  common com8,window,xwindow,ywindow,chwin2,level
3305  common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
3306  common com10,xcent,ycent,nxcent,nycent,imin,imax,background
3307  common com11,lpass,bradius,xave,yave,ibeamn,jradius,bjj
3308  common com15,version,comhead,comdata,moments
3309  common com16,xwcent,ywcent,xpix0,ypix0,pixsize
3310  common com19,smoothing
3311  common com22,flag,numpix
3312  ;
3313    wset,2 & tv,image & wset,0
3314    iobject=image
3315  ;
3316    nxcent=0 & nycent=0
3317    ns=size(image) & xsize=ns(1) & ysize=ns(2) & maxsize=max(ns(1:2))
3318
3319    imin=min(image) & imax=max(image)
3320      backx=total(image(*,0))/xdim+total(image(*,ydim-1))/xdim
3321      backy=total(image(0,*))/ydim+total(image(xdim-1,*))/ydim
3322      background=fix((backx+backy)/4.0)
3323    ave=imax/4
3324
3325    if cerror eq 'R' then begin
3326  ;   zero out beamlet regions in iobject from restart values of center()
3327      for k=1,nbeam do iobject(center(k,2):center(k,3),center(k,4):center(k,5))=0
3328      goto,finish
3329    end
3330  ;
3331    ch=' '
```

```
3332    read,'Find centers <a)uto, p)oint_and_click, q)uit> ',ch
3333    if ch eq 'q' then return
3334    if ch eq 'p' then goto,do_manulely
3335  ;
3336    print,'Threshold: '+string(ave,format='(f5.0)')
3337    read,'Enter adjustment to threshold, LF for none --> ',dax
3338    ave=ave+dax
3339    print,'Adjusted threshold: '+string(ave,format='(f5.0)')
3340
3341    cerror='t'
3342    nbeam=0
3343    check=where(image gt ave)
3344    ipnt=(size(check))(1)
3345    print,'Number of locations above threshold: '+string(ipnt,format='(i5)')
3346    print,'-----------------------------------------------------'
3347    print,' Beamlet  points  x-range   y-range  intensity'
3348  ;
3349  skip1:
3350    check=where(iobject gt ave)
3351    if (size(check))(1) lt 64 then goto,finished
3352    iy=check(0)/xdim
3353    ix=check(0)-iy*xdim
3354    beam=search2d(image,ix,iy,byte(ave),byte(255),/diagonal)
3355  ;
3356    if (size(beam))(1) gt 16 then begin
3357      nbeam=nbeam+1
3358      ib=3
3359      indy=beam/xdim
3360      indx=beam-indy*xdim
3361      ixmin=min(indx)-ib & ixmax=max(indx)+ib
3362      iymin=min(indy)-ib & iymax=max(indy)+ib
3363      if ixmin lt 0 then ixmin=0
3364      if iymin lt 0 then iymin=0
3365      if ixmax ge xdim then ixmax=xdim-1
3366      if iymax ge ydim then iymax=ydim-1
3367      jmin=min(image(ixmin:ixmax,iymin:iymax))
3368      jmax=max(image(ixmin:ixmax,iymin:iymax))
3369
3370      icc=where(image(ixmin:ixmax,iymin:iymax) eq jmax)
3371      ic=icc(0)
3372      nx=ixmax-ixmin+1
3373      iyc=ic/nx
3374      ixc=ic-iyc*nx
3375      iyc=iymin+iyc
3376      ixc=ixmin+ixc
3377  ;
3378      center(nbeam,0)=ixc & center(nbeam,1)=iyc
3379      ix1=ixc-xwindow & ix2=ixc+xwindow & iy1=iyc-ywindow & iy2=iyc+ywindow
3380      if ix1 lt 0 then ix1=1 & if ix2 ge xdim then ix2=xdim-2
3381      if iy1 lt 0 then iy1=1 & if iy2 ge ydim then iy2=ydim-2
3382      center(nbeam,2)=ix1 & center(nbeam,3)=ix2
3383      center(nbeam,4)=iy1 & center(nbeam,5)=iy2
3384      print,format='(4x,i2,4x,i4,3x,i3,":",i3,4x,i3,":",i3,4x,i3,":",i3)',$
3385                    nbeam,(size(beam))(1),ixmin,ixmax,iymin,iymax,jmin,jmax
3386  numpix=intarr(100)
3387  numpix(nbeam)=(size(beam))(1)
3388  print,'nbeam,numpix',nbeam,numpix(nbeam)
3389    end
3390  ;
3391  ; zero out this beamlet from iobject array
3392    iobject(beam)=0
3393    goto,skip1
3394  ;
3395  do_manulely:
3396    nbeam=0
3397    nmask=0
3398    num=intarr(4)
3399    set_contour,num,'d'
3400    ch='1'
3401    fix_beam,ch
3402    goto,finish
3403  ;
3404  finished:
```



```
3405    nmask=0
3406  ;
3407  finish:
3408    xcent=center(*,0)
3409    ycent=center(*,1)
3410    nxcent=nbeam
3411    nycent=nbeam
3412    print,'Number of beamlets '+string(nbeam,format='(i3)')
3413    plot_lets(1:nbeam+2,1)=1
3414  ;
3415    comdata(1)='['+string(xdim,format='(i3)')+','+string(ydim,format='(i3)')+']'+
+$
3416       ' '+string(nbeam,format='(i3)')+' beamlets'+$
3417       ' Smoothing'+string(smoothing,format='(i3)')
3418    comdata(2)='['+string(imin,format='(i3)')+','+$
3419                  string(imax,format='(i3)')+'] '+string(background,format='(i1)')
3420    loc=where(image eq imax) & sloc=size(loc)
3421    ym=loc(0)/xdim & xm=loc(0)-ym*ydim
3422    print,'Background                    =',background
3423    print,'min,max intensity of image ='+comdata(2)
3424    print,'Number of x and y beamlets =',nxcent,nycent
3425    xpix0=xm & ypix0=ym
3426    set_surface
3427  ;
3428  return
3429  end
3430  ;
3431  ;-----------------------------------------------------------------------
3432  ;
3433  pro zero_image,image,ch
3434    ch=' ' & ss=''
3435    nsize=size(image) & nx=nsize(1)-1 & ny=nsize(2)-1
3436    ss=strtrim(strcompress(string(nx)+string(ny)),2)
3437    print,'Enter sub region indexies of image to be "zero_ed" '
3438    print,'  index range IMAGE('+ss+')'
3439  more:
3440    ss='' & read,'    nx1,ny1,nx2,ny2 for diagonally opposite corners ==>',ss
3441      if strlen(ss) eq 0 then return
3442        ss=ss+' '
3443          nb1=strpos(ss,' ',0) & s1=strmid(ss,0,nb1)
3444          nb2=strpos(ss,' ',nb1+1) & s2=strmid(ss,nb1+1,nb2-nb1+1)
3445          nb3=strpos(ss,' ',nb2+1) & s3=strmid(ss,nb2+1,nb3-nb2+1)
3446          nb4=strpos(ss,' ',nb3+1) & s4=strmid(ss,nb3+1,nb4-nb3+1)
3447          nx1=fix(s1) & ny1=fix(s2) & nx2=fix(s3) & ny2=fix(s4)
3448    if (nx1 lt 0) or (nx1 gt nx) or (nx2 lt 0) or (nx2 gt nx) or $
3449       (ny1 lt 0) or (ny1 gt ny) or (ny2 lt 0) or (ny2 gt ny) then goto,error
3450    if ch eq ' ' then begin
3451      read,'Enter value for "zero" i.e. 0 etc. ==> ',ss
3452        if strlen(ss) gt 0 then zero=fix(ss) else zero=0 & end
3453    read,'    Are you sure you want to zero image('+ss+') <y,n> ',ch
3454      if ch ne 'y' then goto,quit
3455    for kx=nx1,nx2 do for ky=ny1,ny2 do image(kx,ky)=zero
3456    goto,quit
3457  error: print,'Error in index range '
3458  quit:
3459    read,'Zero another sub_region <y,n> ',ch
3460      if ch eq 'y' then goto,more
3461    read,'Regenerate beamlets      <y,n> ',ch
3462  ;
3463  return
3464  end
3465  ;
3466  ;-----------------------------------------------------------------------
3467  ;
3468  pro dump_image,image
3469  common com1,filename,dumpname,printname,dataname,title
3470  common com5,plot_lets,center,maxb,nbeam
3471  common com8,window,xwindow,ywindow,chwin2,level
3472  common com10,xcent,ycent,nxcent,nycent,imin,imax,background
3473  ;
3474    ch=' ' & ss=''
3475    read,'Enter B)eamlet, C)orner <b,c> ',ch
3476      if strpos(' bc',ch) lt 1 then return
```

```
    if ch eq 'b' then begin
      read,'Enter beamlet number <1..'+string(nbeam,format='(i3)')+'> ',ss
        nb=fix(ss)
          if (nb lt 1) or (nb gt nbeam) then begin
            print,'Sorry, out of range <1..'+string(nbeam,format='(i3)')
              return & end
        nx=center(nb,0) & ny=center(nb,1)
          nx1=nx-xwindow & ny1=ny-ywindow & nx2=nx+xwindow & ny2=ny+ywindow
    end else if ch eq 'c' then begin
      read,'Enter (x,y)1,(x,y)2 for diagonal corners ==> ',ss
      if strlen(ss) gt 0 then begin & ss=ss+' '
        nb1=strpos(ss,' ',0) & s1=strmid(ss,0,nb1)
        nb2=strpos(ss,' ',nb1+1) & s2=strmid(ss,nb1+1,nb2-nb1+1)
        nb3=strpos(ss,' ',nb2+1) & s3=strmid(ss,nb2+1,nb3-nb2+1)
        nb4=strpos(ss,' ',nb3+1) & s4=strmid(ss,nb3+1,nb4-nb3+1)
        nx1=fix(s1) & ny1=fix(s2) & nx2=fix(s3) & ny2=fix(s4)
    end else begin & nx1=40 & ny1=60 & nx2=57 & ny2=80 & end
      nx=(nx1+nx2)/2 & ny=(ny1+ny2)/2
      ; while xcent,ycent find nb ??
      nb=1
    end
  strange='image region around beamlet '+string(nb)
;
  nxend=nx2 & if nx2-nx1 gt 18 then nx2=nx1+17
;
openw,1,dumpname, /variable
  printf,1,strcompress(filename)+'  ==> '+strcompress(dumpname)
  printf,1,strange
nextpage:
  ss=' kx=' & line='    ' & for kx=nx1,nx2 do line=line+'----'
    for kx=nx1,nx2 do ss=ss+string(kx,format='(i4)') & printf,1,ss
      printf,1,line
        for ky=ny1,ny2 do begin & ss='|'
          for kx=nx1,nx2 do ss=ss+string(image(kx,ky),format='(i4)')
            printf,1,string(ky,format='(i3)')+strtrim(ss,2)
        end
  if nx2 lt nxend then begin
    nx1=nx1+18 & nx2=nx2+18 & if nx2 gt nxend then nx2=nxend
      goto,nextpage & end
  close,1 & print,'File "'+dumpname+'" written '
;
  openr,1,dumpname
    while eof(1) eq 0 do begin & readf,1,ss & print,ss & end
  close,1
return
end
;
;-----------------------------------------------------------------------
;
pro plot_window,chw,ntime
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bij
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
;
  if ntime eq 0 then chwin2=ch2 else $
    begin if chw eq ' ' then return & chw=' ' & ch2=chwin2 & end
  mmtitle='Center of '+$
    string(nbeam,format='(i3)')+' Beamlets, '+strtrim(filename,2)
;
  if ch2 eq 'c' then $
  begin & cm=fltarr(nbeam*1,2)
    x1=0.5*xcmpix*xdim & x1=-x2 & y2=0.5*ycmpix*ydim & y1=-y2
    for k=1,nbeam do cm(k,0)=xcmpix*center(k,0)-x2
    for k=1,nbeam do cm(k,1)=ycmpix*center(k,1)-y2
      plot,cm(1:nbeam,0),cm(1:nbeam,1),$
        xrange=[x1,x2],yrange=[y1,y2],xstyle=1,ystyle=1,$
          psym=2,xtitle='x(cm)',ytitle='y(cm)',title=mmtitle
```



```
    for k=1,nbeam do if plot_lets(k,1) eq 1 then $
      xyouts,xcmpix*(center(k,2)-5)-x2,$
          ycmpix*(center(k,4)+center(k,5))/2-y2,string(k,format='(i3)')
  end else if ch2 eq 'o' and pview eq 'c' then begin
    print,'TEST:plot_window  pview=c and ch2=o'
    print,'TEST ---- 5/09/97 ACP doesnt work ???? remove purge abort rid ......
+'
    oplot,center(1:nbeam,0),center(1:nbeam,1),psym=6     ; psym=square
    for k=1,nbeam do if plot_lets(k,1) eq 1 then $
      xyouts,center(k,0),center(k,1),string(k,format='(i3)')
  end else begin
    plot,center(1:nbeam,0),center(1:nbeam,1),$
        xrange=[0,xdim],yrange=[0,ydim],xstyle=1,ystyle=1,$
          psym=2,xtitle='x(pixel)',ytitle='y(pixel)',title=mmtitle
    for k=1,nbeam do if plot_lets(k,1) eq 1 then $
      xyouts,center(k,2)-5,(center(k,4)+center(k,5))/2,string(k,format='(i3)')
  end
;
  if ch2 eq 'c' or ch2 eq 'n' then return
;
  wx=intarr(5) & wy=wx & x=xwindow & y=ywindow
  wx(0)=x & wx(1)=-x & wx(2)=-x & wx(3)=x & wx(4)=x
  wy(0)=y & wy(1)=y & wy(2)=-y & wy(3)=-y & wy(4)=y
  for k=1,nbeam do if plot_lets(k,1) eq 1 then $
    oplot,wx+center(k,0),wy+center(k,1)
;
return
end
;
;-----------------------------------------------------------------------
;
pro next_file
;
common com1,filename,dumpname,printname,dataname,title
common com15,version,comhead,comdata,moments
;
  pool='0123456789abcdefghijklmnopqrstuvwxyz'
  head='' & suff=' '
  nn=strpos(filename,'.ras')
  head=strmid(filename,0,nn-2)
  suff=strmid(filename,nn-2,2)
  nc1=strpos(pool,strmid(suff,0,1)) & nc2=strpos(pool,strmid(suff,1,1))
  nc2=nc2+1 & if nc2 ge strlen(pool) then begin & nc2=0 & nc1=nc1+1 & end
    ss=strmid(pool,nc1,1)+strmid(pool,nc2,1)
      filename=head+ss+'.ras'
  comdata(0)=filename+' ('+dataname+')     '+version
  ; 5/12/97 comdata(0)=filename+'    '+version
;
return
end
;
;-----------------------------------------------------------------------
;
pro readdata,cerror
common com1,filename,dumpname,printname,dataname,title
common com3,ssin,pview,ch2
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com15,version,comhead,comdata,moments
;
  num=intarr(4) & ss='' & root=''
again:
  if comdata(15) eq '~gbi/' then $
    begin
      ch2=' ' & set_path_name
      if strlen(comdata(15)) gt 0 then begin & ch2='s' & set_path_name & end
    end
  if cerror eq 'R' then return
;
  print,'F)orce raw mode, N)ext_file, Q)uit, R)eturn, X)restore from saved file
+'
  read,'  Enter *.RAS file <LF '+filename+'> f,n,q,r,x)restore ==> ',ss
```

