MARYLIE 3.0
Users' Manual

A Program for Charged Particle Beam Transport Based on Lie Algebraic Methods *

Alex J. Dragt

Center for Theoretical Physics
University of Maryland
College Park, Maryland 20742

Robert D. Ryne

Lawrence Berkeley National Laboratory
Berkeley, California 94720

David R. Douglas

Thomas Jefferson National
Accelerator Facility
Newport News, Virginia 23606

Filippo Neri
C. Thomas Mottershead

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

Étienne Forest

KEK Laboratory
Tsukuba, Ibaraki, Japan 305

Liam M. Healy

Naval Research Laboratory
Washington, DC 20375

Petra Schütt

Technische Universität Darmstadt
D-64289 Darmstadt, Germany

Dan T. Abell

RadiaSoft LLC
Boulder, Colorado 80303

October 2021

*Work supported in part by U.S. Department of Energy Grant DEFG02-96ER40949.

Disclaimer

MaryLie 3.0 consists of a Main Program and approximately 500 Subroutines. Together they comprise approximately 40,000 lines of what is intended, wherever practical, to be standard FORTRAN 77 code. As time has permitted, considerable care has been expended in checking many of these subroutines, and some are at least partially self checking. (For example, the product of two one-meter drift maps should be a two-meter drift map.) Other MaryLie features and their associated subroutines are still under development.

Wherever possible, the user should also make both independent and internal consistency checks of his/her own. Programs should not be run blindly. There is no substitute for understanding. The authors would appreciate receiving information concerning any detected errors or difficulties. A form for this purpose is provided in Chapter 12.

In any case, the MaryLie 3.0 program is provided "as is" without any warranty of any kind. No warranties, expressed or implied, are made that the program and its procedures are free of error, or are consistent with any particular standard of merchantability or fitnesss for a particular purpose, or that they will meet your requirements for any particular application. They should not be relied on for solving a problem whose incorrect solution could result in injury to a person or loss of property. If you use the program in such a manner, it is at your own risk. The authors and their agencies disclaim all liability for direct, incidental, or consequential damage resulting from your use of the program.

Acknowledgement

Initial work on the use of Lie methods in the general area of Nonlinear Dynamics began when the first author (AJD) was a sabbatical visitor in spring 1973 at the Institut des Hautes Etudes Scientifiques (IHES) in the Paris suburb Bures-sur-Yvette. He is indebted to Louis Michel and the IHES for this opportunity and their fine hospitality. Some years later, during a sabbatical visit to the Accelerator Technology Division of the Los Alamos National Laboratory, he realized that Lie methods could be applied to the computation and analysis of Charged Particle Beam Transport. He is is indebted to Richard Cooper and all the other accelerator physicists at Los Alamos for their hospitality then, and for their continued support and encouragement over the ensuing years. He is also grateful to the University of Maryland for its continued support. Finally, research on Lie algebraic methods has been supported by the United States Department of Energy since 1980. All the authors are grateful to DOE and the American people for their nurture of fundamental and applied science.

Assistance

Several have made major contributions to the preparation of this manual and its exhibits including Rachel Needle, Glenn Vanderwoude, and Maura and Patrick Roberts.

Revision Record

June 1998, February 1999, April 1999, October 1999, March 2000, May 2000, October 2002, May 2003, December 2003.

Availability

See section 1.5.

©2003 Alex J. Dragt. All rights reserved.

Contents

1 Introduction to MaryLie **1**
1.1 Brief Program Description . . . 1
1.2 Program Performance and Accuracy . . . 1
1.3 Future Versions of MaryLie . . . 2
1.4 Auxiliary Programs . . . 2
1.4.1 Programs for Preparation of MaryLie Input . . . 3
1.4.2 Programs for Processing of MaryLie Output . . . 3
1.5 Program Availability . . . 3

2 Simple Examples of Use of MaryLie **5**
2.1 Overview . . . 5
2.2 Simple Spot Forming System . . . 5
2.3 Simple Imaging System . . . 12
2.4 Simple Spectrometer . . . 19
2.5 Small Static Storage Ring . . . 29
2.6 Small Storage Ring with Bunchers . . . 49
2.7 Large Storage Ring . . . 66

3 Brief Summary of Theoretical Tools **76**
3.1 Introduction . . . 76
3.2 Hamiltonian Dynamics . . . 76
3.3 Poisson Brackets . . . 79
3.4 Transfer Maps . . . 80
3.5 The Symplectic Condition . . . 81
3.6 Taylor Expansions . . . 82
3.7 Overview of the Lie Algebraic Approach . . . 82
3.8 Lie Operators and Lie Transformations . . . 82
3.9 Factorization Theorem . . . 84
3.10 Polynomial Labeling Scheme . . . 85
3.11 Ingredients of a Lie Algebraic Code . . . 87
3.11.1 Element Library . . . 87
3.11.2 Concatenation Subroutines . . . 89
3.11.3 Ray Trace Subroutines . . . 90
3.11.4 Analysis Subroutines . . . 91
3.11.5 Bookkeeping Subroutines . . . 92
3.11.6 Procedures and Fitting and Optimization Subroutines . . . 92

4 Input/Output Preparation and Formats 93
4.1 General Considerations . . . 93
4.1.1 Units and Constants . . . 93
4.1.2 Scaling and Dimensionless Variables for Phase Space . . . 93
4.2 Input and Output Files . . . 95
4.3 Preparation of Files . . . 96
4.3.1 The Program PREP . . . 96
4.3.2 Free Format Input . . . 96
4.4 Input File Structures . . . 97
4.4.1 File 11 (Master Input File) . . . 97
4.4.2 Initial Conditions File for Ray Traces or Tracking . . . 99
4.4.3 Matrix and Polynomial Input File . . . 100
4.4.4 Input Data for Random Elements . . . 100
4.4.5 Input Data for Procedures and Fitting and Optimization Commands 101
4.5 Output File Structures . . . 101
4.5.1 Files 6 and 12 . . . 101
4.5.2 Final Conditions from a Ray Trace or Tracking . . . 101
4.5.3 Matrix and Polynomial Output File . . . 101

5 Format of MaryLie Master Input File & Use of PREP 102
5.1 General Instructions . . . 102
5.1.1 PREP Input . . . 102
5.1.2 PREP Output . . . 102
5.2 Use of Help Feature . . . 102
5.3 Input From External File . . . 103
5.4 Modes, Editors, and Related Commands . . . 103
5.5 Comment (cmts) Mode . . . 105
5.5.1 Format of #comment Component . . . 105
5.5.2 Sample Conversation in Comments Mode . . . 105
5.5.3 Comment Editor . . . 106
5.5.4 Interspersed Comments . . . 106
5.6 Beam Mode . . . 106
5.6.1 Beam Editor . . . 108
5.6.2 Sample Conversation in Beam Mode . . . 108
5.6.3 Format of #beam Component . . . 109
5.7 Menu Mode . . . 110
5.7.1 Sample Conversation in Menu Mode . . . 115
5.7.2 Format of #menu Component . . . 117
5.7.3 Menu Editor . . . 118
5.8 Line Mode . . . 119
5.8.1 Sample Conversation in Line Mode . . . 119
5.8.2 Format of #lines Component . . . 120
5.8.3 Multiplicative Notation . . . 121
5.8.4 Line Editor . . . 122

5.9 Lump Mode . . . 122
5.9.1 Sample Conversation in Lump Mode . . . 124
5.9.2 Format of the #lumps Component . . . 124
5.9.3 Multiplicative Notation . . . 126
5.9.4 Lump Editor . . . 126
5.10 Loop Mode . . . 127
5.10.1 Sample Conversation in Loop Mode . . . 127
5.10.2 Format of the #loops Component . . . 128
5.10.3 Multiplicative Notation . . . 129
5.10.4 Loop Editor . . . 130
5.11 Labor Mode . . . 131
5.11.1 Sample Conversation in Labor Mode . . . 132
5.11.2 Format of the #labor Component . . . 133
5.11.3 Multiplicative Notation . . . 134
5.11.4 Labor Editor . . . 134
5.12 Use of the Summary Command in PREP . . . 134
5.13 Use of the File Command in PREP . . . 136
5.14 Terminating and Resuming a PREP Run . . . 136

6 Catalog of Beam-Line Elements 139
6.1 Drift . . . 142
6.2 Normal Entry and Exit (Sector) Bending Magnet . . . 143
6.3 Parallel Faced (Rectangular) Bending Magnet (Symmetric) . . . 159
6.4 General Bending Magnet . . . 163
6.5 Plane Rotation . . . 169
6.6 Body of a General Bending Magnet . . . 171
6.7 Fringe Fields for Bending Magnets (Hard Edge) . . . 179
6.8 Combined Function Bending Magnet (Normal Entry and Exit) . . . 182
6.9 Magnetic Quadrupole . . . 194
6.10 Magnetic Sextupole . . . 200
6.11 Magnetic Octupole . . . 208
6.12 Electrostatic Octupole . . . 216
6.13 Short Radio Frequency Cavity . . . 217
6.14 Axial Rotation . . . 218
6.15 Twiss Matrix . . . 222
6.16 Thin Low Order Multipole . . . 226
6.17 “Compressed” Low Order Multipole . . . 230
6.18 J Map . . . 234
6.19 Random Elements . . . 235
6.20 User Specified Subroutine that acts on phase space data . . . 238
6.21 User Specified Subroutine that produces or acts on a map . . . 239
6.22 Dispersion Matrix . . . 240
6.23 Solenoid . . . 241

6.24 Combined Function Magnetic Quadrupole . . . 251
6.25 Marker . . . 255
6.26 Data Point . . . 256
6.27 Rare Earth Cobalt (REC) Quadrupole Multiplet . . . 257
6.28 Space . . . 276
6.29 Change or Write Out Values of Parameters for Combined Function Dipole . . . 277
6.30 Random Map . . . 278
6.31 Arc . . . 281

7 Catalog of Simple Commands 283
7.1 Halt Execution . . . 286
7.2 Perform a Ray Trace . . . 287
7.3 Circulate Around a Loop . . . 291
7.4 Print the Contents of the Master Input File . . . 293
7.5 Input of a Transfer Map . . . 294
7.6 Output of a Transfer Map . . . 296
7.7 Print Transfer Map . . . 297
7.8 Replace with Identity Map . . . 300
7.9 Invert a Map . . . 301
7.10 "Transpose" a Map . . . 302
7.11 Reverse a Map . . . 303
7.12 Reverse Factorize a Map . . . 304
7.13 Mask a Map . . . 305
7.14 Symplectify a Matrix . . . 306
7.15 Square a Map . . . 307
7.16 Store Transfer Map . . . 308
7.17 Get Transfer Map . . . 309
7.18 Aperture Beam with Rectangular Aperture . . . 310
7.19 Aperture Beam with Elliptic Aperture . . . 311
7.20 Window a Beam . . . 312
7.21 Write History . . . 313
7.22 Filter Transfer Map . . . 314
7.23 Close Files . . . 316
7.24 Open Files . . . 317
7.25 Parameter Set Specification . . . 319
7.26 Random Parameter Set Specification . . . 320
7.27 Number Lines in a File . . . 321
7.28 Write Out Parameter Values in a Parameter Set . . . 322
7.29 Write Out Execution Time . . . 323
7.30 Change Drop File . . . 324
7.31 Ring Bell at Terminal . . . 325
7.32 Write Out Value of Merit Function . . . 326
7.33 Write Contents of a Loop . . . 327

7.34 Pause . . . 331
7.35 Change or Write Out Values of Infinities . . . 332
7.36 Change or Write Out Values of Zeroes . . . 333
7.37 Twiss Polynomial . . . 334
7.38 Dispersion Polynomial . . . 335
7.39 Change or Write Out Beam Parameters . . . 336
7.40 Dimensions . . . 340
7.41 Write Out Contents of the UCALC Array . . . 341
7.42 Window a Beam in All Planes . . . 343
7.43 Path Length Information . . . 344
7.44 Show Contents of Arrays . . . 347

8 Catalog of Advanced Commands 349
8.1 Closed Orbit Data . . . 351
8.2 Twiss Analyze Static Map . . . 363
8.3 Twiss Analyze Dynamic Map . . . 381
8.4 Resonance Analyze Static Map . . . 390
8.5 Resonance Analyze Dynamic Map . . . 396
8.6 Translate Basis . . . 403
8.7 Compute Exponential . . . 404
8.8 Static Normal Form Analysis . . . 405
8.9 Dynamic Normal Form Analysis . . . 413
8.10 Static Invariant Analysis . . . 423
8.11 Dynamic Invariant Analysis . . . 429
8.12 Compute Power of Static Normal Form . . . 443
8.13 Compute Power of Dynamic Normal Form . . . 445
8.14 Get Buffer Contents . . . 447
8.15 Fourier Analyze Static Map . . . 448
8.16 Fourier Analyze Dynamic Map . . . 449
8.17 Apply Map to a Function or Moments . . . 450
8.18 Multiply a Polynomial by a Scalar . . . 459
8.19 Add Two Polynomials . . . 460
8.20 Multiply Two Polynomials . . . 461
8.21 Poisson Bracket Two Polynomials . . . 462
8.22 Polar Decomposition of a Map . . . 463
8.23 Evaluate a Polynomial . . . 465
8.24 Compute Chromatic Expansion . . . 467
8.25 Transfer Quadratic Moments . . . 475
8.26 Select Quantities . . . 476
8.27 Write Selected Quantities . . . 478
8.28 Change Tune Range . . . 480
8.29 Apply Script N Inverse . . . 481
8.30 Compute Power of Nonlinear Part . . . 483
8.31 Check Symplectic Condition . . . 484

8.32 Polynomial Scalar Product . . . 485
8.33 Compute Matrix Norm . . . 487
8.34 Generate Beam . . . 488
8.35 Translate Initial Conditions . . . 491
8.36 Principal Planes Analysis . . . 492
8.37 Moment Analysis . . . 493
8.38 Compute Geometry of a Loop . . . 508
8.39 Copy File to Working Array . . . 517
8.40 Merge Files . . . 518
8.41 Compute Logarithm of Normal Form . . . 519

9 Catalog of Procedures and Fitting and ··· 520
9.1 Begin Inner Procedure . . . 522
9.2 Begin Outer Procedure . . . 523
9.3 Terminate Inner Procedure . . . 524
9.4 Terminate Outer Procedure . . . 525
9.5 Specify Quantities to be Fit or Optimized and Set Target Values . . . 526
9.6 Specify Quantities to be Varied . . . 532
9.7 Carry Out Fitting Operation . . . 535
9.8 Minimize Sum of Squares Optimization . . . 538
9.9 General Optimization . . . 540
9.10 Merit Function (Sum of Squares) . . . 542
9.11 Merit Functions (User Written) . . . 543
9.12 Constraints (User Written) . . . 544
9.13 Compute Gradient Matrix . . . 546
9.14 Scan Parameter Space . . . 550
9.15 Capture Parameter Set . . . 551
9.16 Free Parameter Set . . . 556
9.17 Change or Write Out Values of Flags and Defaults . . . 557
9.18 Reset Menu Entries . . . 559
9.19 Spare . . . 562

10 Additional Examples of Use of MaryLie 563
10.1 Overview . . . 563
10.2 Tunes, Chromaticities, and Tune Foot Prints . . . 564
10.2.1 Fitting Tunes and Chromaticities for a Storage Ring . . . 564
10.2.2 Computation of Full Tunes . . . 574
10.2.3 Computation of Tune Foot Print . . . 584
10.3 Octupole Correction & Optimization of a Quadrupole ··· . . . 590
10.3.1 Fitting Three Quadrupole and Three Octupole Strengths . . . 590
10.3.2 Fitting and Optimizing Seven Octupole Strengths . . . 605
10.3.3 Optimizing Using the MSS Optimizer . . . 625
10.4 Sextupole Correction of a Solenoidal Spot Forming ··· . . . 642
10.4.1 Fitting Solenoid and Sextupole Strengths . . . 642

10.4.2 Scanning the Sextupole Strength 662
10.5 Fitting at Multiple Lattice Locations 667
10.6 Construction of Third-Order Achromats 678
10.6.1 Construction of Ordinary Third-Order Achromat 678
10.6.2 Construction of Complete Third-Order Achromat 688
10.7 Study of Final Focus Test Beam Facility 704
10.8 Calculation of Dynamic Aperture for a Storage Ring 728
10.8.1 Dynamic Aperture of Tevatron with Strong Distortion Sextupoles . . 729
10.8.2 History of Lost Particle 761
10.9 Production of Lattice Function Plots 773
10.10 Production of Ray Plots 790
10.11 Production of Beam Moment Plots 800
10.12 Production of Composite Plots with Geometry 815
10.12.1 Lattice Function Plots with Geometry 815
10.12.2 Ray Plots with Geometry 828
10.12.3 Beam Moment Plots with Geometry 838
10.13 Production of Layout Drawings 850

11 References 854
11.1 Fundamental Constants 854
11.2 Other Charged Particle Beam Transport Codes 854
11.3 Accelerator Handbooks 854
11.4 Conventional Accelerator Theory 854
11.5 General Lie Algebraic References 855
11.6 Lie Algebraic/Map Based Accelerator Theory 855
11.7 Applications of Lie Algebraic/Map Based Methods to Specific Problems in Accelerator Physics 856
11.8 Rare Earth Cobalt Quadrupoles 856
11.9 Combined Function Bend 856
11.10 Solenoid 856
11.11 MaryLie (Giorgilli) Indexing Scheme 856
11.12 Random Number Generation 856

12 Comments and Suggestion Form 857

13 Quick Reference Guide to Elements, Commands, ··· 858
13.1 Alphabetical Listing of All Type-Code Mnemonics 858
13.2 Element Type-Code Mnemonics (Alphabetical Order) 864
13.3 Element Type-Code Mnemonics (Functional Order) 866
13.4 Simple Command Type-Code ··· (Alphabetical Order) 868
13.5 Simple Command Type-Code Mnemonics (Functional Order) 870
13.6 Advanced Command Type-Code ··· (Alphabetical Order) 872

13.7 Advanced Command Type-Code ··· (Functional Order) 874
13.8 Procedures and Fitting and Optimization ··· (Alphabetical Order) 876
13.9 Procedures and Fitting ··· (Functional Order) 877
13.10 Input File Functions and Formats 878
13.11 Output File Functions and Formats 879
13.12 Warning and Error Messages 880

14 Tables of Indices, Exponents, and Basis Elements 881
14.1 MaryLie Indices for Monomials in Cartesian Basis 883
14.2 MaryLie Indices for Polynomials in Static Resonance Basis 886
14.3 MaryLie Indices for Polynomials in Dynamic Resonance Basis 892

15 Index 898

Chapter 1

Introduction to MaryLie

1.1 Brief Program Description

MaryLie 3.0 is the first version of the MARYland LIE algebraic particle beam transport and tracking program. It is named after Henrietta Maria, wife of England's King Charles I, and Sophus Lie. Lord Baltimore called the English colony that was to become the state of Maryland "Terra Maria". Sophus Lie was the discoverer of Lie algebras and Lie groups. The purpose of MaryLie is to aid in the design and evaluation of both linear charged particle beam transport systems and circulating storage rings. The program employs algorithms based on a Lie algebraic formulation of charged particle trajectory calculations, and is able to compute transfer maps for and trace rays through single or multiple beam-line elements. This is done for the full 6-dimensional phase space *without* the use of numerical integration or traditional matrix methods. All nonlinearities, including chromatic effects, through third (octupole) order are included. Thus MaryLie 3.0 includes effects one order higher than those usually handled by existing matrix-based programs. In addition, MaryLie is exactly symplectic (canonical) through all orders. The version of MaryLie described in this manual consists of approximately 40,000 lines of code organized into approximately 500 routines.