```
  if (ss eq 'q') or (ss eq 'quit') then $
            begin & cerror='q' & return & end
  if (ss eq 'x') then $
            begin & cerror='R' & return & end
  if (ss eq 'return') or (ss eq 'r') then $
            begin
              if (cerror eq '&') then goto,again $
                else begin & cerror='r' & return & end & end
  if (ss eq 'raw') or (ss eq 'f') then $
            begin & cerror='t' & print,'Image RAW mode' & goto,again & end
  if (ss eq 'n') then $
            begin & cerror=' '
              next_file & print,'Processing file '+filename
            end $
  else begin
      cerror=' '
      if strlen(ss) gt 0 then filename=ss
        comdata(0)=filename+' ('+dataname+')     '+version
        ;  5/12/97 ???? comdata(0)=filename+'    '+version
      read,'Enter output files root name, <LF uses temp>          ==> ',root
        if strlen(root) gt 0 then root=strtrim(root,2) else root='temp'
          dumpname=root+'.dump' & printname=root+'.lpr'
      ch='&'
      data_input,ch
  end
;
return
end
;
;-----------------------------------------------------------------------
;
pro plot_brightness,ntime
;
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelsc1
common com20,scales
;
  if (nbeam eq 0) then begin & print,'RAW MODE' & return & end
  if nmask eq 0 then begin & print,'Must generate mask.' & return & end
  if brightness eq 0 then begin & print,'Brightness=0' & return & end
  if ntime eq 0 then $
    read,'Plot vs r)adius, n)beamlet, b)oth <r,n,b,B_spc_nospc> ',chbsv
  if strlen(chbsv) eq 0 then chbsv='B'
  if (chbsv eq 'B') or (chbsv eq 'b') then !p.multi=[0,2]
  if (chbsv eq 'B') then begin
    if nspcj eq 0 then spcj=1 else spcj=0 & end
  if nspcj eq 0 then begin & ssym1=4 & ssym2=2 & end $
                 else begin & ssym1=2 & ssym2=4 & end
  jx=2.0*current/(3.14159le-6*emitx*betagamma)^2
  jy=2.0*current/(3.14159le-6*emity*betagamma)^2
  jxy=sqrt(jx*jy) & jav=float(strmid(comdata(13),35,10))
  swtype=0
;
; get plot scales
  if scales(4) gt 0 then begin
    rm=scales(5) & y1=scales(6) & y2=scales(7) & y1=10^y1 & y2=10^y2
  end else begin
    if mask(0,2) gt 0 then rm=mask(0,2) else rm=max(jradius)
    y1=min(where(bjj(*,nspcj) gt 0))
    y2=max(bjj(*,nspcj)) & if (y1 eq 0) then y1=0.1*y2
    if (jx lt y1) then y1=jx & if (jy lt y1) then y1=jy
    if y1 lt y2/1000 then y1=y2/1000
    swtype=0 & end
;
; order in accending radius
;
```



```
  if (chbsv eq 'r') or (chbsv eq 'b') or (chbsv eq 'B') then begin
    plot_io,jradius,bjj(*,nspcj),xrange=[0,rm],yrange=[y1,y2],$
      xstyle=swtype,ystyle=swtype,$
        xtitle='radius(cm)',ytitle='Brightness (A/(m-rad)^2)',$
          title=filename,/nodata
    for k=0,nbeam-1 do if plot_lets(k+1,1) eq 1 then $
      oplot,jradius(k:k),bjj(k:k,nspcj),psym=ssym1
    if chbsv eq 'B' then for k=0,nbeam-1 do if plot_lets(k+1,1) eq 1 then $
      oplot,jradius(k:k),bjj(k:k,spcj),psym=ssym2
    xyouts,rm,jx,'<x>' & xyouts,rm,jy,'<y>' & xyouts,rm,brightness,'<J>'
    xyouts,rm,jxy,'<xy>' & xyouts,rm,jav,'<A>'
  end
  if (chbsv eq 'n') or (chbsv eq 'b') or (chbsv eq 'B') then begin
    nn=indgen(nbeam+1)
    plot_io,jradius,bjj(*,nspcj),xrange=[0,nbeam],yrange=[y1,y2],$
      xstyle=swtype,ystyle=swtype,xtitle='Beamlet Number',$
        ytitle='Brightness (A/(m-rad)^2)',title=filename,/nodata
    for k=0,nbeam-1 do if jradius(k) gt 0 then $
      if plot_lets(k+1,1) eq 1 then oplot,nn(k+1:k+1),bjj(k:k,nspcj),psym=ssym1
    if chbsv eq 'B' then for k=0,nbeam-1 do if jradius(k) gt 0 then $
      if plot_lets(k+1,1) eq 1 then oplot,nn(k+1:k+1),bjj(k:k,spcj),psym=ssym2
    xyouts,nbeam-2,jx,'<x>' & xyouts,nbeam-2,jy,'<y>'
    xyouts,nbeam-2,brightness,'<J>'
    xyouts,nbeam-2,jxy,'<xy>' &  xyouts,nbeam-2,jav,'<A>'
  end
  if ntime eq 0 then ypos=0.90 else ypos=-0.03
  xyouts,0.15,ypos,'Emittance '+comdata(8)+' mm-mr    '+comdata(0),/norm
  xyouts,0.15,ypos-0.03,'Amps,Brightness '+comdata(9)+' A/(m-rad)^2',/norm
  xyouts,0.15,ypos-0.06,'J(x),J(y),J(xy),J(Av)='+comdata(13),/norm
  if nspcj eq 1 then ss='Corrected' else ss='Uncorrected'
  xyouts,0.15,ypos-0.09,comhead(10)+comdata(10)+'  Space charge '+ss,/norm
  if ntime eq 1 then ss=comhead(14)+comdata(14)+'  ' else ss=''
  ss=ss+'Mask rotation '+comdata(19)+' deg' & xyouts,0.15,ypos-0.12,ss,/norm
  !p.multi=0
return
end
;
;-----------------------------------------------------------------------
;
pro data_input,redo
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
;
ch=' ' & swx='' & ss='' & num=fltarr(4)
if redo eq '&' then begin & ch='a' & goto,doe & end
;
redo=' '
print,title
a='CHANGE: A)ll, C)urrnt, D)rft, E)nrgy, L)evl, M)ask, P)ixel, R, W)indow,'
print,a+'J, I)mge'
  okchar='A,C,D,E,I,L,M,P,R,W,J'
    read,'                                                  <'+okchar+'>  ==> ',ch
      if strlen(ch) eq 0 then return
        okchar=strlowcase(okchar) & if strpos(okchar,ch) lt 0 then return
  if ch eq 'j' then goto,doj else if ch eq 'i' then goto,doi else $
  if ch eq 'a' then goto,doa else if ch eq 'c' then goto,doc else $
  if ch eq 'd' then goto,dod else if ch eq 'e' then goto,doe else $
  if ch eq 'l' then goto,dol else if ch eq 'm' then goto,dom else $
  if ch eq 'p' then goto,dop else if ch eq 'w' then goto,dow else $
  if ch eq 'r' then goto,dor
;
doe: ss=' ' & sm=' '
    read,'Enter beam Energy  ( MeV) '+strmid(comdata(4),0,8)+'  ==> ',ss
    read,'Enter rest mass    ( MeV) LF uses:'+string(restm)+'  ==> ',sm
    if strlen(sm) gt 0 then restm=float(sm)
    if strlen(ss) gt 0 then begin & redo='j'
```

```
      energy=float(ss) & r=energy/restm
      if r gt 0.1 then begin & gamma=1+r^2 & betagamma=sqrt(gamma^2-1) & end $
        else begin & b2=2*r & betagamma=sqrt(b2/(1-b2)) & end
      comdata(4)='['+string(energy,format='(f8.4)')+',+$
                    string(betagamma,format='(f8.4)')+']    '+$
                    string(restm,format='(f13.4)') & end
    if ch ne 'a' then return
doi: if redo ne 'i' then begin & imin=min(image) & imax=max(image)
       sumx=total(image(*,0))/xdim+total(image(*,ydim-1))/xdim
       sumy=total(image(0,*))/ydim+total(image(xdim-1,*))/ydim
       background=fix((sumx+sumy)/4.0)
       comdata(2)='['+string(imin,format='(i3)')+',+$
                 string(imax,format='(i3)')+'] '+string(background,format='(i3)')
       print,comhead(2)+comdata(2) & end
    if ch ne 'a' then return
doj: read,'Apply space charge correction to brightness <y,n> ',ss
    if strlen(ss) gt 0 then begin
      if ss eq 'y' then nspcj=1 else nspcj=0 & end
    if ch ne 'a' then return
doc: read,'Enter beam Current (Amps) '+strmid(comdata(9),0,8)+'  ==> ',ss
    if strlen(ss) gt 0 then begin & old=current & current=float(ss) & redo='j'
      if old gt 0 then brightness=current*brightness/old else brightness=0
     read,'Enter Mask (Passed Amps)     '+strmid(comdata(9),10,8)+'==> ',ss
      if strlen(ss) gt 0 then ipass(0)=float(ss)
     read,'Through how many holes          '+strmid(comdata(9),19,4)+'  ==> ',ss
      if strlen(ss) gt 0 then ipass(1)=float(ss)
       ss='['+string(current,format='(f8.2)')+'('+$
          string(ipass(0),format='(f6.3)')+'/'+$
            string(ipass(1),format='(f4.0)')+'),'+$
              string(brightness,format='(e10.2)')
       comdata(9)=ss+strmid(comdata(9),34,11)
     end
    if ch ne 'a' then return
dow: read,'Enter window x,y(pixels ) '+comdata(5)+' ==> ',ss
    if strlen(ss) gt 0 then begin & redo='y'
      decode,ss,num,n & if n gt 0 then begin
        comdata(5)='['+string(num(0),format='(i3)')+',+$
                      string(num(1),format='(i3)')+']'
       xwindow=fix(num(0)/2) & ywindow=fix(num(1)/2)
       window=fix((xwindow+ywindow)/2)
    end & end
    if ch ne 'a' then return
dol: read,'Enter "level" e.i. 10%     '+comdata(10)+'       ==> ',ss
    if strlen(ss) gt 0 then begin & level=fix(ss) & redo='y'
      comdata(10)=string(level,format='(i4)') & end
    if ch ne 'a' then return
dod: read,'Enter drift L(m) to image '+comdata(7)+'  ==> ',ss
    if strlen(ss) gt 0 then begin & drift=float(ss) & redo='j'
      comdata(7)=string(drift,format='(f8.4)') & end
    if ch ne 'a' then return
dom: read,'Enter mask rhole,spacing and thickness(cm) '+comdata(6)+'==> ',ss
    if strlen(ss) gt 0 then begin & redo='j'
      decode,ss,num,n & if n gt 0 then begin
        rhole=num(0) & separation=num(1) & thickness=num(2)
        comdata(6)='['+string(rhole,format='(f8.4)')+',+$
                      string(separation,format='(f8.4)')+',+$
                      string(thickness,format='(f8.4)')+'] H' & end & end
    if ch ne 'a' then return
dop: read,'Enter x,y cm/pixel '+comdata(3)+' ==> ',ss
    if strlen(ss) gt 0 then begin & redo='y'
      decode,ss,num,n & if n gt 0 then begin & xcmpix=num(0) & ycmpix=num(1)
         comdata(3)='['+string(xcmpix,format='(f7.5)')+',+$
                       string(ycmpix,format='(f7.5)')+']' & end & end
return
dor: read,'Enter beam size on mask R(cm)       LF(auto calc) ==> ',ss
    if strlen(ss) gt 0 then begin & redo='j'
      decode,ss,num,n & if n gt 0 then mask(0,2)=num(0)
      if mask(0,2) gt 0 then begin
        comdata(14)='['+string(num(0),format='(f8.3)')+$
                                strmid(comdata(14),9,strlen(comdata(14))-10)
        comdata(13)=' '
      end
    end
```



```
return
end
;
;-----------------------------------------------------------------------
;
pro move_curser
;
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com12,nb,type_slice,xslice,yslice,prange,chsave1,chsave2,chsave3
common com13,sub_range
common com15,version,comhead,comdata,moments
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
;
  cc=' ' & zero=0.0 & xwin=125 & ywin=125
  x=xpix0-pixsize & y=ypix0-pixsize & x1=x & y1=y
  print,'TEST:move_curser xpix0,ypix0=',xpix0,ypix0
  print,'TEST:move_curser pixsize=',pixsize
  print,'  Click left button (lower left) right button (upper right) of window'
  print,'  Use buttons L(eft reads, C)enter of image, R)ight returns'
  print,'TEST:move_curser  xwcent,ywcent=',xwcent,ywcent ; ACP 4/30/97 ????
;
  tvcrs,xwin,ywin
;
  lwindo=strpos('w',pview)
  if (strpos('cw',pview) ge 0) and (xwcent gt 0) then docent='t' else docent='f
'
  print,'    pview='+pview+'    docent='+docent+$
        '  subcont='+string(sub_range.subcont(0))+string(sub_range.subcont(2))
;
 ix0=sub_range.subcont(0)
 iy0=sub_range.subcont(2)
 print,'     x(pix)    y(pix)     x(cm)          y(cm)     intensity'
 x1=0 & y1=0
    while (!err lt 2) do $
      begin cursor,x,y,/data
        if x ne x1 and y ne y1 then begin
          x1=x & y1=y
          if pview eq 'c' then begin
            x=ix0+fix(x/xcmpix) & y=iy0+fix(y/ycmpix)
              intensity=image(x,y)
                print,x,y,x1,y1,     ',intensity
          end else begin
            xcm=x*xcmpix & ycm=y*ycmpix
              print,fix(x),fix(y),xcm,ycm
          end
        end
    end
  tvcrs,200,-350
return
end
;
;-----------------------------------------------------------------------
;
pro listdata,cmd
common com1,filename,dumpname,printname,datename,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
;
ch=' ' & ss=''
  if cmd ne ' ' then ch=cmd else $
    read,'List A)ll,B)eam(),C)en(),D)ata,E)mit(),M)ask(),P)hz(),W)ndo '+$
      '<a,b,c,d,e,m,p,w> ',ch
```

```
3912    if ch eq 'b' then goto,listb else if ch eq 'c' then goto,listc else $
3913    if ch eq 'd' then goto,listd else if ch eq 'e' then goto,liste else $
3914    if ch eq 'm' then goto,listm else if ch eq 'p' then goto,listp else $
3915    if ch eq 'w' then goto,listw
3916  ;
3917  listd:
3918    print,comdata(17)
3919    print,'------------------------------------------------------------'
3920      for k=0,14 do print,string(k,format='(i2)')+' '+comhead(k)+comdata(k)
3921      k=19 & print,string(k,format='(i2)')+' '+comhead(k)+comdata(k)
3922    if ch eq 'd' then return
3923    read,'Continue <,n> ',ch & if ch eq 'n' then return
3924  ;
3925  listc:
3926    print,'------------------------------------------------------------'
3927    if strmid(ssin,2,1) eq 'l' then begin
3928      print,'Beamlet centers and x,y range (pixels)'
3929      print,'    k    xcen ycen xmin xmax ymin ymax'+$
3930            '    k    xcen ycen xmin xmax ymin ymax'
3931      nbh=nbeam/2
3932      for k=1,nbh do begin
3933        if plot_lets(k,1) eq 1 then ksw=k else ksw=-k
3934        if plot_lets(k+nbh,1) eq 1 then jsw=k+nbh else jsw=-k-nbh
3935        print,format='(i4,1x,3(2x,2i4),2x,i4,1x,3(2x,2i4))',$
3936           ksw,center(k,0),center(k,1),$
3937                center(k,2),center(k,3),center(k,4),center(k,5),$
3938           jsw,center(k+nbh,0),center(k+nbh,1),$
3939                center(k+nbh,2),center(k+nbh,3),center(k+nbh,4),center(k+nbh,5)
3940      end
3941      if ch eq 'c' then return
3942    end else begin
3943      print,'Beamlet centers (pixels)'
3944      print,'     k xpx ypx     k xpx ypx     k xpx ypx'+$
3945            '     k xpx ypx     k xpx ypx     k xpx ypx'
3946      for k=1,nbeam do begin
3947        if plot_lets(k,1) eq 1 then ksw=k else ksw=-k
3948          ss=ss+string(ksw,format='(i5)')+$
3949            string(center(k,0),format='(i4)')+string(center(k,1),format='(i4)')
3950              if strlen(ss) gt 65 then begin & print,ss & ss='' & end
3951      end & print,ss & if ch eq 'c' then return
3952    end
3953    read,'Continue <,n> ',ch & if ch eq 'n' then return
3954  ;
3955  listb: ; beam is data array
3956    print,'Total_int(pixels) '+$
3957          'is value in beamlet above background for that beamlet.'
3958    print,' ' & print,'Beamletts DATA[*]' & lines=0
3959    if nbeam eq 0 then return
3960    for k=1,nbeam do begin
3961      if lines eq 0 then begin
3962      print,'  n  peak   xmin    xmax  x(abs)   ymin    ymax   y(abs)  '+$
3963                                                          'Total_int   backg'
3964      print,'    pixels  (cm)    (cm)   (cm)    (cm)    (cm)    (cm)    '+$
3965                                                          'pixels      pixels'
3966      end
3967      if plot_lets(k,1) eq 1 then ksw=k else ksw=-k
3968      ss=string(data(k,0),format='(i4)')
3969        for j=1,6 do ss=ss+string(data(k,j),format='(f8.4)')
3970      ss=ss+string(data(k,7),format='(e10.3)')+string(data(k,8),format='(f7.2)')
3971            print,string(ksw,format='(i4)')+' '+ss
3972      lines=lines+1
3973        if lines eq 25 then begin & read,'Continue <,n> ',ch & lines=0
3974          if ch eq 'n' then return & end
3975    end & if ch eq 'd' then return
3976    read,'Continue <,n> ',ch & if ch eq 'n' then return
3977    if ch eq 'd' then return
3978  ;
3979  liste: print,'BEAMLETS' & lines=0 & isum=0
3980    for k=1,nbeam do begin
3981      if lines eq 0 then begin
3982      print,'    k     Emitx      Emity'+$
3983            '    Radius   Current    Brightness   SPC_corrected'
3984      print,'          m-rad      m-rad'+$
```



```
3985            '     cm       Amps       A/(m-rad)^2    A/(m-rad)^2'
3986      end
3987      if plot_lets(k,1) eq 1 then ksw=k else ksw=-k & isum=isum+ibeamn(k-1)
3988      ss='' & ss=string(ksw,format='(i4)')
3989      ss=ss+string(emitn(k,0),format='(e11.3)')
3990      ss=ss+string(emitn(k,1),format='(e11.3)')
3991      ss=ss+string(radius(k-1),format='(f8.4)')
3992      ss=ss+string(ibeamn(k-1),format='(f8.4)')
3993      ss=ss+string(bjj(k-1,0),format='(e14.3)')
3994      ss=ss+string(bjj(k-1,1),format='(e14.3)')
3995      print,ss
3996      lines=lines+1
3997        if lines eq 25 then begin & read,'Continue <,n> ',ch & lines=0
3998          if ch eq 'n' then return & end
3999    end
4000    print,'Sum of currents is '+string(isum,format='(f10.4)')+' Amperes'
4001    if ch eq 'e' then return
4002    read,'Continue <,n> ',ch & if ch eq 'n' then return
4003  ;
4004  listp: print,'Phase[*]' & lines=0
4005    for k=1,nbeam do begin
4006      if lines eq 0 then begin
4007        print,'    k   X_ABS      xpmin     xpmax     Y_ABS    ypmin    ypmax'
4008        print,'         (cm)      (rad)     (rad)      (cm)    (rad)    (rad)'
4009      end
4010      if plot_lets(k,1) eq 1 then ksw=k else ksw=-k
4011      ss='' & ss=string(ksw,format='(i4)')
4012        for j=0,5 do ss=ss+string(phase(k,j),format='(f9.5)')
4013          print,ss
4014      lines=lines+1
4015        if lines eq 25 then begin & read,'Continue <,n> ',ch & lines=0
4016          if ch eq 'n' then return & end
4017    end
4018      print,'..................'
4019      ss='' & ss=string(maxdata-3,format='(i4)')
4020        for j=0,1 do ss=ss+string(phase(maxdata-2,j),format='(f9.5)')
4021        for j=2,5 do ss=ss+string(phase(maxdata-3,j),format='(f9.5)')
4022          print,ss
4023      ss='' & ss=string(maxdata-1,format='(i4)')
4024        for j=0,5 do ss=ss+string(phase(maxdata-1,j),format='(f9.5)')
4025          print,ss
4026    if ch eq 'p' then return
4027    read,'Continue <,n> ',ch & if ch eq 'n' then return
4028  ;
4029  listm: print,'Mask[*]'
4030    nbh=nmask/2
4031    print,'    k      x(cm)     y(cm)   (radius)      ',$
4032          '    k      x(cm)     y(cm)   (radius)'
4033    ix=0 & iy=0
4034    for k=1,nbh do $
4035      print,format='(2(i5,f9.4,f9.4,2x,1h(,f7.3,1h),4x))',$
4036            k,mask(k,0),mask(k,1),mask(k,2),$
4037            k+nbh,mask(k+nbh,0),mask(k+nbh,1),mask(k+nbh,2)
4038    if 2*nbh lt nmask then $
4039      print,format='(38x,i5,f9.4,f9.4,2x,1h(,f7.3,1h))',$
4040            nmask,mask(nmask,0),mask(nmask,1),mask(nmask,2)
4041    if ch eq 'm' then return
4042    read,'Continue <,n> ',ch & if ch eq 'n' then return
4043  ;
4044  listw: print,'Window[*]'
4045    nbh=nbeam/2
4046    print,'    k    xcen ycen xmin xmax ymin ymax       ',$
4047          '    k    xcen ycen xmin xmax ymin ymax'
4048    for k=1,nbh do $
4049      print,format=$
4050        '(i3,3x,1h(,i3,1h,,i3,1h),4i5,6x,i3,3x,1h(,i3,1h,,i3,1h),4i5)',$
4051        k,center(k,0),center(k,1),center(k,0)-xwindow,center(k,0)+xwindow,$
4052          center(k,1)-ywindow,center(k,1)+ywindow,$
4053        k+nbh,center(k+nbh,0),center(k+nbh,1),$
4054          center(k+nbh,0)-xwindow,center(k+nbh,0)+xwindow,$
4055            center(k+nbh,1)-ywindow,center(k+nbh,1)+ywindow
4056    if 2*nbh lt nbeam then $
4057      print,format='(41x,i3,3x,1h(,i3,1h,,i3,1h),4i5)',$
```

```
      nbeam-1,center(nbeam-1,0),center(nbeam-1,1),$
        center(nbeam-1,0)-xwindow,center(nbeam-1,0)+xwindow,$
          center(nbeam-1,1)-ywindow,center(nbeam-1,1)+ywindow
;
return
end
;
;-----------------------------------------------------------------------
;
pro summaryr,ntime
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssln,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
;
  !p.multi=[0,2,0,0,0]
;
; plot 1, TOTAL INTENSITY ------------------------
  yy=data(1:nbeam,7)
  plot,jradius,yy,psym=2,$
    xtitle='Radius(cm)',ytitle='Pixels',$
      title='Total Intensity'+string(data[0,7],format='(f10.0)')
  ss='Ave:'+string(total(data(1:nbeam,7))/nbeam,format='(f8.1)')
  xyouts,0.15,0.25,ss,/norm
  xyouts,0.15,0.93,'Above each beamlets BG.',/norm
  xyouts,0.15,-0.09,filename,/norm
  xyouts,0.15,-0.12,title,/norm
;
  del=0.2*separation
  nrings=fix((max(jradius)+del)/separation)
  nn=intarr(nrings+1) & xx=fltarr(nrings+1) & aa=xx & aa1=xx & aa2=xx
  for k=0,nrings do aa1(k)=999999.0
  for k=0,nbeam-1 do begin
    ir=fix((del+jradius(k))/separation)
      aa(ir)=aa(ir)+data(k+1,7) & nn(ir)=nn(ir)+1
      if data(k+1,7) lt aa1(ir) then aa1(ir)=data(k+1,7)
      if data(k+1,7) gt aa2(ir) then aa2(ir)=data(k+1,7)
  end
  r=0
  for k=0,nrings do begin
    xx(k)=r & r=r+separation & if nn(k) gt 0 then aa(k)=aa(k)/nn(k)
  end
  oplot,xx,aa
  for k=0,nrings do plots,[xx(k),xx(k)],[aa1(k),aa2(k)]
;
; plot 2, PEAK INTENSITY AND BACKGROUND ---------------
  yy=data(1:nbeam,0)
  plot,jradius,yy,psym=2,$
    xtitle='Radius(cm)',ytitle='Pixels',title='Peak Int & Background'
  nn=intarr(nrings+1) & xx=fltarr(nrings+1) & aa=xx & aa1=xx & aa2=xx
  for k=0,nrings do aa1(k)=999999.0
  for k=0,nbeam-1 do begin
    ir=fix((del+jradius(k))/separation)
      aa(ir)=aa(ir)+data(k+1,0) & nn(ir)=nn(ir)+1
      if data(k+1,0) lt aa1(ir) then aa1(ir)=data(k+1,0)
      if data(k+1,0) gt aa2(ir) then aa2(ir)=data(k+1,0)
  end
  r=0
  for k=0,nrings do begin
    xx(k)=r & r=r+separation & if nn(k) gt 0 then aa(k)=aa(k)/nn(k)
  end
  oplot,xx,aa
  for k=0,nrings do plots,[xx(k),xx(k)],[aa1(k),aa2(k)]
;
; now the background
  oplot,jradius,data(1:nbeam,8),psym=4
```



```
  nn=intarr(nrings+1) & xx=fltarr(nrings+1) & aa=xx
  for k=0,nbeam-1 do begin
    ir=fix((del+jradius(k))/separation)
      aa(ir)=aa(ir)+data(k+1,8) & nn(ir)=nn(ir)+1
  end
  r=0
  for k=0,nrings do begin
    xx(k)=r & r=r+separation & if nn(k) gt 0 then aa(k)=aa(k)/nn(k)
  end
  oplot,xx,aa
;
  plots,[0,xx(nrings)],[background,background]
  ss='Ave:'+string(total(data(1:nbeam,0))/nbeam,format='(f8.2)')
  xyouts,0.65,0.28,ss,/norm
  ss='Ave:'+string(total(data(1:nbeam,8))/nbeam,format='(f8.2)')
  xyouts,0.65,0.26,ss,/norm
;
  !p.multi=0 & ss=''
return
end
;
;-----------------------------------------------------------------------
;
pro summary,ntime
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssln,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com15,version,comhead,comdata,moments
;
  if (nbeam eq 0) then begin & print,'RAW MODE' & return & end
;
  !p.multi=[0,2,2,0,0]
;
; plot 1, TOTAL INTENSITY ------------------------
  beamn=findgen(nbeam+2)
  histo=data(0:nbeam+1,7) & histo(0)=0 & histo(nbeam+1)=0
  histx=histo & histy=histo
  plot,beamn,histo,psym=10,$
    xtitle='Beamlet',ytitle='Pixels',$
      title='Total Intensity'+string(data(0,7),format='(f10.0)')
  ss='Ave:'+string(total(data(1:nbeam,7))/nbeam,format='(f8.1)')
  xyouts,0.15,0.65,ss,/norm
  if ntime eq 0 then iy=0.94 else iy=0.90
  xyouts,0.15,iy,'Above each beamlets BG',/norm
;
; plot 2, PEAK INTENSITY AND BACKGROUND ---------------
  histo(1:nbeam)=data(1:nbeam,0)
  plot,beamn,histo,psym=10,$
    xtitle='Beamlet',ytitle='Pixels',title='Peak Int & Background'
  oplot,beamn,data(0:nbeam,8),psym=10
  yy=intarr(2) & xx=yy
  xx(0)=0 & xx(1)=nbeam & yy(0)=background & yy(1)=background
  oplot,xx,yy
  ss='Ave:'+string(total(data(1:nbeam,0))/nbeam,format='(f8.2)')
  xyouts,0.65,0.68,ss,/norm
  ss='Ave:'+string(total(data(1:nbeam,8))/nbeam,format='(f8.2)')
  xyouts,0.65,0.66,ss,/norm
;
; plot 3, BEAMLET RADIUS ------------------------
  histx(1:nbeam)=0.5*(data(1:nbeam,2)-data(1:nbeam,1))
  histy(1:nbeam)=0.5*(data(1:nbeam,5)-data(1:nbeam,4))
  mcm=max(bradius) & if max(histx) gt mcm then mcm=max(histx)
    if max(histy) gt mcm then mcm=max(histy)
  histo(0:nbeam+1)=rhole
  plot,beamn,histo,yrange=[0,mcm],ystyle=1,$
    xtitle='Beamlet',ytitle='cm',title='Beamlet Radius'
```