1.2 Program Performance and Accuracy

MaryLie may be used both for particle tracking around or through a lattice and for analysis of linear and nonlinear lattice properties. It may also be used to transport moments without the need for particle tracking. When used for tracking, it is both versatile and extremely fast. Tracking can be performed element by element, lump by lump, or any mixture of the two. (A lump is a collection of elements combined together and treated by a single transfer map.) The speed for element by element tracking is comparable to that of other tracking codes. When collections of elements can be lumped together to form single transfer maps, tracking speeds can be orders of magnitude faster.

Major portions of MaryLie 3.0 have been benchmarked against numerical integration. All were verified to be exact through third order. When properly used MaryLie has an absolute accuracy of four or more significant figures. (In many cases, depending on the amplitude of betatron oscillations, the absolute accuracy is six or more significant figures.) That is, MaryLie has an accuracy comparable to or better than the accuracy with which

accelerator and beamline components are currently measured.

MaryLie also has extremely powerful analytic tools. They include the calculation of tunes and first and second-order chromaticities and anharmonicities (dependence of tunes on betatron amplitudes) and first, second, and third-order temporal dispersion (dependence of closed-orbit transit time on energy, sometimes called the phase-slip factor, and related to momentum compaction, the dependence of the closed-orbit length on energy); first, second, and third-order ordinary dispersion, and all other linear lattice functions and their energy dependence through second order; nonlinear lattice functions; nonlinear phase-space distortion; transfer map normal forms; nonlinear resonance driving terms; nonlinear invariants; expected Fourier coefficients for tracking data; and moment data including eigenemittances and high-order moments.

To facilitate actual machine design, MaryLie has extensive fitting, scanning, and optimization routines. These routines make it possible to fit, scan, or optimize essentially all quantities that can be computed by MaryLie including those computed in user written subroutines. MaryLie also provides full geometrical information including the orientation and location of beamline elements in 3-dimensional space.

Finally, MaryLie can be used to give an explicit representation for the linear and nonlinear properties of the total transfer map of a system. This information can be used to evaluate or improve the optical quality of a single pass system such as a beam transport line or linear collider. For a circulating system such as a storage ring, it is hoped that the explicit knowledge in this case of the one-turn transfer map can eventually be used to predict the dynamic aperture without the need for extensive long-term tracking.

1.3 Future Versions of MaryLie

Because MaryLie 3.0 is the first version of a Lie algebraic code, it does not have all the features one might possibly desire. For example, MaryLie 3.0 does not treat beam element placement, alignment, and powering errors.

Future versions of MaryLie will have features not found in this first version. Possible improvements include the treatment of additional beam-line elements; handling of magnet placement, alignment, and powering errors; acceleration; space-charge effects; etc. These future versions will become available later as MaryLie 3.1, MaryLie 3.2, etc.

One of the advantages of the Lie algebraic method is that it is readily extended, in principle, to higher orders. It is expected that it will eventually be possible to go to fifth order. This will be the subject of MaryLie 5.0 and its later versions, 5.1, 5.2, etc. Even higher order calculations appear possible, but will require considerably more work. There is no insurmountable difficulty in treating idealized beam-line elements in higher order. The major problem is the realistic treatment of fringe-field and magnet error effects, etc., which is essential for high-order expansions to make sense.

1.4 Auxiliary Programs

MaryLie has been written to make use of several external files both for input and output. This feature makes it possible to design and use other computer programs both to prepare

input for MaryLie and to postprocess MaryLie output. Currently several such programs exist or are in preparation. A brief list of these programs is given below. Users may also wish to write additional programs of their own.

1.4.1 Programs for Preparation of MaryLie Input

- PREP. This program may be used to prepare the Master Input File for MaryLie. Its operation is described in later chapters of this manual.
- MADMARY and MARYMAD. MADMARY takes a lattice, as specified in MAD format, and translates it into the current MaryLie format. MARYMAD does the reverse.
- GENMAP. In normal operation, MaryLie uses an internal library of idealized beam-line elements. However, MaryLie also has the capability to read in and use externally specified transfer maps. GENMAP refers to a series of programs that take as input real (calculated or measured) magnetic and/or electric field data for some beam-line element, and then generate by numerical integration the transfer map describing motion through that element. This transfer map can subsequently be used by MaryLie. In this way, it is possible to use MaryLie to treat real beamline elements including the effect of fringe fields, magnetic imperfections, etc. MaryLie also has a built-in GENMAP routines that are used for various purposes. See sections 6.8, 6.23, 6.27, and 8.7. Finally, the user can in principle write routines of his/her own that call the internal GENMAP routines. See section 6.21

1.4.2 Programs for Processing of MaryLie Output

- GENERAL PLOTTING. When performing ray traces, MaryLie reads phase-space initial conditions from an external file and writes the final phase-space coordinates on another external file. MaryLie also writes various other kinds of data to external files. All these quantities can be plotted against each other to make phase-space or other kinds of plots by use of any of the commonly available plotting programs that read data files in columnar form. See sections 2.2 through 2.7, 10.3, 10.4, 10.7, 10.8, 10.9, 10.11, and 10.12.
- POSTER is a general purpose Post-Script plotting program written by C. Thomas Mottershead. As such, it can be used to produce the general kinds of plots described above. In addition, POSTER can read files produced by the MaryLie *geom* command. See section 8.38. With this capacity POSTER can be used to produce composite plots that include a schematic representation of the underlying beamline, and to produce beamline layout drawings. See sections 10.13 and 10.14.

1.5 Program Availability

Both MaryLie and PREP are written in Fortran 77. Serial scalar versions are currently available for any computer that supports Fortran 77. In addition vectorized and/or par-

allel versions of MaryLie (and serial versions of PREP) are available for certain vectorized/parallel machines.

All of MaryLie 3.0 is believed to conform to Fortran 77 standards except for the use of *include* statements (which are supported by most Fortran 77 computers) and the timing routine *cputime*. The latter is invoked by the MaryLie command *time* (see section 7.29). It is machine dependent, and must be set up by the user if it is to be employed.

Chapter 2

Simple Examples of Use of MaryLie

2.1 Overview

Subsequent chapters of this manual are devoted to an exposition of the theory and application of MaryLie. The purpose of this chapter is to intrigue the reader with a few simple, sometimes whimsical, examples of the use of MaryLie. At this stage, the reader may read as much or as little of these examples as desired. It is suggested that the reader look at the first example, and then at any other examples of interest. For convenience, the examples and their section numbers are listed below. Further examples are given in Chapter 10. It will be seen that MaryLie is essentially a *programming language* that enables the user to simulate, analyze, and manipulate a vast variety of both beamlines and circular machines.

2.2 Simple Spot Forming System 5
2.3 Simple Imaging System 12
2.4 Simple Spectrometer 19
2.5 Small Static Storage Ring 29
2.6 Small Storage Ring with Bunchers 49
2.7 Large Storage Ring 66

The first three examples are of single pass systems, and the remaining examples are of circulating storage systems. Each of these examples could be a research project in its own right. In general, this research has not been done. Consequently, the parameter values employed may not be optimal, and are not necessarily even of physical interest.

The MaryLie 3.0 input and output for these examples will be given in detail, and may be of interest to the reader for later reference. However, only the highlights of the calculations will be described. After the reader has read the remainder of the manual, she or he should be able to understand the examples in detail.

2.2 Simple Spot Forming System

Consider the simple spot forming system shown in figure 2.2.1 below. It consists of a quadrupole triplet followed by a drift space. The quadrupole strengths and the drift lengths have been selected in such a way that (in linear paraxial approximation) incoming parallel rays are brought to a final point focus. Such a system could be the basis of a microprobe.

Figure 2.2.1: Schematic of Simple Spot Forming System.

How well does this system work when account is taken of nonlinear effects? Shown below is the Master Input File for a MaryLie 3.0 run set up to trace rays through this system.

```
#comment
 Exhibit 2.2.1.
 This is a study of a simple spot forming system.
#beam
  1.035274408519500
 5.3289019605700000E-02
  1.000000000000000
  1.000000000000000
#menu
 drs      drft
  0.500000000000000
 drl      drft
   20.0026000000000
 hfq      quad
   1.50000000000000      8.630000000000000E-02   1.00000000000000
   1.00000000000000
 hdq      quad
   3.00000000000000     -8.289450000000000E-02   1.00000000000000
   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 rays     rt
   13.0000000000000       14.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000      0.000000000000000E+00
```

```
 end      end
#lines
 trip
     1*hfq         1*drs         1*hdq         1*drs         1*hfq
 spot
     1*trip        1*drl
#lumps
#loops
#labor
    1*fileout
    1*spot
    1*rays
    1*end
```

The reader will observe that the Master Input File is organized into seven components. They are *#comments*, *#beam*, *#menu*, *#lines*, *#lumps*, *#loops*, and *#labor*, respectively. The *#comments* component allows the user to write comments about the system under study, the calculations to be performed, etc. The *#beam* component specifies the design magnetic rigidity (in Tesla-meters) and the corresponding relativistic β and γ factors for the beam to be "transported" by the system. It also specifies $|q/e|$, the absolute charge in units of e, and the scale length to be employed for the spatial components of phase-space variables. See Section 5.6.3. For this example, the incoming beam is taken to consist of 50 MeV protons, and the scale length is taken to be 1 meter.

The *#menu* component is a list of user specified beamline elements and commands. Each item on the menu consists of a user specified name followed by a MaryLie type code mnemonic. The numbers following each menu item specify lengths, strengths, and other required parameters. For example, *drs* is a short drift of length .5 meters. Similarly, *hfq* is a horizontally focusing quadrupole of length 1.5 meters and strength .0863 Tesla/meter. The two remaining parameters for this quadrupole (both set equal to +1 in this example) indicate that it is to be treated as having hard-edge leading and trailing fringe fields. Finally, *fileout*, *mapout*, *rays*, and *end* are four commands.

The *#lines* component contains a list of user specified names for collections of elements and/or commands drawn from the menu. In this case, *trip* refers to the quadrupole triplet system, and *spot* refers to the entire spot forming system. See figure 2.2.1.

In this simple example, the *#lumps* and *#loops* components are empty, and are not used. Their function will be described in later examples.

Finally, the *#labor* component specifies the system to be considered, and the actual operations and calculations to be performed. In this case, *fileout* causes the Master Input File to be printed out; *spot* causes the calculation of the transfer map for the entire system; *rays* produces a trace of rays through the system; and *end* signifies the end of the calculation. Note that so far, in this example, the command *mapout* is not actually used since it neither appears nor is referenced in *#labor*.

When MaryLie 3.0 is run using the Master Input File just described, the following output is produced:

```
#comment
 Exhibit 2.2.2.
```

```
 This is a study of a simple spot forming system.
#beam
  1.035274408519500
 5.3289019605700000E-02
  1.000000000000000
  1.000000000000000
#menu
 drs      drft
  0.500000000000000
 drl      drft
   20.0026000000000
 hfq      quad
   1.50000000000000      8.630000000000000E-02   1.00000000000000
   1.00000000000000
 hdq      quad
   3.00000000000000     -8.289450000000000E-02   1.00000000000000
   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 rays     rt
   13.0000000000000       14.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000      0.000000000000000E+00
 time     time
  1.00000000000000       12.0000000000000       3.00000000000000
 end      end

#lines
 trip
     1*hfq         1*drs         1*hdq         1*drs         1*hfq
 spot
     1*trip        1*drl
#lumps
#loops
#labor
    1*fileout
    1*spot
    1*rays
    1*time
    1*end

 execution time in seconds =   6.48
```

Note that the Master Input File is printed as a result of the presence of the command *fileout* in *#labor*. However, relatively little else is printed.

The major task of this run is to trace rays. This is accomplished by the presence of the *rays* command in *#labor*. When this command is encountered, initial conditions are read in from a previously prepared external file. The total transfer map for the system is then applied to each initial condition, and the results are written on another external file. The first few lines of each of these files are shown below:

First few lines of initial condition file for ray traces

X	P_x	Y	P_y	τ	P_τ
0.1000E-01	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00	0.0000E+00
0.9980E-02	0.0000E+00	0.6279E-03	0.0000E+00	0.0000E+00	0.0000E+00
0.9921E-02	0.0000E+00	0.1253E-02	0.0000E+00	0.0000E+00	0.0000E+00
0.9823E-02	0.0000E+00	0.1874E-02	0.0000E+00	0.0000E+00	0.0000E+00

First few lines of final condition file resulting from ray traces

X	P_x	Y	P_y	τ	P_τ
-1.61053E-07	-4.04394E-04	0.00000E+00	0.00000E+00	9.78473E-06	0.00000E+00
-1.63302E-07	-4.03585E-04	-4.72204E-08	-2.75206E-05	9.81295E-06	0.00000E+00
-1.69989E-07	-4.01200E-04	-9.38035E-08	-5.49185E-05	9.89883E-06	0.00000E+00
1.80829E-07	-3.97237E-04	-1.39245E-07	-8.21367E-05	1.00411E-05	0.00000E+00

There are six numbers in each line corresponding to the coordinates in 6-dimensional phase space. The first four coordinates describe the transverse phase space. The last two describe the differential time of flight τ and its conjugate momentum P_τ (which, as will be described later, is related to energy deviation).

As indicated in figure 2.2.1, the incoming rays are selected to be *parallel* to the optical axis and to lie on two cylinders having radii of 1.0 cm and 0.5 cm respectively. Consequently, the P_x and P_y entries in the initial condition file (those entries which describe transverse momenta) are all zero. Figure 2.2.2 below shows the result of plotting the first and third columns (the X and Y phase space coordinates) of the initial condition file. Altogether, 200 rays are shown. As advertized, the incoming rays do indeed lie on two cylinders.

If the performance of the spot forming system were perfect, the X and Y entries in the final condition file (resulting from the ray trace) would all be zero. Inspection of the first few lines of the final condition file shows that the X and Y entries are indeed small. However, they are not zero. In fact, their finite value is due to the presence of third-order "spherical type" aberrations. These aberrations come primarily from nonlinear effects associated with quadrupole fringe fields.

Figure 2.2.3 displays the result of plotting the X and Y entries of the final condition file. It shows that the incoming cylinders of parallel rays are not focused to a single point spot. Rather, they produce two "hour glass" shaped figures. The inner and outer hour glasses come from the inner and outer cylinders respectively.

Examination of the figure shows that the relative sizes of the two hour glasses is about 8 to 1. By contrast, the ratio of the two cylinder sizes in figure 2.2.2 is 2 to 1. This is just what is to be expected from third-order aberration effects since $2^3 = 8$.

The relative importance of the various aberrations (through third order) can be examined in quantitative detail by printing out the Lie algebraic representation of the total transfer map for the spot forming system. This can be accomplished by including the command *mapout* in *#labor*. Shown below is the result of such a MaryLie 3.0 run.

Figure 2.2.2: Plot of Initial Conditions for Incoming Rays of Spot Forming System.