```
  oplot,beamn,histx,linestyle=1,psym=10
  oplot,beamn,histy,linestyle=2,psym=10
  xy=0.25*total(data(1:nbeam,2)-data(1:nbeam,1))/nbeam+$
     0.25*total(data(1:nbeam,5)-data(1:nbeam,4))/nbeam
  ss='Ave:'+string(xy,format='(f8.4)')
  xyouts,0.15,0.15,ss,/norm
;
; plot 4, BEAMLET ANGULAR DIVERGENCE ------------------------
; phase(,*)=XABS,xpmin,xpmax,YABS,ypmin,ypmax
; phase(maxdata-1,*)=s11,s12,s22,s33,s34,s44 ; phase(maxdata-2,*)=r12,r14
  histx(1:nbeam)=1000*(phase(1:nbeam,2)-phase(1:nbeam,1))
  histy(1:nbeam)=1000*(phase(1:nbeam,5)-phase(1:nbeam,4))
  xpmin=min(histx(1:nbeam)) & xpmax=max(histx(1:nbeam))
  ypmin=min(histy(1:nbeam)) & ypmax=max(histy(1:nbeam))
  if ypmin lt xpmin then pmin=ypmin else pmin=xpmin
  if ypmax gt xpmax then pmax=ypmax else pmax=xpmax
  pmin=0
  plot,beamn,phase(1:nbeam,1),yrange=[pmin,pmax],$
    xtitle='Beamlet',ytitle='mrad',title='X,Y Angular Spread (max-min)',/nodata
  oplot,beamn,histx,linestyle=1,psym=10
  oplot,beamn,histy,linestyle=2,psym=10
  xy=500*total(phase(1:nbeam,2)-phase(1:nbeam,1))/nbeam+$
     500*total(phase(1:nbeam,5)-phase(1:nbeam,4))/nbeam
  ss='Ave:'+string(xy,format='(f8.3)')
  xyouts,0.65,0.15,ss,/norm
;
  !p.multi=0 & ss=''
  ss=filename+' '+strmid(comdata(9),2,20)+' (Amp), '+$
     string(energy,format='(f7.3)')+'(MeV),  '+$
       string(level,format='(i3)')+' % level'
         xyouts,0.10,0.50,ss,/norm
  if ntime eq 1 then begin & yyx=-0.10
    xyouts,0.10,-0.10,'-----------------------------------------',/norm
      for k=0,14 do begin & yyx=yyx-0.03
        xyouts,0.10,yyx,string(k,format='(i2)')+' '+comhead(k)+comdata(k),/norm
    end
    k=19 & yyx=yyx-0.03
      xyouts,0.10,yyx,string(k,format='(i2)')+' '+comhead(k)+comdata(k),/norm
  end
return
end
;
;-----------------------------------------------------------------------
;
pro fit_gauss
goto,skipit
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhola,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com12,nb,type_slice,xslice,yslice,prange,chsave1,chsave2,chsave3
common com13,sub_range
common com15,version,comhead,comdata,moments
common com18,kmovie,nslice,slices
common com22,flag,numpix
;
width=2*xwindow+1 & width=fix(width)
;
if flag ne 0 then goto, skipover
For j=1,2 do begin
For j=1,nbeam do begin
   flag=1
   gx=fltarr(width) & a=fltarr(6) & gx=0.0 & a=0.0
   gy=fltarr(width) & b=fltarr(6) & gy=0.0 & b=0.0
    nx=center(j,0) & ny=center(j,1)
    nx1=nx-xwindow  & if nx1 lt 1 then nx1=1
    nx2=nx+xwindow  & if nx2 gt xdim then nx1=xdim-1
    ny1=ny-ywindow  & if ny1 lt 1 then ny1=1;    ny2=ny+ywindow  & if ny2 gt yd
```



```
    +im then ny1=ydim-1
      nx1=fix(nx1) & nx2=fix(nx2) & ny1=fix(ny1) & ny2=fix(ny2)
  xpix=intarr(width) & ypix=intarr(width)
  for i=0,width-1 do $ xpix(i)=nx1+i
  for i=0,width-1 do $ ypix(i)=ny1+i

print,'beamlet #',j

peak=max(image(nx1:nx2,ny1:ny2))
cutoff=.01*level*peak

  centerx=where(image(nx1:nx2,ny) eq peak)
  centerx=centerx+nx1
  centerx=fix(centerx)
  centery=where(image(nx,ny1:ny2) eq peak)
  centery=fix(centery)
  centery=centery+ny1
lowflag=0 & k=0
  for k=centerx(0),nx1,-1 do begin
    if (lowflag eq 1) then begin
        image(k,ny)=background
    endif else begin
        if (image(k,ny) lt cutoff) then begin
            image(k,ny)=background
            lowflag=1
        endif
    endelse
  end
lowflag=0
  for k=centerx(0),nx2 do begin
    if (lowflag eq 1) then begin
        image(k,ny)=background
    endif else begin
        if (image(k,ny) lt cutoff) then begin
            image(k,ny)=background
            lowflag=1
        endif
    endelse
  end

lowflag=0
for k=centery(0),ny2 do begin
    if (lowflag eq 1) then begin
        image(nx,k)=background
    endif else begin
        if (image(nx,k) lt cutoff) then begin
            image(nx,k)=background
            lowflag=1
        endif
    endelse
  end

lowflag=0
for k=centery(0),ny1,-1 do begin
    if (lowflag eq 1) then begin
        image(nx,k)=background
    endif else begin
        if (image(nx,k) lt cutoff) then begin
            image(nx,k)=background
            lowflag=1
        endif
    endelse
  end
xpix=where(image(nx1:nx2,ny) ge 0)
xpix=xpix+nx1
xpix=fix(xpix)
ypix=where(image(nx,ny1:ny2) ge 0)
ypix=ypix+ny1
ypix=fix(ypix)

gx=gaussfit(xpix,image(nx1:nx2,ny),a)
gy=gaussfit(ypix,image(nx,ny1:ny2),b)
gx=fix(gx) & gy=fix(gy)
```

```
image(nx1:nx2,ny)=gx
image(nx,ny1:ny2)=gy
   endfor
skipit:
skipover:
return
end
;-----------------------------------------------------------------------
;
pro analyze,ntime
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,lmswch,image,lobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com12,nb,type_slice,xslice,yslice,prange,chsave1,chsave2,chsave3
common com13,sub_range
common com15,version,comhead,comdata,moments
common com18,kmovie,nslice,slices
;
;loadct, 13
gx=fltarr(22) & a=fltarr(6) & gx=0.0 & a=0.0
gy=fltarr(22) & b=fltarr(6) & gy=0.0 & b=0.0
xg=fltarr(201) & xg(0:200)=0.0
pxg=fltarr(201) & pxg(0:200)=0.0
yg=fltarr(201) & yg(0:200)=0.0
pyg=fltarr(201) & pyg(0:200)=0.0
  if (nbeam eq 0) then begin & print,'RAW MODE' & return & end
  if ntime eq 0 then begin
    chsave1=' '
    read,'Enter beamlet number <1..'+string(nbeam,format='(i3)')+'> ',ss
      nb=fix(ss)
        if nb lt 1 or nb gt nbeam then begin
          print,'Entry outside allowed range' & return & end
    ss='Calculate widths based on'+string(chsave1,format='(a1,$)')
    print,ss
    read,' M)inusBack, S)moothed, MS, R)aw data <m,s,ms,r> ',chsave1
    halfwidths=''
  end else halfwidths=comdata(18)
;
    nx=center(nb,0) & ny=center(nb,1)
    nx1=nx-xwindow  & if nx1 lt 1 then nx1=1
    nx2=nx+xwindow  & if nx2 gt xdim then nx1=xdim-1
    ny1=ny-ywindow  & if ny1 lt 1 then ny1=1
    ny2=ny+ywindow  & if ny2 gt ydim then ny1=ydim-1
    nx1=fix(nx1) & nx2=fix(nx2) & ny1=fix(ny1) & ny2=fix(ny2)
;result=sfit(image(nx1:nx2,ny1:ny2),4)
;surface, result
;goto, skip
    nxm=nx2-nx1+1 & nym=ny2-ny1+1
    halfx=0.5*nxm*xcmpix & xradius=xcmpix*indgen(nxm)-halfx
    halfy=0.5*nym*ycmpix & yradius=ycmpix*indgen(nym)-halfy
    if halfx ge halfy then half=halfx else half=halfy
    linex=image(nx1:nx2,ny) & liney=image(nx,ny1:ny2)
;
;
    peak=data(nb,0) & itotal=data(nb,7)
    back=total(image(nx1,ny1:ny2))/(ny2-ny1+1)+$
         total(image(nx2,ny1:ny2))/(ny2-ny1+1)+$
         total(image(nx1:nx2,ny1))/(nx2-nx1+1)+$
         total(image(nx1:nx2,ny2))/(nx2-nx1+1)
    back=0.25*back
;
;          .--- search from peak to min
;         /                  .--- search from peak to min in other direction.
;        /                  /     4/21/97 TEST ACP
;   i1=float(image(nx1,ny)) & i2=float(image(nx2,ny))
;   xback=i1+(i2-i1)*(indgen(nx2-nx1))/(nx2-nx1)
;   i1=float(image(nx,ny1)) & i2=float(image(nx,ny2))
```



```
;   yback=i1+(i2-i1)*(indgen(ny2-ny1))/(ny2-ny1)
;
;                          .---- subtract tilted local background
;                         /
  if strpos(chsave1,'m') ge 0 then begin
    pekx=max(image(nx1:nx2,ny))
    peky=max(image(nx,ny1:ny2))
    ixa=where(image(nx1:nx2,ny) eq pekx)
    iya=where(image(nx,ny1:ny2) eq peky)
    ix=ixa(0) & iy=iya(0)
;
    i1x=float(min(image(nx1:nx1+ix,ny)))
    i2x=float(min(image(nx1+ix:nx2,ny)))
    ix1a=where(image(nx1:nx1+ix,ny) eq i1x)
    ix2a=where(image(nx1+ix:nx2,ny) eq i2x)
    ix1=float(ix1a(0)) & ix2=float(ix+ix2a(0))
    xas=(i2x-i1x)/(ix2-ix1)
    xback=i1x+xas*(indgen(nx2-nx1))
;
    i1y=float(min(image(nx,ny1:ny1+iy)))
    i2y=float(min(image(nx,ny1+iy:ny2)))
    iy1a=where(image(nx,ny1:ny1+iy) eq i1y)
    iy2a=where(image(nx,ny1+iy:ny2) eq i2y)
    iy1=float(iy1a(0)) & iy2=float(iy+iy2a(0))
    yas=(i2y-i1y)/(iy2-iy1)
    yback=i1y+yas*(indgen(ny2-ny1))
  end
;
;gx=gaussfit(xradius,image(nx1:nx2,ny),a)
;gy=gaussfit(yradius,image(nx,ny1:ny2),b)
;
;if nb eq 4 then print,'4 ',a,b
;print,'image(nx1:nx2,ny) ',image(nx1:nx2,ny)
;result=transpose(image(nx,ny1:ny2))
;print,'image(nx,ny1:ny2) ',result
  plot,xradius,linex,xstyle=1,xrange=[-half,half],$
    xtitle='size(cm)',ytitle='Amplitude(pixels)',title=filename
  oplot,yradius,liney
  plots,[-half,half],[back,back]
;
;    for k=0,200 do begin
;      xg(k)=(2*half*k/200)-half & yg(k)=xg(k)
;      z=(xg(k)-a[1])/a[2] & z=-0.5*z^2
;      pxg(k)=a(0)*exp(z)+a(3)+a(4)*xg(k)+a(5)*xg(k)^2
;      z=(yg(k)-b[1])/b(2) & z=-0.5*z^2
;      pyg(k)=b(0)*exp(z)+b(3)+b(4)*yg(k)+b(5)*yg(k)^2
;    endfor
;  oplot,xg,pxg,linestyle=2,thick=2
;  oplot,yg,pyg,linestyle=2,thick=2
;
  if chsave1 eq 's' then begin
    xyouts,0.8,0.90,'Smoothing=5',/norm
    linex=smooth(linex,5) & liney=smooth(liney,5)
      oplot,xradius,linex,linestyle=1 & oplot,yradius,liney,linestyle=2
    end else $
  if chsave1 eq 'm' then begin
    linex=back+linex-xback & liney=back+liney-yback
      oplot,xradius,linex,linestyle=1 & oplot,yradius,liney,linestyle=4
    end else $
  if chsave1 eq 'ms' then begin
    xyouts,0.8,0.90,'Smoothing=5',/norm
    linex=back+smooth(linex,5)-xback & liney=back+smooth(liney,5)-yback
      oplot,xradius,linex,linestyle=3 & oplot,yradius,liney,linestyle=4
  end
;
  xpeak=max(linex)
  ypeak=max(liney)    ; TEST
  if xpeak gt ypeak then peak=xpeak else peak=ypeak
;
  xyouts,0.2,0.87,'Beamlet '+string(nb,format='(i3)')+$
        '   [+string(nx,format='(i3)')+',+string(ny,format='(i3)')+']'+$
        ' peak'+string(peak,format='(i4)')+$
        '  i_tot'+string(itotal,format='(f10.0)')+' above BG',/norm
```

```
  yy=0.84
  xyouts,0.2,yy,halfwidths,/norm
  xyouts,xradius(nxm/4),linex(nxm/4),'X-cut'
  xyouts,yradius(3*nym/4),liney(3*nym/4),'Y-cut'
  if ntime eq 1 then begin
    if nslice gt 0 then begin
      yy=-0.05
        for k=0,nslice do $
          begin & yy=yy-0.03 & xyouts,0.1,yy,slices(k),/norm & end
    end
    goto,conit
  end
;
  tvcrs,150,150 ; place cursor at pixel location 50,50
  nslice=0
  ierr=0
  save=0
  while (ierr ne 4) do begin
    cursor,x,y,/wait
    ipeak=100.0*(y-back)/(peak-back)
    if ipeak ne save then begin
      save=ipeak
      for k=xwindow,1,-1 do if linex(k) lt y then goto,havex1
        havex1 : xmin=(k-xwindow+(y-linex(k))/(linex(k+1)-linex(k)))*xcmpix
      for k=xwindow,2*xwindow-2 do if linex(k) lt y then goto,havex2
        havex2 : xmax=(k-xwindow-(y-linex(k))/(linex(k-1)-linex(k)))*xcmpix
      for k=ywindow,1,-1 do if liney(k) lt y then goto,havey1
        havey1 : ymin=(k-ywindow+(y-liney(k))/(liney(k+1)-liney(k)))*ycmpix
      for k=ywindow,2*ywindow-2 do if liney(k) lt y then goto,havey2
        havey2 : ymax=(k-ywindow-(y-liney(k))/(liney(k-1)-liney(k)))*ycmpix
      xcm=0.5*(xmax-xmin) & ycm=0.5*(ymax-ymin)
      halfwidths='x,y half widths '+$
        string(xcm,format='(f8.3)')+','+$
        string(ycm,format='(f8.3)')+' cm at '+$
        string(ipeak,format='(f8.3)')+' % amplitude'
      print,halfwidths
      yy=yy-0.02 & xyouts,0.2,yy,halfwidths,/norm
      comdata(18)=halfwidths
      slices(nslice)=halfwidths & if nslice lt 20 then nslice=nslice+1
    end
  end
;
  ierr=0
  tvcrs,100,-50
  chsave2=' ' & read,'Contour this beamlet <y,n,a|abort> ',chsave2
  if chsave2 eq 'a' then abort
;
conit:
  if chsave2 ne 'y' then return
    num=intarr(4)
    num(0)=nx1 & num(1)=nx2 & num(2)=ny1 & num(3)=ny2
    set_contour,num,pview
    levels=intarr(20) & a=0 & da=0.05*data(nb,0)
      for k=0,19 do begin & a=a+da & levels(k)=a & end
        contour,image(nx1:nx2,ny1:ny2),levels=levels,xstyle=1,ystyle=1,$
          xtitle='x(pixel)',ytitle='y(pixel)',$
            title='Beamlet '+string(nb,format='(i3)')+'  '+filename
  nn=strlen(comdata(1))-10
  if nn gt 1 then xyouts,0.15,0.90,strmid(comdata(1),10,nn),/norm
  ss='image('+string(nx1)+':'+string(nx2)+','+string(ny1)+':'+string(ny2)+')'
  ss=ss+'  Peak='+string(data(nb,0),format='(i3)')+' divided 20 steps'
  xyouts,0.15,0.87,strtrim(strcompress(ss),2),/norm
  pview='c'
;
;skip:
return
end
;
;-----------------------------------------------------------------------
;
pro do_profiles,ntime
common com1,filename,dumpname,printname,datename,title
common com2,xdim,ydim,imswch,image,iobject
```



```
common com3,ssin,pview,ch2
common com4,data,nspc),phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com12,nb,type_slice,xslice,yslice,prange,chsave1,chsave2,chsave)
common com15,version,comhead,comdata,moments
;
 ss='' & num='' & xlable=0.35 & alfa='abcdefghijklmnopqrstuvwxyz.,/;[]=-'
 heading=filename
 n1=strpos(comdata(1),'S') & n2=strlen(comdata(1))-n1+1 ; get smoothing factor
 if n1 gt 0 then heading=heading+'  ('+strmid(comdata(1),n1,n2)+')'
  if ntime eq 0 then begin
    imswch=' '
    again:
    if ch2 eq 'o' then begin
      if type_slice eq 'x' then $
        read,'Enter slice y=<1..'+string(ydim,format='(i3)')+'> ',num else $
        read,'Enter slice x=<1..'+string(xdim,format='(i3)')+'> ',num
        if strpos(alfa,num) lt 0 then num=fix(num) else num=-1
        if (type_slice eq 'x') then begin
          if (num lt 1) or (num ge ydim) then begin
            print,'Error, y must be smaller then ',ydim & goto,again & end & en
+d
          if (type_slice eq 'y') then begin
           if (num lt 1) or (num ge xdim) then begin
            print,'Error, x must be smaller then ',xdim & goto,again & end & en
+d
      n=prange(4)+1
        if n eq 5 then begin
          if type_slice eq 'x' then prange(n)=xslice else prange(n)=yslice
            n=n+1 & end
      prange(4)=n & prange(n)=num & xlable=0.35+0.05*(n-5)
        if type_slice eq 'x' then xslice=num else yslice=num
      goto,oplotskip
    end else $
    if ch2 eq 'r' then begin
      read,'Enter kmin,kmax,kstep  <1..>  ',kmin,kmax,kstep
        n=4 & prange(4)=5+fix((kmax-kmin)/kstep)
          for k=kmin,kmax,kstep do begin & n=n+1 & prange(n)=k & end
    end else $
    if ch2 eq 'i' then begin
      imswch='i'
      ch2=strmid(ssin,2,1)
      goto,again
    end
  read,'Enter xy_pixel e.g. x120 or y67 <1..'+string(ydim,format='(i3)')+'> ',s
+s
      if strlen(ss) gt 0 then begin & ss=strtrim(ss,2)
        if strmid(ss,0,1) eq 'x' then begin
          type_slice='x' & xslice=fix(strmid(ss,1,strlen(ss))-1)
        end else begin & type_slice='y'
          if strmid(ss,0,1) eq 'y' then js=1 else js=0
            yslice=fix(strmid(ss,js,strlen(ss))-js)
      end & end
    if (type_slice eq 'x') then begin
      if (xslice lt 1) or (xslice ge ydim) then begin
        print,'Error, y must be smaller then ',ydim & goto,again & end
      sr=string(xdim-1,format='(i3)') & end
    if (type_slice eq 'y') then begin
      if (yslice lt 1) or (yslice ge xdim) then begin
        print,'Error, x must be smaller then ',xdim & goto,again & end
      sr=string(ydim-1,format='(i3)') & end
;
    ss='' & read,'Enter min,max pixel range <1..'+sr+'> LF<default> ',ss
    if strlen(ss) eq 0 then begin
      if (type_slice eq 'x') then begin & prange(0)=1 & prange(1)=xdim-1 & end
      if (type_slice eq 'y') then begin & prange(2)=1 & prange(3)=ydim-1 & end
    end else begin & num=intarr(2) & decode,ss,num,n
      if (type_slice eq 'x') then begin
        if (num(0) gt 0) and (num(0) lt xdim) then prange(0)=num(0)
        if (num(1) gt num(0)) and (num(1) lt xdim) then prange(1)=num(1) & end
      if (type_slice eq 'y') then begin
        if (num(0) gt 0) and (num(0) lt ydim) then prange(2)=num(0)
```

```
          if (num(1) gt num(0)) and (num(1) lt ydim) then prange(3)=num(1) & end
      end
    end ; ntime=0
;
oplotskip:
    if imswch eq 'i' then iwork=iobject else iwork=image
    if type_slice eq 'x' then begin
      if ch2 eq 'o' then oplot,iwork(*,xslice) else $
      plot,iwork(*,xslice),xrange=[prange(0),prange(1)],xstyle=1.$
        yrange=[0,imax],ystyle=1,$
          xtitle='x(pixel)',ytitle='Intensity',title=heading
      xyouts,0.2,0.87,'X_slice at y=',/norm
      xyouts,xlable,0.87,','+string(xslice,format='(i3)'),/norm
    end else $
    if type_slice eq 'y' then begin
      if ch2 eq 'o' then oplot,iwork(yslice,*) else $
      plot,iwork(yslice,*),xrange=[prange(2),prange(3)],xstyle=1,$
        yrange=[0,imax],ystyle=1,$
          xtitle='y(pixel)',ytitle='Intensity',title=heading
      xyouts,0.2,0.87,'Y_slice at x=',/norm
      xyouts,xlable,0.87,','+string(yslice,format='(i3)'),/norm
    end
;
    if (ch2 eq 'r') and (ntime eq 0) then begin & nop=prange(4) & yyx=0.87
      for k=5,nop do begin & slice=prange(k)
        if type_slice eq 'y' then $
          oplot,iwork(slice,*) else oplot,iwork(*,slice)
        if (k eq 14) or (k eq 23) or (k eq 32) or (k eq 41) then $
          begin & xlable=0.35 & yyx=yyx-0.03 & end
        xyouts,xlable,yyx,','+string(slice),/norm
        xlable=xlable+0.05
      end;for
    end
;
    if (ntime eq 1) then begin & nop=prange(4) & prange(4)=4 & yyx=0.87
      if nop gt 4 then begin ; do overplots prange(5..nop)
        for k=5,nop do begin & slice=prange(k)
          if type_slice eq 'y' then $
            oplot,iwork(slice,*) else oplot,iwork(*,slice)
        if (k eq 14) or (k eq 23) or (k eq 32) or (k eq 41) then $
          begin & xlable=0.35 & yyx=yyx-0.03 & end
          xyouts,xlable,yyx,','+string(slice),/norm
          xlable=xlable+0.05
        end;for
      end;if nop
    end;if ntime
;
return
end
;
;-----------------------------------------------------------------------
;
pro set_contour,subcont,sw
common com2,xdim,ydim,imswch,image,iobject
common com13,sub_range
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
;
    if sw eq 'd' then begin
      subcont(0)=1 & subcont(1)=xdim-1 & subcont(2)=1 & subcont(3)=ydim-1 & end
    if subcont(0) lt 1 then subcont(0)=1
    if subcont(1) gt xdim-1 then subcont(1)=xdim-1
    if subcont(2) lt 1 then subcont(2)=1
    if subcont(3) gt ydim-1 then subcont(3)=ydim-1
;
    sub_range.scloc='['+string(subcont(0),format='(i3)')+':'+$
              string(subcont(1),format='(i3)')+','+$
              string(subcont(2),format='(i3)')+':'+$
              string(subcont(3),format='(i3)')+']'
    sub_range.subcont=subcont
;
return
end
;
```



```
;-----------------------------------------------------------------------
;
pro set_surface
common com2,xdim,ydim,imswch,image,iobject
common com13,sub_range
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
;
    subsurf=intarr(4)
    subsurf(0)=xpix0-pixsize
      if subsurf(0) lt 1 then begin & subsurf(0)=1 & subsurf(1)=2*pixsize & end $
        else subsurf(1)=xpix0+pixsize
      if subsurf(1) gt xdim-1 then begin
        subsurf(1)=xdim-1 & subsurf(0)=subsurf(1)-2*pixsize
          if subsurf(0) lt 0 then subsurf(0)=0 & end
    subsurf(2)=ypix0-pixsize
      if subsurf(2) lt 1 then begin & subsurf(2)=1 & subsurf(3)=2*pixsize & end $
        else subsurf(3)=ypix0+pixsize
      if subsurf(3) gt ydim-1 then begin
        subsurf(3)=ydim-1 & subsurf(2)=subsurf(3)-2*pixsize
          if subsurf(2) lt 0 then subsurf(2)=0 & end
    sub_range.sloc='['+string(subsurf(0),format='(i3)')+':'+$
              string(subsurf(1),format='(i3)')+','+$
              string(subsurf(2),format='(i3)')+':'+$
              string(subsurf(3),format='(i3)')+']'
    sub_range.subsurf=subsurf
;
return
end
;
;-----------------------------------------------------------------------
;
pro smooth_data,ch
common com2,xdim,ydim,imswch,image,iobject
common com5,plot_lets,center,maxb,nbeam
common com15,version,comhead,comdata,moments
common com19,smoothing
;
again: cc=''
  print,'Smoothing = 0 will allow regeneration of window centers'
  read,'Enter width of sliding box car smoothing, LF exits <3..> ',cc
    if strlen(cc) lt 0 then return
    c1=strmid(cc,0,1)
    if strpos('0123456789',c1) lt 0 then begin
      print,'  ERROR, input must be numeric, try again dummy' & goto,again & en
d
    width=fix(cc)
    if width ge 3 then begin
      image=smooth(image,width)
      comdata(1)='['+string(xdim,format='(i3)')+','+$
                  string(ydim,format='(i3)')+']'+' '+$
                  string(nbeam,format='(i3)')+' beamlets'+$
                  ', Smoothing='+string(width,format='(i2)')
      smoothing=width
    end
    wset,2 & tv,image & wset,0
  if ch ne 's' then begin & cc=' ' & cb=' '
    read,'Regenerate beamlets                          <y,n> ',ch
    read,'Re_Calculate beamlet centers                 <y,n> ',cc
    read,'Subtract global backgournd                   <y,n> ',cb
    if cb eq 'y' then fix_image,'b'
    if cc eq 'y' then ch='C'
  end
;
return
end;
;
;-----------------------------------------------------------------------
;
pro default_data
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,asin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
```