Figure 2.2.3: Focal spot pattern produced by two incoming cylinders of parallel rays.

```
 #comment
  Exhibit 2.2.3.
  This is a study of a simple spot forming system.
 #beam
   1.035274408519500
  5.3289019605700000E-02
   1.000000000000000
   1.000000000000000
 #menu
  drs      drft
   0.500000000000000
  drl      drft
    20.0026000000000
  hfq      quad
    1.50000000000000      8.630000000000000E-02   1.00000000000000
    1.00000000000000
  hdq      quad
    3.00000000000000     -8.289450000000000E-02   1.00000000000000
    1.00000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  rays     rt
    13.0000000000000       14.0000000000000       5.00000000000000
    1.00000000000000       1.00000000000000      0.000000000000000E+00
  end      end
 #lines
  trip
      1*hfq         1*drs         1*hdq         1*drs         1*hfq
  spot
      1*trip        1*drl
 #lumps
 #loops
 #labor
     1*fileout
     1*spot
     1*mapout
     1*end

 matrix for map is :

 7.43444E-07  2.47288E+01  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00
-4.04387E-02  8.08880E-01  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  7.22568E-07  2.28173E+01 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00 -4.38264E-02  8.76642E-01 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 1.00000E+00 2.46784E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00 1.00000E+00

 nonzero elements in generating polynomial are :

  f( 33)=f( 20 00 01 )=-0.45766053436970D-01
```

```
f( 38)=f( 11 00 01 )= 0.14345982412058D+01
f( 53)=f( 02 00 01 )=-0.59832851277335D+02
f( 67)=f( 00 20 01 )=-0.10955092926799D+00
f( 70)=f( 00 11 01 )= 0.46810944831432D+01
f( 76)=f( 00 02 01 )=-0.88881809079482D+02
f( 83)=f( 00 00 03 )=-0.39290849771004D+03
f( 84)=f( 40 00 00 )=-0.16965902443077D-02
f( 85)=f( 31 00 00 )= 0.16215419632088D+00
f( 90)=f( 22 00 00 )=-0.59484282516103D+01
f( 95)=f( 20 20 00 )=-0.16646388617124D-01
f( 96)=f( 20 11 00 )= 0.75198181482571D+00
f( 99)=f( 20 02 00 )=-0.85538134290315D+01
f(104)=f( 20 00 02 )=-0.19073142049137D+00
f(105)=f( 13 00 00 )= 0.99159110467675D+02
f(110)=f( 11 20 00 )= 0.80660002769129D+00
f(111)=f( 11 11 00 )=-0.36679597123554D+02
f(114)=f( 11 02 00 )= 0.41993959894126D+03
f(119)=f( 11 00 02 )= 0.78836842993921D+01
f(140)=f( 04 00 00 )=-0.63693341340996D+03
f(145)=f( 02 20 00 )=-0.10046981396547D+02
f(146)=f( 02 11 00 )= 0.45984530773432D+03
f(149)=f( 02 02 00 )=-0.53047221347237D+04
f(154)=f( 02 00 02 )=-0.30942962923253D+03
f(175)=f( 00 40 00 )=-0.51353235082385D-02
f(176)=f( 00 31 00 )= 0.46634415070127D+00
f(179)=f( 00 22 00 )=-0.15935461945328D+02
f(184)=f( 00 20 02 )=-0.58685382925631D+00
f(185)=f( 00 13 00 )= 0.24279242171843D+03
f(190)=f( 00 11 02 )= 0.32524382342626D+02
f(195)=f( 00 04 00 )=-0.13944546806254D+04
f(200)=f( 00 02 02 )=-0.60171761738398D+03
f(209)=f( 00 00 04 )=-0.15330379170756D+04
```

Again, the Master Input File is printed because the command *fileout* is still in *#labor*. In addition, the transfer matrix is printed as a result of the command *mapout* in *#labor*. Note that the 1,1 and 3,3 entries for the transfer matrix are very small. (Indeed, they could be made to vanish exactly by slightly adjusting various parameter values.) That, of course, is what is required of a spot forming system.

Finally, again as a result of the command *mapout*, the nonzero elements in the Lie algebraic generating polynomial are printed. As will be described later, this polynomial characterizes the nonlinear parts of the transfer map. In particular, "spherical type" aberrations are controlled by the coefficients numbered 140, 149, and 195. Inspection shows that these coefficients are quite large in this example.

2.3 Simple Imaging System

Consider the problem of analyzing the performance of a simple imaging system, half of which is shown in figure 2.3.1 below. The system is required to provide unit magnification when acting on rays of 20 TeV protons, and is made with superconducting magnets. It consists of four quadrupole doublets, and various drifts.

Figure 2.3.1: Half Cell of Simple Quadrupole Imaging System.

The result of a MaryLie 3.0 run, structured to compute the transfer map for this system and then trace rays, is shown below:

```
#comment
 Exhibit 2.3.1.
 This is a study of a simple imaging system.
#beam
  66710.00000000000
  21315.00000000000
  1.000000000000000
  1.000000000000000
#menu
 dl1      drft
   99.9898870000000
 dl2      drft
   100.000000000000
 ds       drft
   5.00000000000000
 qf1      quad
   10.0000000000000       192.642380000000       1.00000000000000
   1.00000000000000
 qf2      quad
   10.0000000000000       192.642395000000       1.00000000000000
   1.00000000000000
 qd1      quad
   10.0000000000000      -192.642380000000       1.00000000000000
   1.00000000000000
 qd2      quad
   10.0000000000000      -192.642395000000       1.00000000000000
   1.00000000000000
 end      end
 rays     rt
   13.0000000000000       14.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000      0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
#lines
 half
```

```
      1*dl1         1*qf1         1*ds          1*qd2         2*dl2
&
      1*qf2         1*ds          1*qd1         1*dl1
  image
      2*half
 #lumps
 #loops
 #labor
     1*fileout
     1*image
     1*mapout
     1*end


 matrix for map is :

 9.99856E-01 -1.28018E-06  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00
-3.22156E-06  1.00014E+00  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  1.00014E+00 -1.28018E-06 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00 -3.22156E-06  9.99856E-01 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 1.00000E+00 1.98067E-06
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00 1.00000E+00
 nonzero elements in generating polynomial are :

  f( 33)=f( 20 00 01 )=-0.38872686692784D-01
  f( 38)=f( 11 00 01 )= 0.34566885965630D+01
  f( 53)=f( 02 00 01 )=-0.48798663436745D+03
  f( 67)=f( 00 20 01 )=-0.38872615870228D-01
  f( 70)=f( 00 11 01 )=-0.34598333034969D+01
  f( 76)=f( 00 02 01 )=-0.48812740619185D+03
  f( 83)=f( 00 00 03 )=-0.99033373115764D-06
  f( 84)=f( 40 00 00 )=-0.83288898279040D-04
  f( 85)=f( 31 00 00 )= 0.25161405938893D-01
  f( 90)=f( 22 00 00 )=-0.62836326304276D+01
  f( 95)=f( 20 20 00 )=-0.30874907661279D-03
  f( 96)=f( 20 11 00 )= 0.23903124877554D-03
  f( 99)=f( 20 02 00 )=-0.39026732544282D+01
  f(104)=f( 20 00 02 )=-0.97111094673821D-01
  f(105)=f( 13 00 00 )= 0.32933448063609D+03
  f(110)=f( 11 20 00 )=-0.31327507049347D-03
  f(111)=f( 11 11 00 )=-0.15267268376928D+02
  f(114)=f( 11 02 00 )= 0.17231939107284D+02
  f(119)=f( 11 00 02 )= 0.86317322338312D+01
  f(140)=f( 04 00 00 )=-0.13424150978119D+05
  f(145)=f( 02 20 00 )=-0.39004786583392D+01
  f(146)=f( 02 11 00 )=-0.17546489785270D+02
  f(149)=f( 02 02 00 )=-0.49593499786268D+05
  f(154)=f( 02 00 02 )=-0.39715437881394D+03
  f(175)=f( 00 40 00 )=-0.83321956145577D-04
  f(176)=f( 00 31 00 )=-0.25194640841094D-01
  f(179)=f( 00 22 00 )=-0.62868169322496D+01
  f(184)=f( 00 20 02 )=-0.97110897267768D-01
  f(185)=f( 00 13 00 )=-0.32960255997846D+03
```

```
f(190)=f( 00 11 02 )=-0.86342917628377D+01
f(195)=f( 00 04 00 )=-0.13431897330774D+05
f(200)=f( 00 02 02 )=-0.39726895527551D+03
f(209)=f( 00 00 04 )=-0.99033373279231D-06
```

The *#lines* component of the Master Input File defines the entire optical system in terms of the line called *image*. The contents of the line *image* is in turn defined in terms of the line *half*. The contents of the line *half* is the same as that shown in figure 2.3.1.

The key entries in the *#labor* component of the Master Input File are *image*, *mapout*, and *rays*. The entry *image* calls for the computation of the transfer map described by the line *image*. The commands *mapout* and *rays* then call for the printing of the map and the tracing of rays through the system

Inspection of the transfer matrix for the system shows that the transverse components are described by what is very nearly a 4 x 4 identity matrix. Consequently, in linear paraxial approximation, the system is imaging with unit magnification as advertised.

However, as indicated by the list of nonzero elements in the generating polynomial, the system has a large number of aberrations. The quantitative importance of these aberrations may be read off from the list, and their effect may be readily seen in ray traces.

Figure 2.3.2 below shows the image formed when 1377 rays are traced from an object composed of the word MARYLIE (written in two lines). The reader familiar with the subject of image formation will recognize the presence of two major aberrations. First, there is distortion. It accounts for the deformation of the letters M and Y in MARYLIE. Second, there are aberrations that depend also on ray direction. They account for the blur in the image. Detailed analysis shows that the total effect is caused mainly by the interplay of third-order distortion, astigmatism, curvature of field, and their higher order consequences due to symplectification.

It is interesting to note that object size governs the relative importance of the aberrations that depend on position as compared to those that depend on ray direction. Figure 2.3.3 shows the result of a MaryLie run when the size of the object (the word MARYLIE) is reduced, but the angular spread in the rays coming from each point in the object is kept unchanged. Now the quality of the image is dominated by "spherical type" aberrations.

It can be shown that most of the aberrations that control the performance of the simple imaging system (as well as the spot forming system) arise from quadrupole fringe field effects. Exhibit 2.3.2 shows a MaryLie 3.0 run for which the hard-edge quadrupole fringe fields have been turned off. (Observe that the last two parameters describing each quadrupole in #menu are now set equal to zero.) Inspection of the list of nonzero elements in the generating polynomial shows that the dominant entries are now much smaller. Correspondingly, figure 2.3.4 displays the image of the (full sized) word MARYLIE when the effect of quadrupole fringe fields is neglected. Now there are no apparent aberration effects.

Of course, a real quadrupole does have fringe fields. Moreover, it can be shown that, in many cases of interest, the effect of real fringe fields is essentially the same as that simulated by the hard-edged quadrupole fringe fields routinely available in MaryLie.

Figure 2.3.2: Image of the word MaryLie produced by simple imaging system including effect of hard-edge quadrupole fringe fields.

Figure 2.3.3: Image of the word MaryLie produced by the same simple imaging system when the size of the object is reduced.

Figure 2.3.4: Image of the word MaryLie produced when the effect of quadrupole fringe fields is neglected.

```
#comment
 Exhibit 2.3.2.
 This is a study of a simple imaging system with the
 effect of quadrupole fringe fields neglected.
#beam
  66710.00000000000
  21315.00000000000
  1.000000000000000
  1.000000000000000
#menu
 dl1      drft
   99.9898870000000
 dl2      drft
   100.000000000000
 ds       drft
   5.00000000000000
 qf1      quad
   10.0000000000000       192.642380000000      0.000000000000000E+00
  0.000000000000000E+00
 qf2      quad
   10.0000000000000       192.642395000000      0.000000000000000E+00
  0.000000000000000E+00
 qd1      quad
   10.0000000000000      -192.642380000000      0.000000000000000E+00
  0.000000000000000E+00
 qd2      quad
   10.0000000000000      -192.642395000000      0.000000000000000E+00
  0.000000000000000E+00
 end      end
```

```
  rays     rt
    13.0000000000000       14.0000000000000       5.00000000000000
    1.00000000000000       1.00000000000000      0.000000000000000E+00
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000

  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
 #lines
  half
      1*dl1         1*qf1         1*ds          1*qd2         2*dl2
&
      1*qf2         1*ds          1*qd1         1*dl1
  image
      2*half
 #lumps
 #loops
 #labor
     1*fileout
     1*image
     1*mapout
     1*end

 matrix for map is :
 9.99856E-01 -1.28018E-06  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00
-3.22156E-06  1.00014E+00  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  1.00014E+00 -1.28018E-06 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00 -3.22156E-06  9.99856E-01 0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 1.00000E+00 1.98067E-06
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00 1.00000E+00

 nonzero elements in generating polynomial are :

  f( 33)=f( 20 00 01 )=-0.38872686692784D-01
  f( 38)=f( 11 00 01 )= 0.34566885965630D+01
  f( 53)=f( 02 00 01 )=-0.48798663436745D+03
  f( 67)=f( 00 20 01 )=-0.38872615870228D-01
  f( 70)=f( 00 11 01 )=-0.34598333034969D+01
  f( 76)=f( 00 02 01 )=-0.48812740619185D+03
  f( 83)=f( 00 00 03 )=-0.99033373115764D-06
  f( 84)=f( 40 00 00 )=-0.36576395844088D-05
  f( 85)=f( 31 00 00 )= 0.72452798622020D-03
  f( 90)=f( 22 00 00 )=-0.25247349820513D+00
  f( 95)=f( 20 20 00 )=-0.71223902931457D-05
  f( 96)=f( 20 11 00 )=-0.55678664776493D-03
  f( 99)=f( 20 02 00 )=-0.77780590984358D-01
  f(104)=f( 20 00 02 )=-0.97111094673821D-01
  f(105)=f( 13 00 00 )= 0.18656675553303D+02
  f(110)=f( 11 20 00 )= 0.55541554577229D-03
  f(111)=f( 11 11 00 )=-0.22030609589099D+00
  f(114)=f( 11 02 00 )=-0.41123941842952D+01
  f(119)=f( 11 00 02 )= 0.86317322338312D+01
```

```
f(140)=f( 04 00 00 )=-0.78894230698405D+03
f(145)=f( 02 20 00 )=-0.77722499566606D-01
f(146)=f( 02 11 00 )= 0.41071535499672D+01
f(149)=f( 02 02 00 )=-0.62932224711005D+03
f(154)=f( 02 00 02 )=-0.39715437881394D+03
f(175)=f( 00 40 00 )=-0.36578660092510D-05
f(176)=f( 00 31 00 )=-0.72594612592996D-03
f(179)=f( 00 22 00 )=-0.25265388679573D+00
f(184)=f( 00 20 02 )=-0.97110897267768D-01
f(185)=f( 00 13 00 )=-0.18672228901964D+02
f(190)=f( 00 11 02 )=-0.86342917628377D+01
f(195)=f( 00 04 00 )=-0.78939756209382D+03
f(200)=f( 00 02 02 )=-0.39726895527551D+03
f(209)=f( 00 00 04 )=-0.99033373279231D-06
```

2.4 Simple Spectrometer

The examples of the simple spot forming system and the simple imaging system both dealt with monoenergtic rays. This example treats rays with differing energies.

Consider the simple spectrometer shown in figure 2.4.1 below. It consists of a 90° normal entry and exit dipole bending magnet preceded and followed by identical drifts. The system employs a 0.5 Telsa dipole field strength and is designed to work with 200 MeV electrons. As indicated in the figure, the drift lengths have been adjusted so that (in the linear approximation) the system is focusing (imaging) in the horizontal direction. Because the dipole is employed with normal entry and exit, the system is not focusing in the vertical direction. That is, the system is not double focusing.

Figure 2.4.1: A simple spectrometer. Electrons in the horizontal plane emanating from point A, and having an energy of 200 MeV, are brought to a focus at point B.