```
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
common com13,sub_range
common com15,version,comhead,comdata,moments
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
common com19,smoothing
;
  sub_range.subsurf(1)=20 & sub_range.subsurf(3)=20
  sub_range.subcont(1)=20 & sub_range.subcont(3)=20
  nspcj=0 & xdim=384 & ydim=576
  level=10
;
  xcmpix=0.0107 & ycmpix=0.0107 ; 4/11/97 GBI data
  rhole=0.005 & separation=0.250 & thickness=0.01524  ;cm
  drift=0.150     ; cm
  restm=36262.2   ; Pottasium
  energy=0.065    ; MeV
  current=0.002   ; Amps
;
  xcmpix=0.0054 & ycmpix=0.0054 ; 7/22/97 GBI data
  rhole=0.005 & separation=0.250 & thickness=0.01524  ;cm
  drift=0.150     ; cm
  restm=36262.2   ; Pottasium
  energy=0.080    ; MeV
  current=0.00216  ; Amps
;
  typemask=0
; xwindow=fix(2*separation/xcmpix) & ywindow=fix(2*separation/ycmpix)
  xwindow=20 & ywindow=20
  window=xwindow*ywindow
  ipass=fltarr(2)
  ipass(0)=0 & ipass(1)=0
  brightness=0.0 & emitx=0 & emity=0
  gamma=1+(energy/restm) & betagamma=sqrt(gamma^2-1)
  xpix0=0 & ypix0=0 & pixsize=512 & set_surface
  maxdata=300
  chelscl=' ' & chellipse='y'
  smoothing=0
  filename='temp.ras'
  dumpname='temp.dump'
  printname='temp.lpr'
  dataname='temp.dat'
  version='VERSION(5/19/97)'
  comhead( 0)='**.RAS' File:      ='
  comhead( 1)='Image(nx,ny)       ='
  comhead( 2)='Imin,Imax,Backg    ='
  comhead( 3)='X,Y (cm/pixel)     ='
  comhead( 4)='Energy(MeV),betag  ='
  comhead( 5)='Window(pixels)     ='
  comhead( 6)='R_Hole,Spacing(cm)='
  comhead( 7)='Drift(meters)      ='
  comhead( 8)='Emittance(mm-mr)   ='
  comhead( 9)='Current,Brightness='
  comhead(10)='Level(percent)     ='
  comhead(11)='xr(cm),r12,rp(mr)  ='
  comhead(12)='yr(cm),r12,rp(mr)  ='
  comhead(13)='J(x),(y),(xy),(av)='
  comhead(14)='R(cm),RP(mr),Ipass='
  comhead(15)='File path name     ='
  comhead(16)='Save path name     ='
  comhead(17)='title ='
  comhead(18)='Halfwidths         ='
  comhead(19)='Mask Rotation(deg)='
  comdata=comhead
  comdata(0)=filename+'    '+version
  comdata(1)='[0,0]'
  comdata(2)='[0,0] 0'
  comdata(3)='['+string(xcmpix,format='(f7.5)')+','+$
                 string(ycmpix,format='(f7.5)')+']'
  comdata(4)='['+string(energy,format='(f8.4)')+','+$
```



```
                 string(betagamma,format='(f8.4)')+']    '+$
                 string(restm,format='(f12.3)')
  comdata(5)='['+string(xwindow,format='(i3)')+','+$
                 string(ywindow,format='(i3)')+']'
  xwindow=fix(xwindow/2) & ywindow=fix(ywindow/2) & window=fix(window/2)
  comdata(6)='['+string(rhole,format='(f8.4)')+','+$
                 string(separation,format='(f8.4)')+','+$
                 string(thickness,format='(f8.4)')+'] N'
  comdata(7)=string(drift,format='(f8.4)')
  ave=0.5*(emitx+emity)
  comdata(8)='['+string(emitx,format='(f8.3)')+','+$
                 string(emity,format='(f8.3)')+']'+$
                 '    ave:'+string(ave,format='(f8.3)')+$
                 '   norm:'+string(ave*betagamma,format='(f9.4)')
  comdata(9)='['+string(current,format='(f8.2)')+'('+$
                 string(ipass(0),format='(f6.3)')+'/'+$
                 string(ipass(1),format='(f4.0)')+'),'+$
                 string(brightness,format='(e10.2)')+' +-'+$
                 string(brightness,format='(e10.2)')+']'
  comdata(10)=string(level,format='(i4)')
  comdata(11)='['+'  0,   0,   0   '+']'
  comdata(12)='['+'  0,   0,   0   '+']'
  comdata(13)='[ , , ]' & comdata(14)='[ , , ]'
  comdata(15)='-gbl/'
  comdata(16)=''
  comdata(18)=''
  comdata(19)='0'
return
end
;
;-----------------------------------------------------------------------
;
pro show_list
;
common com5,plot_lets,center,maxb,nbeam
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
;
  lprtype=strarr(maxptype)
  lprtype( 0)='  0)  Surface plot'
  lprtype( 1)='  1)  Contour plot'
  lprtype( 2)='  2)  Profile plot'
  lprtype( 3)='  3)  Phase plot at mask'
  lprtype( 4)='  4)  Beamlet plot'
  lprtype( 5)='  5)  Brightness plot'
  lprtype( 6)='  6)  Summary plot'
  lprtype( 7)='  7)  Analyze plot'
  lprtype( 8)='  8)  Mask plot'
  lprtype( 9)='  9)  Window plot'
  lprtype(10)=' 10)  TV image plot'
  lprtype(11)=' 11)  Phase plot x-yp, y-xp, xp-yp, x-y'
  lprtype(12)=' 12)  Emittance vs Level'
  lprtype(13)=' 13)  Mask and Beam centers'
  lprtype(14)=' 14)  Phase plot at waist'
  lprtype(15)=' 15)  Phase plot at image(tv)'
  lprtype(16)=' 16)  Phase plot at distance d.'
  lprtype(17)=' 17)  Summary plot vs radius.'
  lprtype(18)=' 18)  Movie profile cuts.'
  lprtype(19)=' 19)  Get Beam Halo'
  lprtype(20)=' 20)  All beamlets plot'
  lprtype(21)=' 21)  Round Mask -- holes in use'
  lprtype(22)=' 22)  Emittance vs mask rotation'
  lprtype(23)=' 23)  Rectangular Mask holes in use'
  lprtype(24)=' 24)  .....'
;
  print,'* will produce hard copy of:'
  for k=0,maxptype do begin
    if ptype(k) ne 0 then print,lprtype(k)
      if (k eq 4) and (ptype(k) ne 0) then begin & ss=''
        for kb=1,maxb do ss=ss+string(plot_lets(kb,0),format='(i4)')
          print,'      '+ss
      end
    end
  print,'-----------------------------------------------------------------'
```

```
  ss='' & ch=' ' & read,'Eliminate any of these plots ? <y,n> ',ch
  if ch eq 'y' then begin & num=intarr(12)
    read,'Enter 0..18 for plots to be eliminated, space delimited ==> ',ss
      decode,ss,num,n
    for j=0,maxptype do for k=0,n-1 do if (j eq num(k)) then ptype(j)=0
  end
;
return
end
;
;-----------------------------------------------------------------------
;
pro toggle_windows,ss
;
  ch=strmid(ss,1,1)
  if strlen(ss) eq 2 then begin
    if ( ch eq 0 or ch eq 2 ) then begin
      wset,ch & wshow,ch,1 & return
    end
  end
;
  nw=0 &  ch=' '
  if (strlen(ss) lt 3) then begin
    read,'S)how H)ide window 0, 2 or 4 (s0,h0,s2,s4,h2,h4)=> ',ss
      if (strlen(ss) ne 2) then return
    start=0
  end else start=1
;
  ch=strmid(ss,start,1)
  if (strpos('sh',ch) lt 0) then return
  nw=fix(strmid(ss,start+1,1))
  if ch eq 's' then kw=1 else kw=0
  if (nw eq 0) or (nw eq 2) or (nw eq 4) then wshow,nw,kw
;
return
end
;
;-----------------------------------------------------------------------
;
pro set_path_name
common com3,ssin,pview,ch2
common com15,version,comhead,comdata,moments
;
  ss='' & swn=''
  if ch2 eq 's' then begin & swn='Save_File ' & nsw=16 & end $
    else begin & swn='Image_File ' & nsw=15 & end
  read,'Directory path name for '+swn+', '+comdata(nsw)+' LF none => ',ss
  if strlen(ss) le 0 then comdata(nsw)='' else begin
    if strmid(ss,strlen(ss)-1,1) ne '/' then $
      comdata(nsw)=ss+'/' else comdata(nsw)=ss
  end
return
end
;
;-----------------------------------------------------------------------
;
pro read_titles
;
  pool='0123456789abcdefghijklmnopqrstuvwxyz'
  dpath='' & head='' & suff=' ' & name='' & nfile=1 & prsw=' '
  read,'Enter directory "path" name       ===> ',dpath
  read,'Enter starting *.RAS file name    ===> ',name
  read,'How many sequentcial files        ===> ',nfile
  read,'Hard copy                   <y,n> ===> ',prsw
  nn=strpos(name,'.ras')
  head=strmid(name,0,nn-2)
  suff=strmid(name,nn-2,2)
  nc1=strpos(pool,strmid(suff,0,1)) & nc2=strpos(pool,strmid(suff,1,1))
  if nfile lt 1 then nfile=1
  if strpos(dpath,'/') lt 0 then dpath=dpath+'/'
;
  ierr=0
  on_ioerror,ioerr
```



```
  if prsw eq 'y' then openw,2,'temp.titles', /variable
  for kf=1,nfile do begin
    openr,unit,dpath+name,/GET_LUN
      hdr=assoc(unit,bytarr(512)) & h=hdr(0)
        close,1 & free_lun,unit
  title=''
    kmax=256 & while h(kmax) eq 32 do kmax=kmax-1
      if kmax-11 lt 78 then for k=10,kmax do title=title+string(h(k)) else $
        begin
          for k=10,25 do title=title+string(h(k))
          for k=max-50,kmax do title=title+string(h(k))
        end
  print,name+'|'+title
  if prsw eq 'y' then printf,2,name+'|'+title
  nc2=nc2+1 & if nc2 ge strlen(pool) then begin & nc2=0 & nc1=nc1+1 & end
    ss=strmid(pool,nc1,1)+strmid(pool,nc2,1)
      if kf lt nfile then name=head+ss+'.ras'
  end
  goto,finish
ioerr: print,!err_string
finish: if prsw eq 'y' then close,2
if prsw eq 'y' then print,'File temp.titles saved for printing'
;
return
end
;
;-----------------------------------------------------------------------
;
pro get_scale,chb
;                                        call by 'cs' contour scale command.
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com13,sub_range
;
;  if strmid(ssin,2,1) eq '+' then begin
;
    if chb eq '+' then begin
      nx=xwindow+2*separation/xcmpix & ny=ywindow+2*separation/ycmpix & end $
    else begin & nx=xwindow & ny=ywindow & end
;
    num=intarr(4)
    num(0)=min(center(1:nbeam,2))-nx/2
    num(1)=max(center(1:nbeam,3))+nx/2
    num(2)=min(center(1:nbeam,4))-ny/2
    num(3)=max(center(1:nbeam,5))+ny/2
    set_contour,num,' '
;
return
end
;
;-----------------------------------------------------------------------
;
pro doplots,chw,ss,ntime
;
common com1,filename,dumpname,printname,datanama,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,rhole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,jbeamn,jradius,bjj
common com12,nb,type_slice,xslice,yslice,prange,chsavel,chsave2,chsavej
common com13,sub_range
common com15,version,comhead,comdata,moments
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
common com17,chbsv,have_done,maxptype,ptype,chellipse,chel6d,chelscl
```

```
common com20,scales
;
  num=intarr(4)
  ssloc=sub_range.ssloc & scloc=sub_range.scloc
  nn=strlen(comdata(1))-10
  if nn gt 1 then smhead='    '+strmid(comdata(1),10,nn) else smhead=''
  if (ptype(3) eq 1) and (ch2 eq 's') then begin ; set emittance plot scales
    read,'Enter emittance scales (xmin,xmax,xpmin,xpmax) LF cancels ==> ',ss
      if strlen(ss) eq 0 then begin & for k=0,3 do num(k)=0 & end $
        else decode,ss,num,n
      scales(0:3)=num & chelscl='S'
    end
  if (ptype(5) eq 1) and (ch2 eq 's') then begin ; set brightness plot scales
    print,'Brightness scales are Jmin=10^min, Jmax=10^max'
      read,'Enter brightness scales (0,rmax,min,max) LF cancels ==> ',ss
        if strlen(ss) eq 0 then begin & for k=0,3 do num(k)=0 & end $
          else decode,ss,num,n
      num(0)=1.0 & scales(4:7)=num & ptype(5)=0 & return
    end
;
  cc=0
  if ntime eq 0 then begin
    if strpos('024',ch2) ge 0 then begin & cc=fix(ch2)
      wset,cc & wshow,cc,1
        if strlen(ss) eq 3 then ch2=strmid(ss,2,1)
    end
  end
;
; s) SURFACE PLOT
  if ptype(0) eq 1 then begin
    subsurf=sub_range.subsurf
    if ntime eq 0 then begin & ss='' & sn='' & sc=''
      sead,'Enter sub_surface range '+ssloc+' q)uits ==> ',ss
        if ss eq 'q' then return
        if strlen(ss) gt 0 then begin
          decode,ss,num,n & if n gt 0 then for k=0,n-1 do subsurf(k)=num(k)
            ss='' & for k=0,3 do ss=ss+string(subsurf(k))
              ssloc=strcompress(strtrim(ss,2)) & end & end
    if ntime eq 1 then print,'Patience ..., surface printing'
      if (subsurf(1)-subsurf(0) gt xdim/2) or $
         (subsurf(3)-subsurf(2) gt ydim/2) then begin
          get_rebin,c,cx,cy & surface,c & end $
      else surface,image(subsurf(0):subsurf(1),subsurf(2):subsurf(3)),color=0
      xyouts,0.2,0.87,'Grid(x1:x2,y1:y2) '+ssloc+'    '+filename,/norm
      xyouts,0.2,0.84,smhead,/norm
      pview='s'
    end
; c) CONTOUR PLOT
  if ptype(1) eq 1 then begin
    if ntime eq 0 and ch2 ne 's' then begin & ss='' & sn='' & sc=''
      read,'Enter sub_contour range '+scloc+' d)efault, q)uits ==> ',ss
        if ss eq 'q' then return
        if ss eq 'd' then set_contour,num,'d' else $
        if strlen(ss) gt 0 then begin
          decode,ss,num,n
            set_contour,num,' ' & end
    end
    scloc=sub_range.scloc
    step=fix(0.05*(imax-background)) & levels=fix(background)+step*indgen(20)
    if ntime eq 1 then print,'Patience ..., contour printing'
    ss='Grid(x1:x2,y1:y2) '+scloc+$
       ' Levels='+string(levels(0),format='(i3)')+'..'+$
       string(levels(19),format='(i3)')+' in steps of '+$
       string(step,format='(i3)')
    subcont=sub_range.subcont
       x1=subcont(0) & x2=subcont(1) & y1=subcont(2) & y2=subcont(3)
       cx=xcmpix*indgen(x2-x1+1) & cy=ycmpix*indgen(y2-y1+1)
       contour,image(subcont(0):subcont(1),subcont(2):subcont(3)),cx,cy,$
         xmargin=[8,8],ymargin=[4,4],levels=levels,xstyle=1,ystyle=1,color=0,$
           subtitle=ss,xtitle='x(cm)',ytitle='y(cm)'
       axis,xaxis=1,xrange=[x1,x2],xstyle=1,xtitle='x(pixels)',ticklen=-0.02
       axis,yaxis=1,yrange=[y1,y2],ystyle=1,ytitle='y(pixels)',ticklen=-0.02
    xyouts,0.02,0.98,filename+smhead,color=0,/norm
```