The result of a MaryLie 3.0 run set up to study this system is shown below:

```
#comment
 Exhibit 2.4.
```

```
 This is a study of a simple electron spectrometer.
#beam
 0.6688305422661700
  391.3868283459600
  1.000000000000000
  1.000000000000000
#menu
 dipole   nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000
  0.500000000000000       1.00000000000000       1.00000000000000
 end      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 trmat    ptm
  0.000000000000000E+00  0.000000000000000E+00   3.00000000000000
   3.00000000000000       1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 rays     rt
   13.0000000000000       14.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000      0.000000000000000E+00
 dr       drft
   1.33766000000000
#lines
 spec
     1*dr          1*dipole      1*dr
#lumps
#loops
#labor
    1*fileout
    1*spec
    1*mapout
    1*trmat
    1*rays
    1*end

 matrix for map is :

-9.99999E-01  2.16906E-06 0.00000E+00 0.00000E+00  0.00000E+00 -2.67533E+00
-7.47574E-01 -9.99999E-01 0.00000E+00 0.00000E+00  0.00000E+00 -1.00000E+00
 0.00000E+00  0.00000E+00 1.00000E+00 4.77651E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 0.00000E+00 1.00000E+00  0.00000E+00  0.00000E+00
 1.00000E+00  2.67533E+00 0.00000E+00 0.00000E+00  1.00000E+00 -7.63506E-01
 0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00  0.00000E+00  1.00000E+00

 nonzero elements in generating polynomial are :

  f( 28)=f( 30 00 00 )=-0.93144360270528D-01
  f( 29)=f( 21 00 00 )= 0.37378645487843D+00
  f( 33)=f( 20 00 01 )=-0.74757564052220D+00
  f( 34)=f( 12 00 00 )=-0.49999918923267D+00
  f( 38)=f( 11 00 01 )= 0.20000040626018D+01
```

```
f( 39)=f( 10 20 00 )=-0.27943308081159D+00
f( 40)=f( 10 11 00 )= 0.19218586424872D+01
f( 43)=f( 10 02 00 )=-0.38044947926129D+01
f( 48)=f( 10 00 02 )=-0.15000121790723D+01
f( 49)=f( 03 00 00 )= 0.44588648591172D+00
f( 53)=f( 02 00 01 )=-0.20064948891860D+01
f( 54)=f( 01 20 00 )= 0.74757321281106D+00
f( 55)=f( 01 11 00 )=-0.35707926209417D+01
f( 58)=f( 01 02 00 )= 0.64267805042870D+01
f( 63)=f( 01 00 02 )= 0.20065106354604D+01
f( 67)=f( 00 20 01 )=-0.37378797178870D+00
f( 70)=f( 00 11 01 )= 0.25708038645726D+01
f( 76)=f( 00 02 01 )=-0.64268055010847D+01
f( 83)=f( 00 00 03 )=-0.89179523414583D+00
f( 84)=f( 40 00 00 )=-0.52224150326159D-01
f( 85)=f( 31 00 00 )= 0.27943262770116D+00
f( 89)=f( 30 00 01 )=-0.37257842445493D+00
f( 90)=f( 22 00 00 )=-0.65412546263926D+00
f( 94)=f( 21 00 01 )= 0.16820436001359D+01
f( 95)=f( 20 20 00 )=-0.52224192667846D-01
f( 96)=f( 20 11 00 )= 0.41914950793973D+00
f( 99)=f( 20 02 00 )=-0.13347157761221D+01
f(104)=f( 20 00 02 )=-0.13082632129161D+01
f(105)=f( 13 00 00 )= 0.74999817577475D+00
f(109)=f( 12 00 01 )=-0.30000042696722D+01
f(110)=f( 11 20 00 )=-0.29921602979568D+00
f(111)=f( 11 11 00 )= 0.97520757686635D+00
f(114)=f( 11 02 00 )= 0.23370935338336D+01
f(119)=f( 11 00 02 )= 0.45000280175421D+01
f(123)=f( 10 20 01 )=-0.55886786323843D+00
f(126)=f( 10 11 01 )= 0.34170158616280D+01
f(132)=f( 10 02 01 )=-0.78753142617592D+01
f(139)=f( 10 00 03 )=-0.20000280110477D+01
f(140)=f( 04 00 00 )=-0.50162182216850D+00
f(144)=f( 03 00 01 )= 0.21179659236963D+01
f(145)=f( 02 20 00 )= 0.40024916288944D+00
f(146)=f( 02 11 00 )=-0.19117958694217D+01
f(149)=f( 02 02 00 )=-0.16758500076646D+01
f(154)=f( 02 00 02 )=-0.48490507651341D+01
f(158)=f( 01 20 01 )= 0.72111221900917D+00
f(161)=f( 01 11 01 )=-0.40517040534188D+01
f(167)=f( 01 02 01 )= 0.12766450201372D+02
f(174)=f( 01 00 03 )= 0.36786182170241D+01
f(175)=f( 00 40 00 )=-0.52224192667846D-01
f(176)=f( 00 31 00 )= 0.49889908271819D+00
f(179)=f( 00 22 00 )=-0.18588566566436D+01
f(184)=f( 00 20 02 )=-0.65413213721516D+00
f(185)=f( 00 13 00 )= 0.31876425698809D+01
f(190)=f( 00 11 02 )= 0.38208251163414D+01
f(195)=f( 00 04 00 )=-0.28789752521441D+01
f(200)=f( 00 02 02 )=-0.94839470235307D+01
f(209)=f( 00 00 04 )=-0.10590132322744D+01
```

```
nonzero elements in second order matrix are :

 t1(  7)=t1( 20 00 00 )=-0.37378675793264D+00
 t1(  8)=t1( 11 00 00 )=-0.99999918923235D+00
 t1( 12)=t1( 10 00 01 )=-0.20000056841418D+01
 t1( 13)=t1( 02 00 00 )=-0.13376594577343D+01
 t1( 17)=t1( 01 00 01 )=-0.40129908629061D+01
 t1( 18)=t1( 00 20 00 )=-0.74757321281106D+00
 t1( 19)=t1( 00 11 00 )=-0.35707938944926D+01
 t1( 22)=t1( 00 02 00 )=-0.64267835458532D+01
 t1( 27)=t1( 00 00 02 )=-0.20065122622686D+01
 t2(  7)=t2( 20 00 00 )=-0.13877787807814D-16
 t2(  8)=t2( 11 00 00 )=-0.13877787807814D-16
 t2( 12)=t2( 10 00 01 )=-0.27755575615629D-16
 t2( 13)=t2( 02 00 00 )=-0.50000000000000D+00
 t2( 17)=t2( 01 00 01 )=-0.10000024366857D+01
 t2( 18)=t2( 00 20 00 )=-0.27943308081159D+00
 t2( 19)=t2( 00 11 00 )=-0.74757290975685D+00
 t2( 22)=t2( 00 02 00 )=-0.99999918923267D+00
 t2( 27)=t2( 00 00 02 )=-0.32474612975120D-05
 t3(  9)=t3( 10 10 00 )=-0.74757290975685D+00
 t3( 10)=t3( 10 01 00 )=-0.15707934317094D+01
 t3( 14)=t3( 01 10 00 )=-0.35707938944926D+01
 t3( 15)=t3( 01 01 00 )=-0.42023767872970D+01
 t3( 21)=t3( 00 10 01 )=-0.10000024366857D+01
 t3( 24)=t3( 00 01 01 )= 0.57413261902772D+00
 t4(  9)=t4( 10 10 00 )=-0.55886616162317D+00
 t4( 10)=t4( 10 01 00 )=-0.19218586424872D+01
 t4( 14)=t4( 01 10 00 )=-0.14951464256221D+01
 t4( 15)=t4( 01 01 00 )=-0.35707926209417D+01
 t4( 21)=t4( 00 10 01 )=-0.74757594357739D+00
 t4( 24)=t4( 00 01 01 )=-0.25708038645726D+01
 t5(  7)=t5( 20 00 00 )= 0.37378766873350D+00
 t5(  8)=t5( 11 00 00 )= 0.10000016259161D+01
 t5( 12)=t5( 10 00 01 )= 0.10000121790723D+01
 t5( 13)=t5( 02 00 00 )= 0.20064959737214D+01
 t5( 17)=t5( 01 00 01 )= 0.26753536674612D+01
 t5( 18)=t5( 00 20 00 )= 0.37378797178870D+00
 t5( 19)=t5( 00 11 00 )= 0.10000024366857D+01
 t5( 22)=t5( 00 02 00 )= 0.26753286879845D+01
 t5( 27)=t5( 00 00 02 )= 0.66887561213844D+00

nonzero elements in third order matrix are :

 u1( 28)=u1( 30 00 00 )=-0.27943285425628D+00
 u1( 29)=u1( 21 00 00 )=-0.11213584554734D+01
 u1( 33)=u1( 20 00 01 )=-0.14951509779892D+01
 u1( 34)=u1( 12 00 00 )=-0.19999963515485D+01
 u1( 38)=u1( 11 00 01 )=-0.50000097511198D+01
 u1( 39)=u1( 10 20 00 )=-0.27943308081159D+00
 u1( 40)=u1( 10 11 00 )=-0.14951464256221D+01
 u1( 43)=u1( 10 02 00 )=-0.40707942998761D+01
 u1( 48)=u1( 10 00 02 )=-0.35000272002167D+01
```

```
u1( 49)=u1( 03 00 00 )=-0.20064878309384D+01
u1( 53)=u1( 02 00 01 )=-0.60194852098259D+01
u1( 54)=u1( 01 20 00 )= 0.80049927785190D+00
u1( 55)=u1( 01 11 00 )= 0.46089928282951D+01
u1( 58)=u1( 01 02 00 )=-0.14759688968051D+01
u1( 63)=u1( 01 00 02 )=-0.86948546235550D+01
u1( 67)=u1( 00 20 01 )=-0.14951512810444D+01
u1( 70)=u1( 00 11 01 )=-0.85708168631484D+01
u1( 76)=u1( 00 02 01 )=-0.17986352150007D+02
u1( 83)=u1( 00 00 03 )=-0.26753683263812D+01
u2( 28)=u2( 30 00 00 )=-0.20816681711722D-16
u2( 33)=u2( 20 00 01 )=-0.14571677198205D-15
u2( 34)=u2( 12 00 00 )=-0.20816681711722D-16
u2( 38)=u2( 11 00 01 )= 0.11102230246252D-15
u2( 39)=u2( 10 20 00 )=-0.69388939039072D-17
u2( 40)=u2( 10 11 00 )=-0.27755575615629D-16
u2( 43)=u2( 10 02 00 )=-0.55511151231258D-15
u2( 48)=u2( 10 00 02 )=-0.61062266354384D-15
u2( 49)=u2( 03 00 00 )=-0.49999959461617D+00
u2( 53)=u2( 02 00 01 )=-0.50000162372801D+00
u2( 55)=u2( 01 11 00 )= 0.74757351586528D+00
u2( 58)=u2( 01 02 00 )= 0.49999959461617D+00
u2( 63)=u2( 01 00 02 )=-0.10000089316083D+01
u2( 67)=u2( 00 20 01 )=-0.27943398825823D+00
u2( 70)=u2( 00 11 01 )=-0.14951506749340D+01
u2( 76)=u2( 00 02 01 )=-0.20000040626022D+01
u2( 83)=u2( 00 00 03 )=-0.32474718438813D-05
u3( 30)=u3( 20 10 00 )= 0.16653345369377D-15
u3( 31)=u3( 20 01 00 )=-0.37378645487842D+00
u3( 35)=u3( 11 10 00 )= 0.42671126479216D+00
u3( 36)=u3( 11 01 00 )=-0.14292029772358D+01
u3( 42)=u3( 10 10 01 )=-0.14951506749340D+01
u3( 45)=u3( 10 01 01 )=-0.61416043756418D+01
u3( 50)=u3( 02 10 00 )=-0.92920500415430D+00
u3( 51)=u3( 02 01 00 )=-0.29623833617786D+01
u3( 57)=u3( 01 10 01 )=-0.60000142721308D+01
u3( 60)=u3( 01 01 01 )=-0.17768425853934D+02
u3( 64)=u3( 00 30 00 )=-0.27943285425628D+00
u3( 65)=u3( 00 21 00 )=-0.13347140878695D+01
u3( 68)=u3( 00 12 00 )=-0.25707905366266D+01
u3( 73)=u3( 00 10 02 )=-0.20000146157580D+01
u3( 74)=u3( 00 03 00 )=-0.76352754186107D+00
u3( 79)=u3( 00 01 02 )=-0.48712382645289D+01
u4( 30)=u4( 20 10 00 )=-0.12143064331838D-16
u4( 31)=u4( 20 01 00 )=-0.55886616162317D+00
u4( 35)=u4( 11 10 00 )= 0.87786491384764D+00
u4( 36)=u4( 11 01 00 )=-0.10684345547215D+01
u4( 42)=u4( 10 10 01 )=-0.55886797651645D+00
u4( 45)=u4( 10 01 01 )=-0.41645921082604D+01
u4( 50)=u4( 02 10 00 )= 0.80049802272480D+00
u4( 51)=u4( 02 01 00 )=-0.17146015460171D+01
u4( 57)=u4( 01 10 01 )=-0.10684370723398D+01
u4( 60)=u4( 01 01 01 )=-0.95708180262792D+01
```

```
u4( 64)=u4( 00 30 00 )=-0.20889677067138D+00
u4( 65)=u4( 00 21 00 )=-0.12772310196927D+01
u4( 68)=u4( 00 12 00 )=-0.26694299940634D+01
u4( 73)=u4( 00 10 02 )=-0.74758079901346D+00
u4( 74)=u4( 00 03 00 )=-0.25707934317102D+01
u4( 79)=u4( 00 01 02 )=-0.45708327407763D+01
u5( 28)=u5( 30 00 00 )= 0.93144662752742D-01
u5( 29)=u5( 21 00 00 )= 0.37378766873350D+00
u5( 33)=u5( 20 00 01 )= 0.11213693785478D+01
u5( 34)=u5( 12 00 00 )= 0.10000020313009D+01
u5( 38)=u5( 11 00 01 )= 0.40000267991940D+01
u5( 39)=u5( 10 20 00 )= 0.55886774996041D+00
u5( 40)=u5( 10 11 00 )= 0.19218642775287D+01
u5( 43)=u5( 10 02 00 )= 0.22337069932108D+01
u5( 48)=u5( 10 00 02 )= 0.25000422192713D+01
u5( 49)=u5( 03 00 00 )= 0.17835517356330D+01
u5( 53)=u5( 02 00 01 )= 0.60195177898336D+01
u5( 54)=u5( 01 20 00 )= 0.11213630062007D+01
u5( 55)=u5( 01 11 00 )= 0.15707977220230D+01
u5( 58)=u5( 01 02 00 )= 0.23191094090778D+01
u5( 63)=u5( 01 00 02 )= 0.60195775186149D+01
u5( 67)=u5( 00 20 01 )= 0.11213696816060D+01
u5( 70)=u5( 00 11 01 )= 0.45708292655895D+01
u5( 76)=u5( 00 02 01 )= 0.83386436657300D+01
u5( 83)=u5( 00 00 03 )= 0.15606846026334D+01
```

The *#menu* component of the Master Input File contains, among other entries, the element *dipole*. The first two parameters under this element specify the dipole bend angle and strength, respectively. The remaining two parameters (both set equal to +1 in this example) indicate that the dipole is to be treated as having hard-edge leading and trailing fringe fields. The *#lines* component of the Master Input File defines the simple spectrometer in terms of the line called *spec*. The key entries in the *#labor* component are *spec*, *mapout*, *tmat*, and *rays*. The entry *spec* calls for the computation of the transfer map described by the line *spec*. The command *mapout* calls for the printing of the transfer map for the system. The command *tmat* calls for the printing of the second- and third-order transfer matrices for the system. Finally, *rays* produces a ray trace.

There are several observations to be made about the MaryLie output. First, note that the 1,2 component of the transfer matrix vanishes, and the 3,4 component does not. Thus the system is indeed imaging in the horizontal plane, but not in the vertical plane. Second, note that the 1,6 component of the transfer matrix is nonzero. Consequently, the system is dispersive as desired for a spectrometer. Finally, observe that there are a large number of entries in the generating polynomial describing the nonlinear behavior of the system. As a result, the system has a large number of second and third order aberrations. Many of these aberrations arise primarily from the dipole fringe fields.

It will be shown in a later section that aberrations are completely and most compactly described in terms of the Lie algebraic generating polynomial for the nonlinear portion of the transfer map. Indeed, second order aberrations are described by the monomials numbered 28 through 83, and third order aberrations are described by the monomials numbered 84 through 209. However, as also will be shown later, it is possible as well to characterize a

transfer map in terms of a Taylor series expansion. When this is done through third order, there is a set of coefficients T_{ijk} (the so-called T matrix or second-order transfer matrix) for the quadratic terms, and a set of coefficients U_{ijkl} for the cubic terms.

If desired, MaryLie can be used to produce these Taylor series coefficients. In this particular MaryLie run, the second- and third-order transfer matrix for the simple spectrometer have been produced as a result of the command *tmat*. Note that these matrices fill several pages. By contrast, the Lie algebraic representation of all the same information requires only a single page. Consequently, preliminary (and perhaps most) analyses of the importance of aberrations are most easily made using the Lie algebraic representation.

How well does this simple spectrometer work? To study this question, two sets of rays may be launched from the point A of figure 2.4.1. The first set is selected to have $P_\tau = 5 \times 10^{-5}$ and 25 different combinations of P_x and P_y values. The second set is selected to have $P_\tau = -5 \times 10^{-5}$, and the same combination of P_x and P_y values. (As indicated earlier, and will be described later, the quantity P_τ is related to energy deviation.) The phase-space coordinates for all these rays are stored in an initial condition file. The first few lines of this file are shown below:

First few lines of initial condition file for spectrometer ray traces

X	P_x	Y	P_y	τ	P_τ
0.0000E+00	-0.4000E-02	0.0000E+00	-0.4000E-02	0.0000E+00	0.5000E-04
0.0000E+00	-0.4000E-02	0.0000E+00	-0.2000E-02	0.0000E+00	0.5000E-04
0.0000E+00	-0.4000E-02	0.0000E+00	0.0000E+00	0.0000E+00	0.5000E-04
0.0000E+00	-0.4000E-02	0.0000E+00	0.2000E-02	0.0000E+00	0.5000E-04
0.0000E+00	-0.4000E-02	0.0000E+00	0.4000E-02	0.0000E+00	0.5000E-04

Note that the X, Y, and τ coordinates of all rays are taken to be zero. This is because the point A has been selected to lie on the "design orbit" or "chief ray" for the spectrometer. Also, the P_x and P_y values have been selected from a square array. Figure 2.4.2 below shows the result of plotting the second and fourth (the P_x and P_y phase-space coordinates) of the initial condition file.

Now suppose MaryLie is used to trace these two sets of rays through the spectrometer to the final point B. Figure 2.4.3 shows the result obtained in the X, Y plane, and figure 2.4.4 shows the same final rays in terms of the variables P_τ and X.

Inspection of figures 2.4.3 and 2.4.4 shows that the rays arrive at approximately two horizontal (X) locations depending on their P_τ values. Thus, the system is horizontally focusing as expected. Moreover, there is a wide spread of Y values since the system is not vertically focusing. Finally, there is some spread in the two horizontal arrival values depending on the P_x and P_y values. It can be shown that this spread arises primarily from second-order aberrations described by the coefficients numbered 49 and 58 in the Lie algebraic generating polynomial for the total transfer map.

The observant reader may wonder why Fig. 2.4.3 appears to have only 30 points as the result of tracing 50 rays. The explanation is that the arrival point of a ray is nearly the same for corresponding initial P_x values.

Figure 2.4.2: Plot of Initial Conditions for Incoming Rays of the Simple Spectrometer.

Figure 2.4.3: Plot of the pattern produced at the focus B of the spectrometer as the result of two sets of incoming rays.

Figure 2.4.4: Horizontal arrival point x of final rays as a function of energy deviation.

As can be seen from either of figures 2.4.3 and 2.4.4, the two sets of rays with different values of P_τ are well separated in X. Therefore, the spectrometer has been able to resolve the two different values of P_τ. Now suppose the angular spread of the incoming rays is increased by a factor of 1.5 so that P_x, P_y now reach the extreme values .006, and suppose the two P_τ values are the same as before. Figures 2.4.5 and 2.4.6 show the result of ray tracing for this case. Evidently, the X values of the two sets of rays now overlap due to aberration effects. Correspondingly, the two P_τ values are no longer resolved.

Figure 2.4.5: Plot of the final ray pattern produced at B when the angular spread of the incoming rays is increased.

Figure 2.4.6: Horizontal arrival point x of final rays as a function of energy deviation in the case where the angular spread of the incoming rays is increased.

2.5 Small Static Storage Ring

The examples of the simple spot forming system, the simple imaging system, and the simple spectrometer all dealt with single pass systems. They were therefore not concerned with the long-term behavior of trajectories. By contrast, this example treats a circulating storage system for which long-term behavior is of paramount importance.

Consider the small Proton Storage Ring (called the PSR and similar to that at the Los Alamos National Laboratory) shown in figure 2.5.1 below. It is designed to store protons with a nominal energy of 800 MeV using H^- injection, and is arranged as a ten-sided separated function lattice consisting of straight sections and bends. The lattice is composed entirely of dipoles, quadrupoles, and drifts with the exception of two pairs of sextupoles. When these sextupoles are turned off, the lattice has ten identical periods. When the sextupoles are turned on, the symmetry of the lattice is reduced. Inspection of figure 2.5.1 shows that in this latter case the lattice can be viewed as consisting of two identical halves, and thus it has two identical periods.

Figure 2.5.1: Layout of Proton Storage Ring.

Further inspection of the lattice shows that it can be viewed as being composed of three kinds of periods. These periods and their contents are shown in figure 2.5.2.

Figure 2.5.2: Details and dimensions of constituent periods of Proton Storage Ring.

To analyze the performance of the proton storage ring, it is useful to make two sets of MaryLie runs. The first set of runs will be made to study the one-turn transfer map for the ring. The second set will examine long-term orbit behavior.

The results of a preliminary run to study the one-turn transfer map are shown below:

```
#comment
 Exhibit 2.5.1
 This is a study of a small static storage ring.
 This MARYLIE run will analyze the one lattice period
 and one turn transfer maps.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
```

```
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hdq      quad
  0.500000000000000      -1.92000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hcs      sext
  0.500000000000000      0.000000000000000E+00
 vcs      sext
  0.500000000000000      0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
   13.0000000000000       14.0000000000000       5.00000000000000
   500.000000000000       1.00000000000000      0.000000000000000E+00
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 iden     iden
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
#lumps
#loops
#labor
```

```
    1*fileout
    1*nsex
    1*chrom
    1*iden
    1*ring
    1*chrom
    1*mapout
    1*fin

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.225410281174760
first order horizontal chromaticity = -9.281703934545682E-002
second order horizontal chromaticity =  0.104559371148683
horizontal tune when delta =  1.000000000000000E-003
 0.225317568694785

vertical tune =  0.225543770585176
first order vertical chromaticity = -0.211132387676014
second order vertical chromaticity =  0.252919932812798
vertical tune when delta =  1.000000000000000E-003
 0.225332891117433

tune separation when delta=  1.000000000000000E-003
-1.532242264759565E-005

normalized anharmonicities
 hhn=  7.695357766309113E-003
 vvn=  1.802675106441745E-002
 hvn=  2.675634626130963E-002

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.254102811747596
first order horizontal chromaticity = -0.928170393454570
second order horizontal chromaticity =   1.04559371148684
horizontal tune when delta =  1.000000000000000E-003
 0.253175686947853

vertical tune =  0.255437705851761
first order vertical chromaticity =  -2.11132387676014
second order vertical chromaticity =   2.52919932812798
vertical tune when delta =  1.000000000000000E-003
 0.253328911174329

tune separation when delta=  1.000000000000000E-003
-1.532242264760675E-004

normalized anharmonicities
```

```
 hhn=  7.695357766309278E-002
 vvn=  0.180267510644177
 hvn=  0.267563462613096

matrix for map is :

 8.79899E-01  6.43546E+00  0.00000E+00  0.00000E+00  0.00000E+00 -3.09884E+00
-2.82743E-01 -9.31451E-01  0.00000E+00  0.00000E+00  0.00000E+00 -3.07914E-01
 0.00000E+00  0.00000E+00 -9.04734E-01  6.22828E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -2.82058E-01  8.36415E-01  0.00000E+00  0.00000E+00
 1.14711E+00  4.86799E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.30681E+01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=-1.14530840551473E-02
 f( 29)=f( 21 00 00 )= 2.67517165681652E-02
 f( 33)=f( 20 00 01 )= -1.1736777355489
 f( 34)=f( 12 00 00 )= 8.77407347896805E-04
 f( 38)=f( 11 00 01 )= -6.7730221731743
 f( 39)=f( 10 20 00 )=-6.01680421031948E-02
 f( 40)=f( 10 11 00 )= 0.67669439436590
 f( 43)=f( 10 02 00 )= -1.8306262274435
 f( 48)=f( 10 00 02 )= -6.0154921926825
 f( 49)=f( 03 00 00 )= 0.16593635071889
 f( 53)=f( 02 00 01 )= -23.642071165236
 f( 54)=f( 01 20 00 )=-3.44440710710079E-02
 f( 55)=f( 01 11 00 )= 3.85161039824409E-02
 f( 58)=f( 01 02 00 )=  1.4627612575911
 f( 63)=f( 01 00 02 )= -8.7202857803725
 f( 67)=f( 00 20 01 )= -2.4822497040586
 f( 70)=f( 00 11 01 )=  16.699387190479
 f( 76)=f( 00 02 01 )= -57.101990751187
 f( 83)=f( 00 00 03 )= -34.310069304168
 f( 84)=f( 40 00 00 )=-1.37013931670580E-02
 f( 85)=f( 31 00 00 )=-0.16784866108730
 f( 89)=f( 30 00 01 )=-0.31703479901659
 f( 90)=f( 22 00 00 )=-0.92855072951694
 f( 94)=f( 21 00 01 )= -2.9203989725275
 f( 95)=f( 20 20 00 )=-0.12024362379291
 f( 96)=f( 20 11 00 )= 0.62813570271803
 f( 99)=f( 20 02 00 )= -1.3387286297467
 f(104)=f( 20 00 02 )= -3.4667565555761
 f(105)=f( 13 00 00 )= -2.6538649658506
 f(109)=f( 12 00 01 )= -8.5904681059051
 f(110)=f( 11 20 00 )=-0.57379494301836
 f(111)=f( 11 11 00 )=  2.4415653749948
 f(114)=f( 11 02 00 )= -6.2836394557255
 f(119)=f( 11 00 02 )= -22.589602187849
 f(123)=f( 10 20 01 )= -1.5269177137599
 f(126)=f( 10 11 01 )=  8.6399711664208
 f(132)=f( 10 02 01 )= -9.5208000216435
 f(139)=f( 10 00 03 )= -13.291152725777
```

```
f(140)=f( 04 00 00 )= -4.9592065206488
f(144)=f( 03 00 01 )=  1.9246183284199
f(145)=f( 02 20 00 )= -1.2337505894646
f(146)=f( 02 11 00 )=  7.7049990878473
f(149)=f( 02 02 00 )= -34.462207995042
f(154)=f( 02 00 02 )= -63.008955902638
f(158)=f( 01 20 01 )= -5.3711963846800
f(161)=f( 01 11 01 )=  23.661202919812
f(167)=f( 01 02 01 )=  36.336091551217
f(174)=f( 01 00 03 )= -27.562892961455
f(175)=f( 00 40 00 )=-2.57477900501080E-02
f(176)=f( 00 31 00 )= 0.32207137603156
f(179)=f( 00 22 00 )= -1.9603982221369
f(184)=f( 00 20 02 )= -6.2193628931016
f(185)=f( 00 13 00 )=  6.2167715680490
f(190)=f( 00 11 02 )=  40.018453416659
f(195)=f( 00 04 00 )= -11.878247622830
f(200)=f( 00 02 02 )= -117.09988758729
f(209)=f( 00 00 04 )= -51.586059604493
```

The reader is invited to compare the contents of the lines *nsex*, *tsex*, and *lsex* with the contents of the three periods shown in figure 2.5.2. He or she should also compare the contents of the lines *half* and *ring* with the contents of the lattice as shown in figure 2.5.1. Next, the reader should note that although 0.5 meter sextupoles occur in *#menu* under the user names *hcs* and *vcs*, the second parameter for each sextupole (which specifies the sextupole strength) has been set equal to zero. Finally, the reader should observe that the fringe fields for all the quadrupoles have been "turned off" since the last two parameters associated with each quadrupole are both zero.

Some entries in the *#labor* component of the Master Input File also call for comment. The entry *nsex* calls for the computation of the transfer map described by the line *nsex* (corresponding to a lattice period with no sextupoles). The command *chrom* then calls for the computation of the tunes, chromaticities, and anharmonicities (tune dependence on betatron amplitude). Next, the transfer map is reset to the identity map by means of the command *iden*.

Subsequently, the entry *ring* and the (repeated) command *chrom* call respectively for the computation of the transfer map for the full ring and the computation of its tunes, chromaticities, etc. Finally, *mapout* calls for the printing out of the transfer map for the full ring, i.e., the one-turn transfer map.

To see what results from these entries and commands, look at the two blocks of output that follow the *#labor* component. The reader should find a listing of the tune, first-order chromaticity, and second-order chromaticity for both the horizontal and vertical degrees of freedom. The first block of output gives this information for the transfer map corresponding to a single lattice period, and the second block gives this information for a full turn.

Evidently, the horizontal and vertical tunes for a single lattice period are $.2254\ldots$ and $.2255\ldots$, respectively. Correspondingly, the first-order horizontal and vertical chromaticities are $-.092\ldots$ and $-.211\ldots$, respectively.

Since the full ring is composed of ten periods, one would expect the tunes and chromaticities for the full ring (given in the second block of output) to be 10 times as large as those for a single lattice period. This is indeed the case. However, because MaryLie computes tunes directly from the total transfer map, it gives only the fractional (and not the integer) parts of tunes. Thus, the horizontal and vertical tunes for one full turn are 2.254 . . . and 2.255 . . ., and MaryLie gives the values .254 . . . and .255 . . ., respectively. See sections 8.2 and 10.10.

Finally, it should be observed that since the sextupoles are off for this particular MaryLie run, the chromaticities computed are the natural chromaticities of the ring. Correspondingly, the ring is composed entirely of what are normally regarded as linear elements. However, inspection of the generating polynomial for the nonlinear part of the one-turn transfer map shows that there are, in fact, a large number of nonzero elements. Thus, as should have already been obvious from the previous examples of single pass systems, even the so-called linear elements have nonlinear effects.

Suppose it is desired to run the Proton Storage Ring in a mode for which the design orbit has no first-order chromaticities. This can be accomplished by giving the sextupoles suitable nonzero strengths. Shown below is a MaryLie run for this case:

```
#comment
 Exhibit 2.5.2.
 This is a study of a small static storage ring.
 This MARYLIE run will analyze the one turn transfer map when
 chromaticity correcting sextupoles are turned on.
 The effect of hard-edge quadrupole fringe fields is also included.
#beam
  4.869148131759700
  0.8494258478922000
  1.000000000000000
  1.000000000000000
#menu
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       2.72000000000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.500000000000000      -1.92000000000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.500000000000000       1.62000000000000
 vcs      sext
```

```
  0.500000000000000      -3.34000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
   13.0000000000000       14.0000000000000       5.00000000000000
   500.000000000000       1.00000000000000      0.000000000000000E+00
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 iden     iden
 fin      end
#lines
 nsex
     1*drl       1*hdq         1*drs         1*bend        1*drs   &
     1*hfq       1*drl
 tsex
     1*drl       1*hdq         1*drs         1*bend        1*drs   &
     1*hfq       1*drvs        1*hcs         1*drml
 lsex
     1*drml      1*vcs         1*drvs        1*hdq         1*drs   &
     1*bend      1*drs         1*hfq         1*drl
 half
     1*nsex      1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
#lumps
#loops
#labor
    1*fileout
    1*ring
    1*chrom
    1*mapout
    1*fin

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation
horizontal tune =  0.2541028117475962
first order horizontal chromaticity = -2.9470801637403564E-04
second order horizontal chromaticity =   48.16025275066844
horizontal tune when delta =  9.9999999999999999E-04
 0.2541506772923305

vertical tune =  0.2554377058517620
first order vertical chromaticity =  2.5713854609451646E-03
second order vertical chromaticity =   5.969652580601808
vertical tune when delta =  9.9999999999999999E-04
```

```
 0.2554462468898036

tune separation when delta=  9.9999999999999999E-04
-1.2955695974730605E-03

normalized anharmonicities
 hhn=   4.463887843064638
 vvn=  -1.491178186878906
 hvn= -0.1610999831430029

matrix for map is :

 8.79899E-01  6.43546E+00  0.00000E+00 0.00000E+00 0.00000E+00 -3.09884E+00
-2.82743E-01 -9.31451E-01  0.00000E+00 0.00000E+00 0.00000E+00 -3.07914E-01
 0.00000E+00  0.00000E+00 -9.04734E-01 6.22828E+00 0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -2.82058E-01 8.36415E-01 0.00000E+00  0.00000E+00
 1.14711E+00  4.86799E+00  0.00000E+00 0.00000E+00 1.00000E+00  1.30681E+01
 0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=-0.24592784728790D+00
 f( 29)=f( 21 00 00 )=-0.30604852898231D+01
 f( 33)=f( 20 00 01 )=-0.64563088742748D+00
 f( 34)=f( 12 00 00 )=-0.15583145022099D+02
 f( 38)=f( 11 00 01 )= 0.18502158940524D+01
 f( 39)=f( 10 20 00 )=-0.12483369572898D+00
 f( 40)=f( 10 11 00 )= 0.27148128836245D+01
 f( 43)=f( 10 02 00 )=-0.87122374564362D+01
 f( 48)=f( 10 00 02 )=-0.84099767967792D+01
 f( 49)=f( 03 00 00 )=-0.25619634656306D+02
 f( 53)=f( 02 00 01 )=-0.98753574315669D+01
 f( 54)=f( 01 20 00 )=-0.41957680678127D+01
 f( 55)=f( 01 11 00 )= 0.41001609885339D+02
 f( 58)=f( 01 02 00 )=-0.10461799484601D+03
 f( 63)=f( 01 00 02 )= 0.12493092450878D+01
 f( 67)=f( 00 20 01 )= 0.24840176792545D+01
 f( 70)=f( 00 11 01 )=-0.24548154751239D+02
 f( 76)=f( 00 02 01 )= 0.37914232657814D+02
 f( 83)=f( 00 00 03 )=-0.43860487074368D+02
 f( 84)=f( 40 00 00 )=-0.53238969994112D+00
 f( 85)=f( 31 00 00 )=-0.52691073709927D+01
 f( 89)=f( 30 00 01 )=-0.76197059527104D+01
 f( 90)=f( 22 00 00 )=-0.33384606194653D+02
 f( 94)=f( 21 00 01 )=-0.45466953399356D+02
 f( 95)=f( 20 20 00 )= 0.30176483526703D+00
 f( 96)=f( 20 11 00 )=-0.73344825077518D+01
 f( 99)=f( 20 02 00 )= 0.48285862483518D+01
 f(104)=f( 20 00 02 )=-0.70911206118901D+02
 f(105)=f( 13 00 00 )=-0.13016712967035D+03
 f(109)=f( 12 00 01 )=-0.77668739786113D+02
 f(110)=f( 11 20 00 )= 0.10964795411111D+02
 f(111)=f( 11 11 00 )=-0.96814661908489D+02
```

```
 f(114)=f( 11 02 00 )= 0.67959248309980D+02
 f(119)=f( 11 00 02 )=-0.41584221544659D+03
 f(123)=f( 10 20 01 )=-0.71637764336681D+01
 f(126)=f( 10 11 01 )= 0.47588769596512D+02
 f(132)=f( 10 02 01 )=-0.19324624884052D+03
 f(139)=f( 10 00 03 )=-0.21947536145204D+03
 f(140)=f( 04 00 00 )=-0.33768147778230D+03
 f(144)=f( 03 00 01 )= 0.22187955157124D+03
 f(145)=f( 02 20 00 )= 0.26092739583358D+02
 f(146)=f( 02 11 00 )=-0.29953156756094D+03
 f(149)=f( 02 02 00 )= 0.33203293824300D+03
 f(154)=f( 02 00 02 )=-0.12804942840066D+04
 f(158)=f( 01 20 01 )=-0.33119338725585D+02
 f(161)=f( 01 11 01 )= 0.38145383287607D+03
 f(167)=f( 01 02 01 )=-0.13066261557003D+04
 f(174)=f( 01 00 03 )=-0.22257971295707D+03
 f(175)=f( 00 40 00 )= 0.21231457461353D+00
 f(176)=f( 00 31 00 )=-0.60270556962489D+01
 f(179)=f( 00 22 00 )= 0.43923145828609D+02
 f(184)=f( 00 20 02 )=-0.72981558046921D+01
 f(185)=f( 00 13 00 )=-0.12013612969892D+03
 f(190)=f( 00 11 02 )= 0.50696669600891D+02
 f(195)=f( 00 04 00 )= 0.43239694128681D+02
 f(200)=f( 00 02 02 )=-0.38530783430069D+03
 f(209)=f( 00 00 04 )=-0.44005157367639D+03
```

Note that the first-order chromaticities are now inconsequential. Also, the second-order chromaticities have changed. This is partly due to turning on the sextupoles, and partly due to the inclusion of hard-edge quadrupole fringe field effects. (Observe that the quadrupoles appearing in *#menu* now have their fringe fields turned on.) Finally, the reader should compare the nonlinear content of the transfer map (the generating polynomial) for the two cases for which the sextupoles are turned off and turned on. As might be expected, the nonlinear content is much higher for the case in which the sextupoles are turned on.