```
    pview='c'
  end
;
; p)PROFILE PLOT
  if ptype(2) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., profile printing'
      do_profiles,ntime & pview='p'
    end
; a)PHASE SPACE
  if ptype(3) eq 1 then begin
    if (ntime eq 0) and (ch2 eq '6') then ptype(11)=1
    if (ntime eq 1) then print,'Patience ..., phase plot printing'
    get_emittance,'n'
      if ptype(11) eq 1 then begin
        plot_6d_space,ntime
          if ntime eq 1 then plot_phase_space,ntime
      end else begin & ch2=' ' & plot_phase_space,ntime & end
      pview='a'
    end
; b)BEAMLETT PLOT
  if ptype(4) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., beamlet printing'
      plot_beamlet,ntime & pview='b'
    end
; j)BRIGHTNESS PLOT
  if ptype(5) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., brightness printing'
        get_emittance,'n'
        plot_brightness,ntime & pview='B'
    end
; v)SUMMARY PLOT
  if ptype(6) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., summary printing'
        summary,ntime & pview='s'
    end
; a)ANALYZE
  if ptype(7) eq 1 then begin & pview='a'
    if ntime eq 1 then print,'Patience ..., analyze printing'
      analyze,ntime
    end
; i)mask and centers
  if ptype(8) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., mask printing'
        plot_mask,ss & pview='m'
    end
; w)indow
  if ptype(9) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., window printing'
        plot_window,chw,ntime & pview='w'
    end
; t)TV
  if ptype(10) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., TV printing'
      plot_tv_frame,ntime & pview='t'
    end
; e) emittance vs level
  if ptype(12) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Plotting emittance vs Levels'
      do_levels,ntime
  end
; lw)mask and beam centers
  if ptype(13) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., mask printing'
        ch2='w' & plot_mask,'w' & pview='m'
  end
; ew)ellipse transformed to waist
  if ptype(14) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Plotting ellipse at waist'
        ch2='w' & plot_phase_space,ntime
  end
; el)ellipse transformed to tv image
  if ptype(15) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Plotting ellipse at tv image'
```

```
        ch2='i' & plot_phase_space,ntime
  end
; ed)ellipse transformed to distance d
  if ptype(16) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Plotting ellipse at distance d.'
        ch2='d' & plot_phase_space,ntime
  end
; vr)Summary vs radius
  if ptype(17) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Summary vs radius printing'
        ch2='r' & summaryr,ntime & pview='s'
  end
; M)Movie
  if ptype(18) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Running the 'Movie' to printer'
      movie,ntime
  end
;
  if ptype(19) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Printing halo of beam'
      get_halo,ntime
  end
;
  if ptype(20) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Printing all beamlets'
      plot_all_beam,ntime
  end
;
  if ptype(21) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Printing Round Mask -- holes in use
+'
      gen_round,'s'
  end
;
  if ptype(22) eq 1 then begin
    if ntime eq 1 then print,'Patience ..., Printing emittance vs mask rotation
+'
      vary_mask,ntime
  end
;
  if ptype(23) eq 1 then begin
    if ntime eq 1 then $
      print,'Patience ..., Printing Rectangular Mask holes in use'
      ch2='s' & gen_rectangular,ch2
  end
;
  if ptype(24) eq 1 then begin
  end
;
; always make last plot boggus -- IDL dosnt end with show page
  if ntime eq 1 then begin
    print,'Patient ..., ending plot file'
      ch2=' ': & plot_mask,'_'
  end
;
  if cc gt 0 then wset,0
return
end
;
;-----------------------------------------------------------------------
;
pro pv3
;
common com1,filename,dumpname,printname,dataname,title
common com2,xdim,ydim,imswch,image,iobject
common com3,ssin,pview,ch2
common com4,data,nspcj,phase,maxdata,mask,nmask,typemask
common com5,plot_lets,center,maxb,nbeam
common com7,xcmpix,ycmpix,drift,thole,separation,thickness
common com8,window,xwindow,ywindow,chwin2,level
common com9,current,energy,betagamma,restm,brightness,emitx,emity,emitn,nemit
common com10,xcent,ycent,nxcent,nycent,imin,imax,background
common com11,ipass,bradius,xave,yave,ibeamn,jradius,bjj
```



```
common com12,nb,type_slice,xslice,yslice,prange,chsave1,chsave2,chsave3
common com13,sub_range
common com14,s11,s12,s22,s33,s34,s44,xcn,ycn,xpcn,ypcn,r12,r34,lx,ly,ld,idx,idy
common com15,version,comhead,comdata,moments
common com16,xwcent,ywcent,xpix0,ypix0,pixsize
common com17,chbsy,have_done,maxptype,ptype,chellipse,chel6d,chelscl
common com18,kmovie,nslice,slices
common com19,smoothing
common com20,scales
common com21,nmax,masklist
common com22,flag,numpix
;
  sub_range=(subrange,ssloc:'',scloc:'',subsurf:intarr(4),subcont:intarr(4))

  ss=''
  title=' '
  ip.color=0 & ip.background=16777215    ; 256**3-1
  comhead=strarr(20)
  scales=fltarr(8)
  slices=strarr(21)
  nmask=0
  flag=0
;
  default_data
  nbeam=1 & xdim=384 & ydim=576 & image=bytarr(xdim,ydim)
;  nbeam=1 & xdim=512 & ydim=490 & image=bytarr(xdim,ydim)
  chbsv=' ' & chw='W' & cc=' ' & ch='&' & ch2=' ' & cerror='&'
  window,0,title='IMAGE ANALYSIS - HIF GBI  (WINDOW 0)',xsize=590,ysize=590
  greetings,cerror
  window,2,title='TV IMAGE (WINDOW 2)',$
                                xsize=xdim,ysize=ydim,xpos=60,ypos=520
  wshow,2,0 ; hid this window for future use
  window,4,title='PEPPERPOT (WINDOW 4)',$
                                xsize=xdim,ysize=ydim,xpos=60,ypos=100
  wshow,4,0 ; hid this window for future use
;
  ierr=0 & xslice=20 & yslice=20 & type_slice='y' & maxb=0
  imswch=' ' & nemit=0 & ntime=0
  prange=intarr(50) & prange(4)=4
  maxptype=25 & ptype=bytarr(maxptype+1)
  nxcent=0 & nycent=0 & ytlt=0.9
;            0     1   2    3    4    5    6    7   8
; data(,*)=pixel,xmin,xmax,xave,ymin,ymax,yave,sum,backg
;
  data=fltarr(maxdata,9)
;                          .---- xcent,ycent,xmin,xmax,ymin,ymax for object
  center=intarr(maxdata,6)
;                        .------ xcm,ycm,radius
  mask=fltarr(maxdata,3)
  plot_lets=intarr(maxdata,2)
  emitn=fltarr(maxdata,3)
  bradius=fltarr(maxdata)
  xcent=intarr(maxdata) & ycent=xcent
  nmax=150 & masklist=fltarr(2*nmax+1,3)   ; masklist(0,0)=nx, masklist(0,1)=ny
  moments=fltarr(10)
;
newfile:
  chel6d=' '
  pview=' '
  xwcent=0 & ywcent=0
  kmovie=intarr(5) ; kmin,kmax,kstep,ksmooth,junk
  if comdata(15) ne '-gbi/' then begin
    read,'Save restore file, r)egenerate from RAS file          <y,n,r> ==> ',ss
      if ss eq 'y' then begin
        if dataname ne 'temp.dat' then $
          print,'This data was 'restored' from file '+dataname
        read,'Enter 'dump' filename, LF uses 'temp.dat' ==> ',ss
          if strlen(ss) gt 0 then dataname=ss else dataname='temp.dat'
            dumpfile,dataname,'s'
      end else if ss eq 'r' then begin
        cc=' ' & get_centers,cc & goto,next
;         cc=' ' & get_centers,cc & goto,next
      end
```

```
ans=' '
time=0.0
 read,'Do you wish to write emittance and moment results to file?(y,n)   ',ans
  if ans eq 'y' then begin
      read,'enter time (e.g. 100)',time
      spotrad=fltarr(200)
      beamave=0
      for k=1,nbeam do begin
        beamave=beamave+data(k,7)/nbeam
      end
      xemitav=0 & yemitav=0
      for k=1,nbeam do begin
        xemitav=xemitav+emitn(k,0)
        yemitav=yemitav+emitn(k,1)
        spotrad(k)=sqrt(mask(k,0)^2+mask(k,1)^2)
      end
      xemitav=1e6*xemitav/nbeam
      yemitav=1e6*yemitav/nbeam
      print, ' '
      openu,5,'emitall.dat',/append
      openw,8,'moment.dat',/append
      print,'Writing to emittance summary files "emitall.dat"and "emitspot.dat".
...'
      emitx=1e6*betagamma*emitx & emity=1e6*betagamma*emity
      printf,5,filename,time,emitx,emity
      print,filename,time,emitx,emity,'    to emitall.dat!'
      print,8,filename,time,moments(*),'    to moment.dat!'
      printf,8,filename,time,moments(*)
      close,5 & close,8
    endif else begin
      print, ' '
      print,'No emittance data written' & $
    endelse
  end
;
  readdata,cerror
    if cerror eq 'r' then goto,next  ; return to main program
    if cerror eq 'R' then begin      ; restore from "save" file
      read,'Enter "saved" filename, LF uses temp.dat, q)uits ==> ',ss
        if ss eq 'q' then goto,printit
        if strlen(ss) gt 0 then $
          dumpfile,ss,'r' else dumpfile,'temp.dat','r'
          set_surface
          set_contour,prange(46:49),'d'
          have_done='f' & chw='W'
          for k=0,maxptype do ptype(k)=0
          wset,0 & end
    if cerror eq 'q' then goto,printit
  ioerror=' '
;
rasfile:
  readfile,ioerror,cerror
  if ioerror eq 't' then goto,newfile
;
  prange(0)=1 & prange(1)=xdim & prange(2)=1 & prange(3)=ydim
  set_contour,prange(46:49),'d'
  have_done='f'
  if (cerror eq 'R') or (ioerror eq 'R') then begin
;    fit_gauss &
     get_beamlets,0
;print,'gaussed it!  '
        get_emittance,' '
      goto,next
  end
;
gen_again:
  if ch ne '&' then print,'Hold on, generating image from file '+filename
  if (cerror ne 't') and ((nxcent gt 0) and (nycent gt 0)) then begin
;   fit_gauss
    get_beamlets,0
      gen_mask,'A'
        get_emittance,' '
    end
```



```
  wset,2 & wshow,2,1
  tv,image & xyouts,0.2,0.90,filename,/norm
  xyouts,0.2,-0.07,filename,/norm & xyouts,0.2,-0.10,title,/norm
  wset,0
  tvscl,iobject
;
plotit:
;  if (nbeam eq 0) then print,'RAW MODE'
print,'RAW MODE'
;
next:
  if ntime eq 0 then begin
    for k=0,maxptype do if ptype(k) eq 1 then ptype(k)=-1
     read,'Enter: H)elp (A,B,C,D,E,H,I,J,L,M,N,O,P,Q,R,S,T,V,W,X,Y,Z,*)--> ',ss
     if ch eq 'h' or ch eq 'H' then wshow,0,1 ; help command hid window
     ssin=ss
       ch=strmid(ss,0,1)
         ss=strlowcase(ss)
           if strlen(ss) gt 1 then ch2=strmid(ss,1,1) else ch2=' '
     if ch eq 'O' then begin & ch='o' & have_done='f' & end
;
      if ch ge '0' and ch le '9' then begin & ch2='n' & ptype(4)=1 & end else $
      if ch eq 'A' then autoview else $
      if ch eq 'B' then fix_image,'b' else $
      if ch eq 'E' then fix_image,'E' else $
      if ch eq 'H' then begin & wshow,0,0 & readme & goto,next & end else $
      if ch eq 'M' then ptype(18)=1 else $
      if ch eq 'P' then set_path_name else $
      if ch eq 'R' then dumpfile,'','r' else $
      if ch eq 'S' then begin
                          if (ch2 eq 'a') then adjust_centers else begin
                          cc=' ' & smooth_data,cc
                 if (cc eq 'C') then get_centers,cerror
                          if (cerror eq 'r') then goto,next
                          if (cc eq 'y') or (cc eq 'C') then goto,gen_again
                          end
                        end else $
      if ch eq 'T' then read_titles else $
      if ch eq 'W' then toggle_windows,ss else $
      if ch eq 'a' then ptype(7)=1 else $
      if ch eq 'b' then begin
                          if ch2 eq 'a' then ptype(20)=1 else ptype(4)=1
                        end else $
      if ch eq 'c' then begin
                          if (ch2 eq 's' ) then get_scale,strmid(ss,2,1)
                          ptype(1)=1
                        end else $
      if ch eq 'd' then begin & data_input,cc
                           if cc eq 'y' then goto,gen_again
                           if cc eq 'j' then get_emittance,cc
                        end else $
      if ch eq 'e' then begin
                          if strpos('024',ch2) ge 0 then $
                                  chc=strmid(ss,2,1) else chc=ch2
                          if chc eq 'l' then ptype(12)=1 else $
                          if chc eq 'w' then ptype(14)=1 else $
                          if chc eq 'i' then ptype(15)=1 else $
                          if chc eq 'd' then begin
                                             print,'Enter distance (meters) '
                                             ld=get_word(ssin,2)
                                             ptype(16)=1 & end else $
                          ptype(3)=1
                        end else $
      if ch eq 'f' then fix_data else $
      if ch eq 'g' then ptype(19)=1 else $  ; get beam halo
      if ch eq 'h' then begin & wshow,0,0 & helpme & goto,next & end else $
      if ch eq 'I' then begin
                          if ch2 eq 'k' then rotate_mask,ch2 else $
                          if ch2 eq 'r' then begin
                            ptype(21)=1 & gen_mask,ch2
                              if ch2 eq 'q' then goto,next
                            ch2=' '
                          end else $
```