It is well known that nonlinearities in the one-turn transfer map can produce nonlinear structure resonances. Below is a MaryLie run illustrating typical 1/3 integer resonance behavior. The quadrupole strengths have been adjusted to give a horizontal tune of approximately 1.33 for half a turn, and the sextupoles have been readjusted to again achieve negligible first-order chromaticities.

```
#comment
 Exhibit 2.5.3.
 This is a study of a small static storage ring.
 This MARYLIE run will analyze the half turn transfer map, and then
 track multiple half turns through the lattice.
#beam
  4.869148131759700
 0.8494258478922000
  1.000000000000000
  1.000000000000000
#menu
 drvs     drft
```

```
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       3.13000000000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.500000000000000      -1.92000000000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.500000000000000       2.65000000000000
 vcs      sext
  0.500000000000000      -5.01000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      0.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
   310.000000000000       1.00000000000000      0.000000000000000E+00
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 iden     iden
 fin      end
#lines
 nsex
     1*drl      1*hdq         1*drs         1*bend        1*drs    &
     1*hfq      1*drl
 tsex
     1*drl      1*hdq         1*drs         1*bend        1*drs    &
     1*hfq      1*drvs        1*hcs         1*drml
 lsex
     1*drml     1*vcs         1*drvs        1*hdq         1*drs    &
     1*bend     1*drs         1*hfq         1*drl
 half
     1*nsex     1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
#lumps
#loops
#labor
```

```
    1*fileout
    1*iden
    1*raysin
    1*half
    1*chrom
    1*mapout
    1*track
    1*fin

 twiss analysis of static map

 tunes and chromaticities for delta defined in terms of momentum deviation
 horizontal tune =  0.3301854790746201
 first order horizontal chromaticity = -5.0836441463590513E-04
 second order horizontal chromaticity =   32.78527399985234
 horizontal tune when delta =  9.9999999999999999E-04
  0.3302177559842053

 vertical tune =  6.3227244365018311E-02
 first order vertical chromaticity = -8.9058574997704199E-04
 second order vertical chromaticity =  -24.35635555300784
 vertical tune when delta =  9.9999999999999999E-04
  6.3201997423715326E-02

 tune separation when delta=  9.9999999999999999E-04
  0.2670157585604900

 normalized anharmonicities
  hhn=   38.34696150586799
  vvn=  0.6125175935029140
  hvn=   2.867346907397344



 matrix for map is :

 3.71731E-01  4.86128E+00  0.00000E+00 0.00000E+00 0.00000E+00 -3.49162E+00
-3.07966E-01 -1.33728E+00  0.00000E+00 0.00000E+00 0.00000E+00 -1.01272E-01
 0.00000E+00  0.00000E+00  5.48139E-01 2.60973E+00 0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.10952E-01 1.29610E+00 0.00000E+00  0.00000E+00
 1.11295E+00  5.16158E+00  0.00000E+00 0.00000E+00 1.00000E+00  1.05182E+01
 0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00 0.00000E+00  1.00000E+00

 nonzero elements in generating polynomial are :

  f( 28)=f( 30 00 00 )=-0.51714051077699D+00
  f( 29)=f( 21 00 00 )=-0.48790573879345D+01
  f( 33)=f( 20 00 01 )=-0.12713167778801D+01
  f( 34)=f( 12 00 00 )=-0.14356813797105D+02
  f( 38)=f( 11 00 01 )=-0.86091649476806D+01
  f( 39)=f( 10 20 00 )=-0.57480702313803D+00
  f( 40)=f( 10 11 00 )= 0.21535856911966D+01
  f( 43)=f( 10 02 00 )=-0.23385635802931D+00
```

```
f( 48)=f( 10 00 02 )=-0.37903489409162D+01
f( 49)=f( 03 00 00 )=-0.10095440524513D+02
f( 53)=f( 02 00 01 )=-0.28749629612880D+02
f( 54)=f( 01 20 00 )=-0.48676979484183D+01
f( 55)=f( 01 11 00 )= 0.25178527593201D+02
f( 58)=f( 01 02 00 )=-0.24955355122107D+02
f( 63)=f( 01 00 02 )= 0.24870572685000D+00
f( 67)=f( 00 20 01 )= 0.14190256085394D+01
f( 70)=f( 00 11 01 )=-0.62848065640486D+01
f( 76)=f( 00 02 01 )=-0.12548412723025D+02
f( 83)=f( 00 00 03 )=-0.19367900524518D+02
f( 84)=f( 40 00 00 )=-0.11226234932478D+01
f( 85)=f( 31 00 00 )=-0.18508072322998D+02
f( 89)=f( 30 00 01 )=-0.32033394382143D+01
f( 90)=f( 22 00 00 )=-0.11657515022811D+03
f( 94)=f( 21 00 01 )=-0.27687744938574D+02
f( 95)=f( 20 20 00 )= 0.36539976777681D+00
f( 96)=f( 20 11 00 )= 0.29849078410504D+01
f( 99)=f( 20 02 00 )=-0.33854616670401D+02
f(104)=f( 20 00 02 )=-0.28225116719140D+02
f(105)=f( 13 00 00 )=-0.30562227805297D+03
f(109)=f( 12 00 01 )=-0.10141517819812D+03
f(110)=f( 11 20 00 )=-0.14302896201977D+01
f(111)=f( 11 11 00 )= 0.32788559634194D+02
f(114)=f( 11 02 00 )=-0.23241803238853D+03
f(119)=f( 11 00 02 )=-0.17486265747802D+03
f(123)=f( 10 20 01 )= 0.23607064778068D+01
f(126)=f( 10 11 01 )=-0.47863190256744D+01
f(132)=f( 10 02 01 )=-0.55908627470362D+02
f(139)=f( 10 00 03 )=-0.62274396302562D+02
f(140)=f( 04 00 00 )=-0.33795772995004D+03
f(144)=f( 03 00 01 )=-0.55322509665632D+02
f(145)=f( 02 20 00 )=-0.10918051207894D+02
f(146)=f( 02 11 00 )= 0.31345401800685D+02
f(149)=f( 02 02 00 )=-0.32070340565228D+03
f(154)=f( 02 00 02 )=-0.46674394906494D+03
f(158)=f( 01 20 01 )= 0.42047841022217D+01
f(161)=f( 01 11 01 )= 0.55497877942788D+02
f(167)=f( 01 02 01 )=-0.37957949141255D+03
f(174)=f( 01 00 03 )=-0.95542857277093D+02
f(175)=f( 00 40 00 )=-0.71081436705546D+00
f(176)=f( 00 31 00 )= 0.13608837848327D+02
f(179)=f( 00 22 00 )=-0.87030851323349D+02
f(184)=f( 00 20 02 )=-0.51342116123916D+01
f(185)=f( 00 13 00 )= 0.22956951922200D+03
f(190)=f( 00 11 02 )= 0.32330038339298D+02
f(195)=f( 00 04 00 )=-0.24720909402934D+03
f(200)=f( 00 02 02 )=-0.16804924009243D+03
f(209)=f( 00 00 04 )=-0.96401192952643D+02
```

Observe that the fractional part of the horizontal tune for the half-turn transfer map is .330 . . ., and that the first order chromaticities are near zero. Next, note from the listing of the generating polynomial that the transfer map has noticeable nonlinear content.

Comment should also be made about some of the entries in *#labor*. Since the command *raysin* follows the command *iden*, the ray trace produced by *raysin* is performed using the identity map. This is simply a convenient way of making the initial conditions appear as the first few entries in the final condition file. Later, the entry *half* eventually followed by the command *track* results in the tracking of multiple turns (half-turn by half-turn) around the ring. Tracking results are recorded on the final condition file after every half turn. The first few lines of the final condition file are shown below. The first 3 lines reproduce the content of the initial condition file (as a result of the command *rays*), and subsequent lines show the result of repeatedly applying the half-turn transfer map to these 3 initial conditions.

First few lines of final condition file from tracking run

X	P_x	Y	P_y	τ	P_τ
0.10000E-01	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
0.50000E-02	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
0.25000E-02	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00
0.37290E-02	-0.31237E-02	0.00000E+00	0.00000E+00	0.11325E-01	0.00000E+00
0.18633E-02	-0.15511E-02	0.00000E+00	0.00000E+00	0.56131E-02	0.00000E+00
0.93072E-03	-0.77275E-03	0.00000E+00	0.00000E+00	0.27944E-02	0.00000E+00
-0.13783E-01	0.30100E-02	0.00000E+00	0.00000E+00	-0.50733E-03	0.00000E+00
-0.68445E-0	0.14956E-02	0.00000E+00	0.00000E+00	-0.28377E-03	0.00000E+00
-0.34099E-02	0.74557E-03	0.00000E+00	0.00000E+00	-0.14956E-03	0.00000E+00

Now look at figure 2.5.3 below. It shows a composite of results from several tracking runs. The 3 closed curves in the figure result from plotting the X, P_x content of the final condition file from the tracking run just shown and described. The remaining unbounded curves come from additional tracking runs.

Evidently, as expected, nonlinearities in the one-turn transfer map lead to instability if the betatron tune is near a 1/3 integer and the betatron amplitude is too large. It is also easy to verify that this effect is due to the chromaticity sextupoles. To see this, one simply makes another MaryLie tracking run, for the same initial conditions, with the sextupoles turned off. Figure 2.5.4 below shows the result of making and plotting such a run. Now all phase-space trajectories are stable in that they appear to lie on smooth ellipses.

Nonlinearities in the one-turn transfer map can also have significant effects on particle orbits even far away from resonant tunes. Shown below is an analysis and tracking run made for a case in which the quadrupoles have been adjusted to give horizontal and vertical tunes of 3.254... and 2.253..., respectively, and the sextupoles have been reset to again achieve very small first-order chromaticities. Evidently, as can be seen from the polynomial listing, the one-turn transfer map has noticeable nonlinear content.

```
#comment
 Exhibit 2.5.4.
 This is a study of a small static storage ring.
 Quad strengths are set to achieve horizontal and vertical tunes of
 approximately 3.25 and 2.25, respectively.
 Sextupole strengths are set to give negligible first order
 chromaticities.  This MARYLIE run will analyze the one turn transfer
 map and then track for 30,000 half turns.
```

Figure 2.5.3: Horizontal phase-space plot illustrating nonlinear instability when the horizontal tune is near a 1/3 integer value.

Figure 2.5.4: Horizontal phase-space plot made with the same initial conditions as figure 2.5.3 and sextupoles turned off.

```
 #beam
   4.869148131759700
  0.8494258478922000
   1.000000000000000
   1.000000000000000
 #menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       3.75000000000000       1.00000000000000
    1.00000000000000
  hdq      quad
   0.500000000000000      -2.20000000000000       1.00000000000000
    1.00000000000000
 hcs      sext
  0.500000000000000       4.18000000000000
 vcs      sext
  0.500000000000000      -7.84000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
   30000.0000000000       3.00000000000000      0.000000000000000E+00
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 iden     iden
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
```

```
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
#lumps
#loops
#labor
    1*fileout
    1*raysin
    1*ring
    1*chrom
    1*mapout
    1*iden
    1*half
    1*track
    1*fin
    1 ray(s) read in from file  13

twiss analysis of static map
det in fxpt is     0.4198E+01

tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.2547786892702122
first order horizontal chromaticity = -4.0271619374618478E-04
second order horizontal chromaticity =   32.65808774228796
horizontal tune when delta =  9.9999999999999999E-04
 0.2548109446417608

vertical tune =  0.2530051390161866
first order vertical chromaticity =  1.7975433225818153E-03
second order vertical chromaticity =  -22.15511182954404
vertical tune when delta =  9.9999999999999999E-04
 0.2529847814476796

tune separation when delta=  9.9999999999999999E-04
 1.8261631940811318E-03

normalized anharmonicities
 hhn=   53.24862334223061
 vvn=   26.85912432585413
 hvn=   30.24380891672483
det in fxpt is     0.4198E+01

matrix for map is :

 1.25112E+00  5.02535E+00  0.00000E+00  0.00000E+00  0.00000E+00 -9.78199E-01
-5.25419E-01 -1.31116E+00  0.00000E+00  0.00000E+00  0.00000E+00 -4.37925E-01
 0.00000E+00  0.00000E+00 -1.14286E+00  6.61944E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.41867E-01  1.10510E+00  0.00000E+00  0.00000E+00
 1.06186E+00  3.48330E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.51190E+01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= 0.19670225454821D+00
 f( 29)=f( 21 00 00 )= 0.55152652415966D+01
 f( 33)=f( 20 00 01 )=-0.52761919484391D+01
 f( 34)=f( 12 00 00 )= 0.50172369830512D+01
 f( 38)=f( 11 00 01 )=-0.41689831423080D+01
 f( 39)=f( 10 20 00 )=-0.22581270407549D+01
 f( 40)=f( 10 11 00 )= 0.93058486048550D+01
 f( 43)=f( 10 02 00 )= 0.74603243460861D+01
 f( 48)=f( 10 00 02 )=-0.22331812657246D+02
 f( 49)=f( 03 00 00 )=-0.22300968643423D+02
 f( 53)=f( 02 00 01 )= 0.16829903574911D+02
 f( 54)=f( 01 20 00 )=-0.76520697166645D+01
 f( 55)=f( 01 11 00 )= 0.53632722962837D+02
 f( 58)=f( 01 02 00 )=-0.71439832264728D+02
 f( 63)=f( 01 00 02 )=-0.43535209192095D+02
 f( 67)=f( 00 20 01 )= 0.28783874201457D-01
 f( 70)=f( 00 11 01 )=-0.24187912067743D+02
 f( 76)=f( 00 02 01 )= 0.79837794288049D+02
 f( 83)=f( 00 00 03 )=-0.41213129683146D+02
 f( 84)=f( 40 00 00 )=-0.67522289869404D+01
 f( 85)=f( 31 00 00 )=-0.44405063115873D+02
 f( 89)=f( 30 00 01 )=-0.56025060463220D+02
 f( 90)=f( 22 00 00 )=-0.96093705954530D+02
 f( 94)=f( 21 00 01 )=-0.39066499193696D+03
 f( 95)=f( 20 20 00 )=-0.20441377679327D+02
 f( 96)=f( 20 11 00 )= 0.15993051219359D+03
 f( 99)=f( 20 02 00 )=-0.31303598415936D+03
 f(104)=f( 20 00 02 )=-0.15430076765801D+03
 f(105)=f( 13 00 00 )= 0.12246481473571D+03
 f(109)=f( 12 00 01 )=-0.13128610510905D+04
 f(110)=f( 11 20 00 )=-0.73677929798981D+02
 f(111)=f( 11 11 00 )= 0.48244695142527D+03
 f(114)=f( 11 02 00 )=-0.77062655918068D+03
 f(119)=f( 11 00 02 )=-0.45537527754588D+03
 f(123)=f( 10 20 01 )=-0.88444610014570D+02
 f(126)=f( 10 11 01 )= 0.69461302445136D+03
 f(132)=f( 10 02 01 )=-0.13793237596248D+04
 f(139)=f( 10 00 03 )=-0.27258305008961D+03
 f(140)=f( 04 00 00 )= 0.64897838878841D+02
 f(144)=f( 03 00 01 )=-0.86404816059709D+03
 f(145)=f( 02 20 00 )=-0.98934033800144D+02
 f(146)=f( 02 11 00 )= 0.37345004409082D+03
 f(149)=f( 02 02 00 )=-0.27558907518709D+02
 f(154)=f( 02 00 02 )=-0.12974461873269D+04
 f(158)=f( 01 20 01 )=-0.14910629304975D+03
 f(161)=f( 01 11 01 )= 0.14394371034592D+04
 f(167)=f( 01 02 01 )=-0.34420211469510D+04
 f(174)=f( 01 00 03 )= 0.20695569207841D+02
 f(175)=f( 00 40 00 )=-0.47242815161365D+01
 f(176)=f( 00 31 00 )= 0.90692996273993D+02
 f(179)=f( 00 22 00 )=-0.65569116002824D+03
```

```
f(184)=f( 00 20 02 )=-0.78455200631995D+02
f(185)=f( 00 13 00 )= 0.21448424880106D+04
f(190)=f( 00 11 02 )= 0.48481894507819D+03
f(195)=f( 00 04 00 )=-0.27992273467860D+04
f(200)=f( 00 02 02 )=-0.54256518967410D+03
f(209)=f( 00 00 04 )=-0.33688405208407D+03
```

With regard to the tracking calculations performed in this particular MaryLie run, the first few lines of the final condition file (the first line of which is identical to the initial condition) are listed below. They are produced by tracking a single particle for 30,000 half turns and writing out the resulting phase-space coordinates on every third half turn. (Note that the fourth parameter value associated with the command *track* in #menu is 3.0×10^4, and the fifth parameter value is 3.)

First few lines of final condition file from tracking

X	P_x	Y	P_y	τ	P_τ
1.00000E-02	0.00000E+00	1.00000E-02	0.00000E+00	0.00000E+00	0.00000E+00
-1.83645E-03	3.93826E-03	-1.27542E-02	-2.00523E-03	-4.94300E-03	0.00000E+00
-1.17979E-02	5.51657E-03	1.00493E-02	3.23615E-03	-5.59053E-03	0.00000E+00
-1.61643E-02	4.21440E-03	-2.36376E-03	-2.58659E-03	-1.42556E-03	0.00000E+00

Horizontal and vertical phase-space projections produced by plotting the X, P_x and Y, P_y contents of the entire final condition file are shown below in figures 2.5.5 and 2.5.6. Observe that the horizontal and vertical phase-space projections, even for a single particle, do not form nice ellipses. Rather, each ellipse appears to be braided, and if the tracking run were extended for enough turns, would fill out a thick band. It can be shown that this effect arises from the nonlinear interaction between the horizontal and vertical degrees of freedom due primarily to sextupoles. Indeed, if the sextupoles are turned off, both phase-space plots become nice ellipses. Moreover, if the sextupoles are left on but only the horizontal degree of freedom is excited (rather than both horizontal and vertical as in this example), then the phase- space plot is again a thin (although somewhat distorted) ellipse.

One might wonder about the long-term effect of the nonlinearities in this example. Do the thickened ellipses that appear in figures 2.5.5 and 2.5.6 become continually thicker and larger as time goes on eventually leading to particle loss? Or do they merely thicken a finite amount but remain bounded? One way to study this question is to use the MaryLie normal form options. They enable one to distinguish, at least in part, between nonlinear phase- space behavior that can be predicted and is harmless to long-term stability, and phase-space behavior that is chaotic and potentially unstable. Their use is illustrated, among other things, in the next section.

Figure 2.5.5: Horizontal phase-space projection for trajectory of a single particle in a ring with sextupoles.

Figure 2.5.6: Vertical phase-space projection for trajectory of a single particle in a ring with sextupoles.

2.6 Small Storage Ring with Bunchers

The example of the small static storage ring treated a time independent system. The purpose of this section is to give an example of the application of MaryLie to a time dependent system. This system will be a small electron storage ring with radio frequency bunching cavities. It is patterned after a design under consideration for construction at Karlsruhe. (In actuality, the Los Alamos Proton Storage Ring, upon which the small static storage ring example of section 2.5 was modeled, also has radio frequency cavities. However, their effect was ignored in order to simplify the earlier discussion.)

Consider the small electron storage ring shown in figure 2.6.1. It is designed to circulate approximately 1.4 GeV electrons for the production of synchrotron radiation. The ring consists of 4 identical arcs and two radio frequency cavities. Each arc is composed of various drifts, a 90° superconducting bending magnet, 4 quadrupoles, and 2 sextupoles (not shown). The lattice has 4-fold symmetry when the cavities are turned off, and 2-fold symmetry when they are identically powered.

Figure 2.6.1: Layout of Electron Storage Ring illustrating 4 arcs and 2 RF cavities.

The content of an arc is shown in figure 2.6.2. Note that the arc has reflection symmetry about the center of the dipole if the effect of the sextupoles is ignored.

The results of a MaryLie run to study various aspects of this ring are shown below. The quadrupoles have been set to achieve static horizontal and vertical tunes of 2.176 . . . and 1.118 . . ., respectively; and the sextupoles have been set to achieve near zero first-order chromaticities.

Figure 2.6.2: Details and dimensions of arc for the Electron Storage Ring. The sextupoles shown here are not shown in figure 2.6.1.

```
#comment
 Exhibit 2.6.1.
 This is a study of a small ring with buncher.
 This MARYLIE run will first calculate the transverse tunes and
 chromaticities for a static ring without RF cavities.  Next it will
 compute all three tunes when the RF cavities are included.
 Finally, it will track for 12,000 turns.
#beam
  4.787400000236000
  2807.643747712000
  1.000000000000000
  1.000000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000E+00
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.6094450000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.0110200000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       8.91000000000000
 vcs      sext
  0.200000000000000      -31.2000000000000
 cvty     srfc
  -350000.000000000       502000000.000000
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 tunes    tadm
   1.00000000000000      0.000000000000000E+00   3.00000000000000
```

```
  0.000000000000000E+00
 raysin   rt
   13.0000000000000       14.0000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
   12000.0000000000       3.00000000000000      0.000000000000000E+00
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 iden     iden
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 half
     1*arc         1*cvty        1*arc
 statring
     4*arc
 dynring
     2*half
#lumps
#loops
#labor
    1*fileout
    1*statring
    1*chrom
    1*mapout
    1*iden
    1*dynring
    1*tunes
    1*mapout
    1*raysin
    1*track
    1*fin

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:
horizontal tune =  0.1767754018607071
first order horizontal chromaticity =  8.4331690833771913E-04
second order horizontal chromaticity =   61.84375239529651
horizontal tune when delta =  9.9999999999999999E-04
 0.1768380889300107

vertical tune =  0.1187408593290881
first order vertical chromaticity = -5.0971918876900108E-04
second order vertical chromaticity =   50.75190581253528
vertical tune when delta =  9.9999999999999999E-04
```

```
 0.1187911015157119

tune separation when delta=  9.9999999999999999E-04
 5.8046987414298845E-02

normalized anharmonicities
 hhn=   26.37172974127064
 vvn=   68.90926349103454
 hvn=  -161.4909890590244

matrix for map is :

 4.44023E-01 7.63417E+00  0.00000E+00 0.00000E+00  0.00000E+00 -1.01718E+00
-1.05164E-01 4.44023E-01  0.00000E+00 0.00000E+00  0.00000E+00 -1.92402E-01
 0.00000E+00 0.00000E+00  7.34361E-01 1.41205E+00  0.00000E+00  0.00000E+00
 0.00000E+00 0.00000E+00 -3.26273E-01 7.34361E-01  0.00000E+00  0.00000E+00
 1.92402E-01 1.01718E+00  0.00000E+00 0.00000E+00  1.00000E+00 -1.94986E+00
 0.00000E+00 0.00000E+00  0.00000E+00 0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= 0.45215379538286D-01
 f( 29)=f( 21 00 00 )=-0.22046694853331D+00
 f( 33)=f( 20 00 01 )= 0.21307843039387D-01
 f( 34)=f( 12 00 00 )=-0.34001019319329D+01
 f( 38)=f( 11 00 01 )=-0.80852510867486D+01
 f( 39)=f( 10 20 00 )=-0.23700155752496D+01
 f( 40)=f( 10 11 00 )=-0.72221004647340D+01
 f( 43)=f( 10 02 00 )= 0.99716164303407D+01
 f( 48)=f( 10 00 02 )=-0.24163272701699D+00
 f( 49)=f( 03 00 00 )= 0.24871284484431D+02
 f( 53)=f( 02 00 01 )= 0.10293038577256D+02
 f( 54)=f( 01 20 00 )= 0.50118468271474D+01
 f( 55)=f( 01 11 00 )=-0.44809549519846D+01
 f( 58)=f( 01 02 00 )= 0.10485631631995D+02
 f( 63)=f( 01 00 02 )=-0.10110704197038D+02
 f( 67)=f( 00 20 01 )=-0.43240254040841D+01
 f( 70)=f( 00 11 01 )=-0.14604513970980D+02
 f( 76)=f( 00 02 01 )= 0.18184820623932D+02
 f( 83)=f( 00 00 03 )=-0.30157132940637D+01
 f( 84)=f( 40 00 00 )=-0.81736280922983D+00
 f( 85)=f( 31 00 00 )= 0.11682773681669D+02
 f( 89)=f( 30 00 01 )=-0.57373843211935D+01
 f( 90)=f( 22 00 00 )=-0.10667462524008D+03
 f( 94)=f( 21 00 01 )= 0.57774080071438D+02
 f( 95)=f( 20 20 00 )= 0.10610937449426D+02
 f( 96)=f( 20 11 00 )=-0.25346373720784D+02
 f( 99)=f( 20 02 00 )=-0.17887202924927D+01
 f(104)=f( 20 00 02 )=-0.33795119642547D+02
 f(105)=f( 13 00 00 )= 0.55094300709058D+03
 f(109)=f( 12 00 01 )=-0.34940813399429D+03
 f(110)=f( 11 20 00 )= 0.47360502589164D+02
 f(111)=f( 11 11 00 )=-0.35352607194411D+02
```

```
 f(114)=f( 11 02 00 )= 0.16557547039968D+03
 f(119)=f( 11 00 02 )= 0.25564377797871D+03
 f(123)=f( 10 20 01 )= 0.54619011834984D+02
 f(126)=f( 10 11 01 )=-0.14758283906768D+03
 f(132)=f( 10 02 01 )=-0.72171909711283D+02
 f(139)=f( 10 00 03 )=-0.82876422714614D+02
 f(140)=f( 04 00 00 )=-0.12623673101388D+04
 f(144)=f( 03 00 01 )= 0.11300019453141D+04
 f(145)=f( 02 20 00 )=-0.11207086004803D+03
 f(146)=f( 02 11 00 )= 0.28089975149293D+03
 f(149)=f( 02 02 00 )=-0.85697906810901D+03
 f(154)=f( 02 00 02 )=-0.74989176416074D+03
 f(158)=f( 01 20 01 )= 0.95032003682075D+02
 f(161)=f( 01 11 01 )= 0.46315498241881D+03
 f(167)=f( 01 02 01 )= 0.17210946628600D+03
 f(174)=f( 01 00 03 )= 0.34618343480949D+03
 f(175)=f( 00 40 00 )=-0.39584960702938D+00
 f(176)=f( 00 31 00 )= 0.34012760236674D+02
 f(179)=f( 00 22 00 )=-0.29803015500570D+02
 f(184)=f( 00 20 02 )= 0.22800959614659D+02
 f(185)=f( 00 13 00 )=-0.13414827063085D+03
 f(190)=f( 00 11 02 )=-0.16360310517214D+03
 f(195)=f( 00 04 00 )= 0.41037425482445D+02
 f(200)=f( 00 02 02 )=-0.29158811088700D+03
 f(209)=f( 00 00 04 )=-0.82952558576731D+02

twiss analysis of dynamic map

horizontal tune =  0.1766133182729722
vertical tune =  0.1187408593290881
temporal tune = -1.7314966426091506E-02

normalized anharmonicities
 hhn=   26.32234081842545
 vvn=   68.66906498937427
 ttn=   2.564081220741627
 hvn=  -161.0403476609988
 htn=  0.8264933075857925
 vtn=  0.5946233516118033

matrix for map is :

 4.45947E-01 7.57790E+00  0.00000E+00 0.00000E+00 -1.70376E-02 -9.93256E-01
-1.05212E-01 4.45947E-01  0.00000E+00 0.00000E+00  5.58342E-04 -1.92985E-01
 0.00000E+00 0.00000E+00  7.34361E-01 1.41205E+00  0.00000E+00  0.00000E+00
 0.00000E+00 0.00000E+00 -3.26273E-01 7.34361E-01  0.00000E+00  0.00000E+00
 1.92985E-01 9.93256E-01  0.00000E+00 0.00000E+00  9.93076E-01 -1.94279E+00
-5.58342E-04 1.70376E-02  0.00000E+00 0.00000E+00  5.12524E-03  9.93076E-01
 nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= 0.45272402371279D-01
 f( 29)=f( 21 00 00 )=-0.21619291913033D+00
 f( 32)=f( 20 00 10 )= 0.52159116379009D-03
```

```
f( 33)=f( 20 00 01 )= 0.22809711783722D-01
f( 34)=f( 12 00 00 )=-0.33266111187308D+01
f( 37)=f( 11 00 10 )= 0.27446242180918D-01
f( 38)=f( 11 00 01 )=-0.80348939219789D+01
f( 39)=f( 10 20 00 )=-0.23721440107942D+01
f( 40)=f( 10 11 00 )=-0.72230756994141D+01
f( 43)=f( 10 02 00 )= 0.99743831183019D+01
f( 46)=f( 10 00 20 )=-0.14198633319081D-04
f( 47)=f( 10 00 11 )= 0.69285355799949D-02
f( 48)=f( 10 00 02 )=-0.22958036119541D+00
f( 49)=f( 03 00 00 )= 0.24283780818181D+02
f( 52)=f( 02 00 10 )=-0.17675339230659D+00
f( 53)=f( 02 00 01 )= 0.99501518205640D+01
f( 54)=f( 01 20 00 )= 0.49357178993046D+01
f( 55)=f( 01 11 00 )=-0.45466022612201D+01
f( 58)=f( 01 02 00 )= 0.10617210625343D+02
f( 61)=f( 01 00 20 )= 0.37055997589258D-03
f( 62)=f( 01 00 11 )=-0.27990717530384D-01
f( 63)=f( 01 00 02 )=-0.10072004315623D+02
f( 66)=f( 00 20 10 )=-0.22404457233068D-01
f( 67)=f( 00 20 01 )=-0.43501278829361D+01
f( 69)=f( 00 11 10 )=-0.18153809343687D-01
f( 70)=f( 00 11 01 )=-0.14615671070792D+02
f( 75)=f( 00 02 10 )= 0.37486925842262D-01
f( 76)=f( 00 02 01 )= 0.18217899125922D+02
f( 80)=f( 00 00 30 )=-0.11894584377677D-06
f( 81)=f( 00 00 21 )=-0.27911087363220D-04
f( 82)=f( 00 00 12 )= 0.24256888920499D-01
f( 83)=f( 00 00 03 )=-0.29893516527248D+01
f( 84)=f( 40 00 00 )=-0.81849404478391D+00
f( 85)=f( 31 00 00 )= 0.11651079604483D+02
f( 88)=f( 30 00 10 )=-0.13920760824296D-01
f( 89)=f( 30 00 01 )=-0.57505430076205D+01
f( 90)=f( 22 00 00 )=-0.10599786106222D+03
f( 93)=f( 21 00 10 )= 0.24172901812277D+00
f( 94)=f( 21 00 01 )= 0.58020142165025D+02
f( 95)=f( 20 20 00 )= 0.10625101726234D+02
f( 96)=f( 20 11 00 )=-0.25357087882830D+02
f( 99)=f( 20 02 00 )=-0.17881415870767D+01
f(102)=f( 20 00 20 )=-0.23139584357480D-02
f(103)=f( 20 00 11 )=-0.86155728411927D-03
f(104)=f( 20 00 02 )=-0.33800412976428D+02
f(105)=f( 13 00 00 )= 0.54463794602174D+03
f(108)=f( 12 00 10 )=-0.22781682264849D+01
f(109)=f( 12 00 01 )=-0.35293093264212D+03
f(110)=f( 11 20 00 )= 0.47692799517677D+02
f(111)=f( 11 11 00 )=-0.35974719511946D+02
f(114)=f( 11 02 00 )= 0.16629562793653D+03
f(117)=f( 11 00 20 )=-0.93974817575462D-01
f(118)=f( 11 00 11 )=-0.53253606073286D+00
f(119)=f( 11 00 02 )= 0.25492742947047D+03
f(122)=f( 10 20 10 )= 0.88421335400037D-01
f(123)=f( 10 20 01 )= 0.54785968578495D+02
```

```
f(125)=f( 10 11 10 )=-0.12029885863179D+00
f(126)=f( 10 11 01 )=-0.14784356766062D+03
f(131)=f( 10 02 10 )= 0.15288776819962D+00
f(132)=f( 10 02 01 )=-0.72003436494803D+02
f(136)=f( 10 00 30 )=-0.10249898861021D-01
f(137)=f( 10 00 21 )=-0.50678910476843D-01
f(138)=f( 10 00 12 )=-0.11701369149759D+00
f(139)=f( 10 00 03 )=-0.82981002985973D+02
f(140)=f( 04 00 00 )=-0.12462350349190D+04
f(143)=f( 03 00 10 )= 0.19943296100117D+01
f(144)=f( 03 00 01 )= 0.11328563670199D+04
f(145)=f( 02 20 00 )=-0.11067647113872D+03
f(146)=f( 02 11 00 )= 0.27084848589679D+03
f(149)=f( 02 02 00 )=-0.84834634851423D+03
f(152)=f( 02 00 20 )=-0.15917450414469D+01
f(153)=f( 02 00 11 )=-0.39905668810137D+01
f(154)=f( 02 00 02 )=-0.75350189263427D+03
f(157)=f( 01 20 10 )= 0.38202021663940D+00
f(158)=f( 01 20 01 )= 0.96791184584807D+02
f(160)=f( 01 11 10 )=-0.28565426214560D+01
f(161)=f( 01 11 01 )= 0.45678847238498D+03
f(166)=f( 01 02 10 )= 0.24310048159471D+01
f(167)=f( 01 02 01 )= 0.17905042205389D+03
f(171)=f( 01 00 30 )=-0.31324939973368D+00
f(172)=f( 01 00 21 )=-0.12181205220735D+01
f(173)=f( 01 00 12 )=-0.21760723071158D+01
f(174)=f( 01 00 03 )= 0.34501431914347D+03
f(175)=f( 00 40 00 )=-0.39022737321736D+00
f(176)=f( 00 31 00 )= 0.33968821547973D+02
f(179)=f( 00 22 00 )=-0.29783342441179D+02
f(182)=f( 00 20 20 )=-0.38273027582899D-03
f(183)=f( 00 20 11 )= 0.41191036080077D+00
f(184)=f( 00 20 02 )= 0.23321286693937D+02
f(185)=f( 00 13 00 )=-0.13403115928927D+03
f(188)=f( 00 11 20 )= 0.19164445611293D-02
f(189)=f( 00 11 11 )=-0.76827567222306D+00
f(190)=f( 00 11 02 )=-0.16456662147632D+03
f(195)=f( 00 04 00 )= 0.40941098905118D+02
f(198)=f( 00 02 20 )=-0.18844971967649D-02
f(199)=f( 00 02 11 )= 0.10120639947267D+01
f(200)=f( 00 02 02 )=-0.29069138010614D+03
f(205)=f( 00 00 40 )=-0.23553455722979D-01
f(206)=f( 00 00 31 )=-0.12704550327439D+00
f(207)=f( 00 00 22 )=-0.32046328354160D+00
f(208)=f( 00 00 13 )=-0.35661776470151D+00
f(209)=f( 00 00 04 )=-0.83158327801083D+02
```

As indicated in the *#comments* entry, and as can be inferred from *#labor*, this MaryLie run (thanks to the line *statring* and the commands *statanal* and *chrom*) first analyzes the properties of the one-turn transfer map when the RF cavities are turned off (omitted from the ring). Inspection of the tunes and chromaticities obtained for this case of a static ring shows that they do indeed have the desired values. Subsequently, the one-turn transfer map

of the static ring is written out for future reference.