```
                                if ch2 eq 'v' then ptype(22)=1 else $
                                if ch2 eq 'w' then ptype(13)=1 else $
                                if ch2 eq 'g' then begin
                                  if strmid(ssin,2,1) eq 's' then ptype(23)=1
                                    gen_mask,ch2
                                end else ptype(8)=1
                              end else $
        if ch eq 'j' then ptype(5)=1 else $
        if ch eq 'l' then begin & listdata,ch2 & goto,next & end else $
        if ch eq 'm' then move_curser else $
        if ch eq 'n' then goto,newfile else $
        if ch eq 'o' then begin & output,have_done,ch & goto,next & end else $
        if ch eq 'p' then ptype(2)=1 else $
        if ch eq 's' then ptype(0)=1 else $
        if ch eq 't' then ptype(10)=1 else $
        if ch eq 'u' then begin
                              read,'Abort on what beamlet 1..nbeam ',iabort
                              if iabort gt 0 and iabort le nbeam then $
                                get_beamlet,iabort
                              end else $
        if ch eq 'v' then begin
                              if ch2 eq 'r' then ptype(17)=1 else ptype(6)=1
                              end else $
        if ch eq 'w' then ptype(9)=1 else $
        if ch eq 'x' then begin & get_beamlets,0 & get_emittance,'x' & end else $
        if ch eq 'y' then begin & dump_image,image & goto,next & end else $
        if ch eq 'z' then begin & zero_image,image,ch
                              if ch eq 'y' then goto,gen_again else goto,next
                              end else $
        if (ch eq '*') or (ch eq 'q') then begin
                      if ch2 eq 'l' then begin & show_list & goto,next & end
                      if ch2 ne '*' then goto,printit
                        for k=1,maxb do plot_lets(k,0)=0 & maxb=0
                        for k=0,maxptype do ptype(k)=0
          end else goto,next
    end
;
; according to non zero values of ptype() doplots does the plotting
    doplots,chw,ss,ntime
      if ntime eq 0 then goto,next
;
;-----------------------------------------------------------
printit:
  if ntime eq 0 then begin
    if ch eq '*' then goto,nowprint
      cc=' ' & read,'Hard copy <y,n,r)eturn,s)ave,b)ypass> ',cc
        if cc eq 'b' then goto,bypass
        if cc eq 's' then begin
          print,'This data was "restored" from file '+dataname
          read,'Enter "save" filename, LF uses "temp.dat" ==> ',ss
            if strlen(ss) gt 0 then dumpfile,ss,'s' else dumpfile,'temp.dat','s
+'
              goto,bypass
          end
        if cc eq 'r' then goto,plotit
        if cc ne 'y' then goto,myexit
    nowprint: ytit=-0.15
    psname=' '
    read,'Output filename?',psname
      for k=0,maxptype do if ptype(k) ne 0 then ptype(k)=1; & psname='image.ps'
        ntime=1 & set_plot,'ps' & device,/encapsul,filename=psname & goto,ploti
+t
;         ntime=1 & set_plot,'ps' & device,filename=psname & goto,plotit
  end else begin
      ;device,/close
;     spawn,'lpr '+psname & set_plot,'x'
       !p.color=0 & !p.background=16777215 & end
;  if ch eq '*' then begin & for k=1,maxb do plot_lets(k,0)=0 & maxb=0
  if ch eq '*' then begin & for k=1,1 do plot_lets(k,0)=0 & maxb=0
    ytit=0.9 & ntime=0 & for k=0,maxptype do ptype(k)=0 & goto,next & end
;
myexit:
  print,'Hold on, writing "dump" files'
```



```
  ch='E' & output,have_done,ch
  print,'File : '+printname+','+dumpname+' written'
bypass:
ans=' '
time=0.0
read,'Do you wish to write emittance and moment results to file?(y,n)   ',ans
 if ans eq 'y' then begin
read,'enter time (e.g. 100)',time
  spotrad=fltarr(200)
beamave=0
  for k=1,nbeam do begin
    beamave=beamave+data(k,7)/nbeam
  end
  xemitav=0 & yemitav=0
  for k=1,nbeam do begin
    xemitav=xemitav+emitn(k,0)
    yemitav=yemitav+emitn(k,1)
    spotrad(k)=sqrt(mask(k,0)^2+mask(k,1)^2)
  end
  xemitav=1e6*xemitav/nbeam
  yemitav=1e6*yemitav/nbeam
  print, ' '
  openu,5,'emitall.dat',/append
  openw,8,'moment.dat',/append
  print,'Writing to emittance summary files "emitall.dat"and "emitspot.dat"...'
  emitx=1e6*betagamma*emitx & emity=1e6*betagamma*emity
  printf,5,filename,time,emitx,emity
  print,filename,time,emitx,emity,'    to emitall.dat!'
  print,8,filename,time,moments(*),'    to moment.dat!'
  printf,8,filename,time,moments(*)
  close,5 & close,8
 endif else begin
  print, ' '
  print,'No emittance data written' & $
  endelse
read,'Do you wish to write beamlet peak data to file?(y,n)   ',ans
 if ans eq 'y' then begin
   openw,9,'peak.dat'
  printf,9,nbeam
  print,'nbeam=',nbeam
  for k=1,nbeam do begin
    print,mask(k,0),mask(k,1),data(k,7)
    printf,9,mask(k,0),mask(k,1),data(k,7)
  end
  close,9
 endif
print,'  '
print,'  '
end
```

References

[1] Taubes, G. "News and Commentary" *Science*, Vol. 262, December, 1992.

[2] Friedman, A. et al, "Recirculating Induction Accelerators as Drivers for Heavy Ion Fusion" *Phys. Fluids B: Plasma Physics* **5**, 2698 (1993)

[3] Sharp, W., "A Comparison of Driver Concepts for Heavy Ion Fusion" *LLNL ICF Quarterly Report*, Vol. 4, No. 2, March 1994

[4] Barnard J., et al, "Recirculating Induction Accelerators as Drivers for Heavy Ion Fusion" *Phys. Fluids B***5** (7), July 1993 2698-2706.

[5] Moir, R., "HYLIFE-II Inertial Confinement Fusion Power Plant Design" *Particle Accelerators*, 1992, Vols. 37-38, pp. 467-480

[6] Godlove, T., Keefe, D., Vaughan, D., "Heavy-Ion Accelerator Research for Inertial Fusion" LBL Publication, August, 1987.

[7] Friedman, A., Bangerter, R., Herrmannsfeldt, W., "Progress in Heavy Ion Drivers for Inertial Fusion" LLNL Report UCRL-JC-117332, December, 1994.

[8] Private communication with D. Callahan, October, 1997

[9] Cable, M., et al, "Heavy Ion Fusion Experiments at LLNL" Eigth ICFA Advanced Beam Dynamics Workshop on Space Charge Dominated Beams and Applications of High Brightness Beams, Bloomington, Indiana. October, 1995.

[10] Barnard J. J. et al, "Study of Recirculating Induction Accelerators as Drivers for Heavy Ion Fusion" LLNL Report UCRL-LR-108095, September 1991.

[11] Reiser, M. Theory and Design of Charged Particle Beams John Wiley and Sons, Inc., New York, 1994, p. 196.

[12] Kirby, H., et al, "Development of Solid-State Induction Modulators for High PRF Accelerators" LLNL Report UCRL-JC-119582, July, 1995.

[13] Barnard, J., "Updated Recirculator Design Parameters", LLNL HIF Note No. 26, May, 1994.

[14] Barnard J., et al, "Physics Design and Scaling of Recirculating Induction Accelerators: From Benchtop Prototypes to Drivers" *Fusion Engineering and Design* 32-33 (1996) 247-258.

[15] Friedman, A., and Grote,D., "Three -Dimensional Particle Simulation of Heavy Ion Fusion Beams" *Phys. Fluids B*4 (7), July 1992

[16] "K^+ Source for the LLNL Recirculator Experiment", HIFAR Note 405, Lawrence Berkeley Laboratory, January, 1994.

[17] Wilson, M., et al, "Heavy Ion Fusion Impulse Injector Design, Construction, and Checkout" LLNL Report UCRL-ID-DRAFT, August, 1997.

[18] Karpenko, V., et al, "Mechanical Design of Recirculating Accelerator Experiments for Heavy Ion Fusion" LLNL Report UCRL-JC-119583, May, 1995.

[19] Reiser, M. Theory and Design of Charged Particle Beams John Wiley and Sons, Inc., New York, 1994

[20] Friedman, A., et al, "Construction of the First Recirculating Induction Accelerator" LLNL Proposal for a Director's Initiative LDRD Project, April, 1994.

[21] Reiser, M. Theory and Design of Charged Particle Beams John Wiley and Sons, Inc., New York, 1994, p. 363.

[22] Reiser, M. Theory and Design of Charged Particle Beams John Wiley and Sons, Inc., New York, 1994, p. 150.

[23] Reiser, M. Theory and Design of Charged Particle Beams John Wiley and Sons, Inc., New York, 1994, p. 360.

[24] Kapchinskij, I., and Vladmirskij, V. "Limitations of Proton Beam Current in a Strong Focusing Linear Accelerator Associated with Beam Space Charge", *Proc.*

of the International Symposium on High Energy Accelerators, (CERN, Geneva, 1959), p. 274.

[25] Reiser, M. Theory and Design of Charged Particle Beams John Wiley and Sons, Inc., New York, 1994, Chapter 6.

[26] "Possible Emittance Increase Through Filamentation Due to Space Charge in Continuous Beams" *IEEE Trans. Nucl. Sci.***NS-18**, No. 3, 1101, 1971.

[27] Sacherer, F., "RMS Envelope Equations with Space Charge" *IEEE Trans. Nucl. Sci.***NS-18**, No. 3, 1105, 1971.

[28] Paul, A. et al, "Probing the Electron Distribution Inside the ATA Beam Pulse", *Nucl. Instrum. Methods Phys. Rev. A*300 (1991) 137-150.

[29] Barnard, J., "Emittance Growth from Rotated Quadrupoles in Heavy Ion Accelerators", 1995 Partcle Accelerator Conference, Dallas, TX.

[30] Lawson, J., Lapostelle, P., Gluckstern, R., "Emittance, Entropy, and Information" *Particle Accelerators*, 1973, Vol. 5, pp. 61-65

[31] Tiefenback, M., "Space-Charge Limits on the Transport of Ion Beams in a Long Alternating Gradient System" LBL Report 22465, 1986, p. 38.

[32] Signal Analytics Corporation, IPLab Spectrum ver. 3.1 Software User's Guide, 1996, p. 160.

[33] Research Systems, Inc., Interactive Data Language ver. 4.0 Software User's Guide, 1995.

[34] Li, C., Petrasso, R., "Charged-particle stopping powers in inertial confinement fusion plasma" *Physical Review Letters* Vol. 70, No. 20, May, 1993, pp. 3059-62.

[35] Knoll, G., Radiation Detection and Measurement John Wiley and Sons, Inc., New York, 1989

[36] Ziegler, J., Biersack, J., Littmark, U., The Stopping and Range of Ions in Matter, Vol. 1 Pergamon Press, Inc., New York, 1985

[37] Galileo Electro-Optics Corporation Product Brochure, 1996

[38] Kaminsky, M., Atomic and Ionic Impact Phenomena on Metal Surfaces Academic Press, Inc., New York, 1965, p. 325.

[39] Princeton Instruments Corporation, PG-200 Programmable Pulse Generator Operating Manual, Ver. 2, rev. A, August 26, 1996.

[40] Princeton Instruments Corporation, ITE/CCD Detector Operating Manual, Ver. 1, rev. A, April 8, 1996.

[41] Princeton Instruments Corporation, ST-138 Camaera Controller Operating Manual, Ver. 3, rev. A, May 9, 1996, p. 25.

[42] Dragt, A., Neri, F., Rangarajan, G., General Moment Invariants for Linear Hamiltonian Systems, *Phys. Rev. A.*,Vol. 45, No. 4, 1992, 2572-2585.

[43] Halbach, K., "Conceptual Design of a Permanent Quadrupole Magnet with Adjustable Strength", *Nucl. Instrum. Methods Phys. Rev. A* 206 (1983) 353-354.

[44] Spall, J., "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation", *IEEE Trans. Automatic Control*, vol. 37, pp. 332-341.

[45] Spall, J., "A Model-Free Approach to Optimal Signal Light Timing for System-Wide Traffic Control", Proceedings of the 33rd Conference on Decision and Control, Lake Buena Vista, Fl., December 1994, pp. 1868-1875

[46] Spall, J., "A Neural Network Controller for Systems with Unmodeled Dynamics with Applications to Wastewater Treatment", *IEEE Trans. Systems, Man, and Cybernetics*, vol. 27, 1997.

[47] Lund, S., et al, "Numerical Simulation of Intense Beam Experiments at LLNL and LBL" 12th International Symposium on Heavy Ion Inertial Fusion, Heidelberg, Germany, September 24-27, 1997.

[48] Friedman, A., et al, "Progress Toward a Prototype Recirculating Induction Accelerator for Heavy Ion Fusion", 1995 Partcle Accelerator Conference, Dallas, TX.

[49] Wolfram, S., *Mathematica*, A System for Doing Mathematics by Computer Addison-Wesley Publishing Company, Inc., Redwood City, Ca, 1988

[50] Barnard, J., et al, "Emittance Growth in Heavy Ion Recirculators", C. R. Hoffman (ed.), Proc Linear Accelerator Conf., Ottawa, August 1992, AECL 10728, 1992, p. 229.

[51] Lee, E., Fessenden, T., Laslett, L., "Transportable Charge in a Periodic Alternating Gradient System" *IEEE Trans. Nucl. Sci.*NS-18, No. 32, 2489, 1985.

[52] BICRON Corporation Scintillator Product Brochure, 1990, p. B2.

[53] Spall, J., "Implementation of the Simultaneous Perturbation Algorithm for Stochastic Optimization", *American Statistician*, September, 1995.

[54] Private communication with S. Eylon, September, 1997

[55] Hopkins, H., LLNL HIF Note Number 96-5 "Physics design and Data Analysis for a Gata Beam Imager Diagnostic Instrument, July 2, 1996.