Next (thanks to the line *dynring* and the command *tunes*), the horizontal, vertical, and temporal (synchrotron) tunes are computed for the time dependent (dynamic) one-turn transfer map. That the RF cavities are now included can be seen by observing that the line *dynring* is defined in terms of the line *half*, and the line *half* includes the element *cvty*. Inspection of the element *cvty* in *#menu* shows that it refers to a short (zero length) RF buncher having a voltage of 350 KeV and a frequency of 502 MHz. Note that the sign of the cavity voltage is negative, as is required to achieve stable synchrotron motion, since the ring is operating above transition. See section 6.13 and 8.1.

The effect of the RF cavities can be seen analytically in at least two ways. First, observe that the synchrotron tune is nonzero when the cavities are turned on. Moreover, comparison of the horizontal tunes for the static and dynamic rings shows that the introduction of the RF cavities slightly changes the horizontal tune. Consequently, there is synchro-betatron coupling.

Second, it should be noted that the one-turn transfer maps for the static and dynamic rings are different. In particular, the 6,5 entry of the matrix for the dynamic one-turn transfer map is nonzero. Also, there are many more entries in the generating polynomial for the nonlinear part of the dynamic one-turn transfer map.

The effect of the RF cavities is also apparent in the phase-space data generated by the tracking portion of the MaryLie run. The first few lines of the final condition file (the first line of which is identical to the initial condition) are listed below. They are produced by tracking a single particle for 12,000 turns and writing out the resulting phase-space coordinates every third turn. Note that the P_τ phase-space coordinate (related to the energy) now changes due to synchrotron oscillation.

First few lines of final condition file from tracking run

X	P_x	Y	P_y	τ	P_τ
0.50000E-02	0.00000E+00	0.50000E-02	0.00000E+00	0.00000E+00	0.10000E-02
-0.75953E-02	0.14178E-03	-0.35432E-02	-0.21199E-02	-0.69645E-02	0.93484E-03
0.35812E-02	-0.85306E-04	-0.19390E-02	0.26666E-02	-0.12571E-01	0.79454E-03
-0.66481E-02	0.10591E-03	0.52487E-02	-0.70013E-03	-0.17555E-01	0.54973E-03
0.56676E-02	-0/32624E-03	-0.40548E-02	-0.13211E-02	-0.19619E-01	0.279095E-03

Horizontal, vertical, and temporal phase-space projections produced by plotting the X, P_x and Y, P_y and τ, P_τ contents of the entire final condition file are shown in figures 2.6.3, 2.6.4, and 2.6.5. These figures illustrate two features that are worthy of attention here. First, figure 2.6.5 shows that there are indeed synchrotron oscillations as expected. Second, all 3 figures indicate that the 3 phase-space projections, even for a single particle, do not lie on nice ellipses. As was the case with the static storage ring, it can be shown that the "thickening" of the horizontal and vertical ellipses is due in part to nonlinear sextupole effects. Moreover in this case, the thickening of the temporal ellipse is due to synchro-betatron coupling, and additional thickening of the horizontal ellipse is caused by dispersion.

At this point, as with the case of the static storage ring, one may again wonder about the long-term effect of nonlinearities. One way to approach this question is to proceed as follows: In the case of no nonlinearities, the phase-space trajectory of a stable particle lies on

Figure 2.6.3: Horizontal phase-space projection for trajectory of a single particle in a ring with sextupoles and bunchers.

Figure 2.6.4: Vertical phase-space projection for trajectory of a single particle in a ring with sextupoles and bunchers.

Figure 2.6.5: Temporal phase-space projection for trajectory of a single particle in a ring with sextupoles and bunchers.

a 3-dimensional torus in 6-dimensional phase space. (The horizontal, vertical, and temporal projections of this torus then produce three nice ellipses.) One might hope that the addition of small nonlinear effects simply distorts this torus. Indeed, the existence of so-called KAM tori can be proved in some cases. Moreover, using Lie algebraic techniques, it is possible to compute the shape such distorted tori would have. In fact, it is possible to compute a "normal form" transformation map whose inverse would transform such a distorted torus back into a *perfect* torus. (A perfect torus is the topological product of circles.) It is this transformation inverse map that is computed by the commands *dnor*, *normap*, and *inv*. See the line *cleanup*. Finally, MaryLie could be used to apply this inverse map to the results of a tracking run, and the transformed phase-space points thus obtained should lie on a perfect torus.

The results of following this procedure for the case at hand are illustrated in the MaryLie run shown below. This run produces tracking data and computes the inverse normalizing map. The command *mapout* prints out this map to show the reader that the inverse normal form map has nontrivial linear and nonlinear content. Finally, this map is applied to the tracking data with the aid of the commands *phasein* and *xfrmdata*.

```
#comment
 Exhibit 2.6.2.
 This is a study of a small ring with buncher.
 This MARYLIE run will compute the one-turn map and the normalizing map
 (script A) for the dynamic ring.  Then it will track for 12,000 turns.
 Finally, it will apply script A inverse to the tracking (phase space) data.
#beam
```

```
  4.787400000236000
  2807.643747712000
  1.000000000000000
  1.000000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000E+00
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.6094450000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.0110200000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       8.91000000000000
 vcs      sext
  0.200000000000000      -31.2000000000000
 cvty     srfc
  -350000.000000000       502000000.000000
 inv      inv
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 tunes    tadm
   1.00000000000000      0.000000000000000E+00   3.00000000000000
  0.000000000000000E+00
 dnor     dnor
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00
 normap   gbuf
   1.00000000000000       1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
   12000.0000000000       3.00000000000000      0.000000000000000E+00
 phasein  rt
   14.0000000000000       15.0000000000000     -1.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 xfrmdata rt
  0.000000000000000E+00   15.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000      0.000000000000000E+00
 fin      end
```

```
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 iden     iden
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 half
     1*arc         1*cvty        1*arc
 statring
     4*arc
 dynring
     2*half
 cleanup
    dnor  iden  normap  inv  mapout  phasein  xfrmdata
#lumps
#loops
#labor
    1*fileout
    1*dynring
    1*raysin
    1*track
    1*cleanup
    1*fin

 matrix for map is :

 3.42760E-01 -3.85186E-34 -3.04104E-20  1.58120E-20 -9.09186E-19  6.21097E-01
 1.06109E-17  2.91212E+00  6.12098E-20 -3.18262E-20 -2.96632E-03  3.74484E-17
 3.85291E-17 -2.27930E-16  6.93319E-01  0.00000E+00  1.14384E-17  2.82857E-16
 4.12874E-17  1.76283E-16 -4.21693E-17  1.44234E+00  6.66292E-17  1.94556E-17
 8.51799E-18  3.93992E-01 -4.48665E-18  2.33285E-18  2.17430E-01  0.00000E+00
-4.67616E-03 -6.90005E-16  4.10774E-17 -2.13583E-17 -1.81334E-17  4.59072E+00

 nonzero elements in generating polynomial are :

  f( 28)=f( 30 00 00 )=-0.10460366964122D+01
  f( 29)=f( 21 00 00 )= 0.69200308272702D+00
  f( 30)=f( 20 10 00 )= 0.19898531389056D-15
  f( 31)=f( 20 01 00 )=-0.97347766627371D-14
  f( 32)=f( 20 00 10 )=-0.83158066949627D-02
  f( 33)=f( 20 00 01 )= 0.60612509412607D+00
  f( 34)=f( 12 00 00 )= 0.16648016481781D+00
  f( 35)=f( 11 10 00 )= 0.27279872506370D-14
  f( 36)=f( 11 01 00 )= 0.42375978544118D-15
  f( 37)=f( 11 00 10 )= 0.23026162999557D-01
  f( 38)=f( 11 00 01 )= 0.58613392995786D+00
  f( 39)=f( 10 20 00 )= 0.20511507145183D+02
  f( 40)=f( 10 11 00 )=-0.29972861950604D+02
```

```
f( 41)=f( 10 10 10 )=-0.86409166793028D-14
f( 42)=f( 10 10 01 )= 0.27786813771905D-14
f( 43)=f( 10 02 00 )=-0.24571580215646D+02
f( 44)=f( 10 01 10 )=-0.14373908883572D-13
f( 45)=f( 10 01 01 )=-0.10103670383497D-14
f( 46)=f( 10 00 20 )= 0.94716588280191D-03
f( 47)=f( 10 00 11 )= 0.12964325061954D-01
f( 48)=f( 10 00 02 )=-0.59465760190480D-01
f( 49)=f( 03 00 00 )=-0.24197396160921D-01
f( 50)=f( 02 10 00 )=-0.21098197225429D-14
f( 51)=f( 02 01 00 )=-0.34622240280119D-14
f( 52)=f( 02 00 10 )= 0.20963195822836D-01
f( 53)=f( 02 00 01 )=-0.20161635759263D+00
f( 54)=f( 01 20 00 )= 0.16460477389170D+02
f( 55)=f( 01 11 00 )= 0.28998616728426D+02
f( 56)=f( 01 10 10 )= 0.10829774863246D-13
f( 57)=f( 01 10 01 )= 0.21228854055368D-14
f( 58)=f( 01 02 00 )=-0.19433941567307D+02
f( 59)=f( 01 01 10 )=-0.10830474022225D-13
f( 60)=f( 01 01 01 )= 0.22334003872122D-14
f( 61)=f( 01 00 20 )= 0.14647159365356D-02
f( 62)=f( 01 00 11 )=-0.10199373053827D-01
f( 63)=f( 01 00 02 )=-0.17738193455811D-01
f( 64)=f( 00 30 00 )=-0.22237968545890D-14
f( 65)=f( 00 21 00 )=-0.10577369403063D-14
f( 66)=f( 00 20 10 )= 0.74266804499367D-01
f( 67)=f( 00 20 01 )= 0.36656647331060D+00
f( 68)=f( 00 12 00 )= 0.20728918613886D-14
f( 69)=f( 00 11 10 )=-0.76396352091883D-01
f( 70)=f( 00 11 01 )= 0.13890161158937D+00
f( 71)=f( 00 10 20 )= 0.81737494981975D-18
f( 72)=f( 00 10 11 )= 0.18562798797641D-15
f( 73)=f( 00 10 02 )= 0.21835856493026D-16
f( 74)=f( 00 03 00 )=-0.63262946709291D-14
f( 75)=f( 00 02 10 )=-0.12123175636308D+00
f( 76)=f( 00 02 01 )= 0.18273761587177D+00
f( 77)=f( 00 01 20 )=-0.19969271071889D-16
f( 78)=f( 00 01 11 )= 0.10002559395463D-15
f( 79)=f( 00 01 02 )= 0.30290671594400D-16
f( 80)=f( 00 00 30 )=-0.17525279110915D+00
f( 81)=f( 00 00 21 )=-0.90165480465597D-05
f( 82)=f( 00 00 12 )=-0.26261509257058D+00
f( 83)=f( 00 00 03 )= 0.27205651724059D-02
f( 84)=f( 40 00 00 )= 0.64895158004707D+00
f( 85)=f( 31 00 00 )=-0.63078289525923D+02
f( 86)=f( 30 10 00 )= 0.11075828167483D-12
f( 87)=f( 30 01 00 )= 0.38733655058272D-12
f( 88)=f( 30 00 10 )= 0.19730226060826D-01
f( 89)=f( 30 00 01 )= 0.11022865304340D+00
f( 90)=f( 22 00 00 )=-0.57605066262867D+00
f( 91)=f( 21 10 00 )=-0.33862756313845D-12
f( 92)=f( 21 01 00 )= 0.30753675862786D-12
f( 93)=f( 21 00 10 )=-0.25914603597188D+00
```

```
f( 94)=f( 21 00 01 )= 0.31074041102328D+00
f( 95)=f( 20 20 00 )=-0.22929114528247D+03
f( 96)=f( 20 11 00 )= 0.13078909602871D+03
f( 97)=f( 20 10 10 )=-0.30794334939343D-12
f( 98)=f( 20 10 01 )=-0.14594010721922D-12
f( 99)=f( 20 02 00 )=-0.27168370001907D+03
f(100)=f( 20 01 10 )=-0.47781000303471D-12
f(101)=f( 20 01 01 )= 0.12308842943721D-12
f(102)=f( 20 00 20 )= 0.72400292463216D-01
f(103)=f( 20 00 11 )=-0.17985311762636D+02
f(104)=f( 20 00 02 )= 0.27908608434503D+00
f(105)=f( 13 00 00 )=-0.36988524143543D+02
f(106)=f( 12 10 00 )= 0.17835763545761D-12
f(107)=f( 12 01 00 )= 0.41731695485390D-12
f(108)=f( 12 00 10 )= 0.19334186177349D+00
f(109)=f( 12 00 01 )=-0.40065051923326D+00
f(110)=f( 11 20 00 )= 0.20520120279298D+03
f(111)=f( 11 11 00 )= 0.31013799054283D+03
f(112)=f( 11 10 10 )=-0.75021472111021D-13
f(113)=f( 11 10 01 )= 0.13695012017578D-13
f(114)=f( 11 02 00 )= 0.21711188678069D+03
f(115)=f( 11 01 10 )= 0.18984684498544D-12
f(116)=f( 11 01 01 )= 0.30265389873074D-13
f(117)=f( 11 00 20 )= 0.75110205029637D+00
f(118)=f( 11 00 11 )= 0.32024270651727D-01
f(119)=f( 11 00 02 )=-0.46166662933022D+01
f(120)=f( 10 30 00 )= 0.36316852919621D-12
f(121)=f( 10 21 00 )=-0.39273965881445D-13
f(122)=f( 10 20 10 )=-0.28267347981104D+02
f(123)=f( 10 20 01 )= 0.21197033992734D+02
f(124)=f( 10 12 00 )= 0.39437038957407D-12
f(125)=f( 10 11 10 )= 0.19233218494030D+02
f(126)=f( 10 11 01 )= 0.18537290025478D+03
f(127)=f( 10 10 20 )=-0.41117546018783D-14
f(128)=f( 10 10 11 )= 0.49428353684671D-13
f(129)=f( 10 10 02 )= 0.16279858176900D-14
f(130)=f( 10 03 00 )=-0.15485952548739D-12
f(131)=f( 10 02 10 )= 0.17189421279801D+02
f(132)=f( 10 02 01 )=-0.33690145720357D+02
f(133)=f( 10 01 20 )= 0.96954561663828D-14
f(134)=f( 10 01 11 )=-0.30591978459213D-13
f(135)=f( 10 01 02 )= 0.12746563234660D-13
f(136)=f( 10 00 30 )= 0.55260023701988D+01
f(137)=f( 10 00 21 )=-0.30169192630793D-01
f(138)=f( 10 00 12 )=-0.35631152567884D+00
f(139)=f( 10 00 03 )=-0.40336903343816D-01
f(140)=f( 04 00 00 )=-0.45693469250418D+00
f(141)=f( 03 10 00 )=-0.44692845707631D-12
f(142)=f( 03 01 00 )= 0.20478311095529D-12
f(143)=f( 03 00 10 )=-0.13035493863339D+00
f(144)=f( 03 00 01 )=-0.56851306886882D+00
f(145)=f( 02 20 00 )= 0.21162300673632D+03
f(146)=f( 02 11 00 )= 0.11091257274726D+03
```

```
f(147)=f( 02 10 10 )=-0.24894743851286D-12
f(148)=f( 02 10 01 )=-0.10421190733649D-12
f(149)=f( 02 02 00 )= 0.28935183856521D+03
f(150)=f( 02 01 10 )=-0.26036453147490D-12
f(151)=f( 02 01 01 )= 0.78091269768035D-13
f(152)=f( 02 00 20 )=-0.13578593989600D+00
f(153)=f( 02 00 11 )=-0.18535719712035D+02
f(154)=f( 02 00 02 )=-0.21570043691224D+00
f(155)=f( 01 30 00 )=-0.23661322271043D-12
f(156)=f( 01 21 00 )= 0.36850780905810D-12
f(157)=f( 01 20 10 )= 0.16104994058979D+01
f(158)=f( 01 20 01 )=-0.56892181282795D+02
f(159)=f( 01 12 00 )=-0.70371945371337D-13
f(160)=f( 01 11 10 )=-0.42482532637124D+02
f(161)=f( 01 11 01 )= 0.52043569511038D+02
f(162)=f( 01 10 20 )=-0.12538448629495D-13
f(163)=f( 01 10 11 )= 0.32841884726300D-13
f(164)=f( 01 10 02 )=-0.88716089162818D-14
f(165)=f( 01 03 00 )= 0.57227401111737D-12
f(166)=f( 01 02 10 )= 0.19596962487988D+02
f(167)=f( 01 02 01 )= 0.60883631992542D+02
f(168)=f( 01 01 20 )= 0.43902563215762D-14
f(169)=f( 01 01 11 )= 0.26597595968999D-13
f(170)=f( 01 01 02 )= 0.25601889625554D-14
f(171)=f( 01 00 30 )=-0.25813399199839D-02
f(172)=f( 01 00 21 )=-0.16594672191018D+01
f(173)=f( 01 00 12 )=-0.36820662666965D-02
f(174)=f( 01 00 03 )= 0.12221350313491D-01
f(175)=f( 00 40 00 )= 0.12727395025196D+02
f(176)=f( 00 31 00 )= 0.89783562113194D+02
f(177)=f( 00 30 10 )= 0.16641883626633D-12
f(178)=f( 00 30 01 )= 0.76428316260937D-13
f(179)=f( 00 22 00 )=-0.86484837939725D+02
f(180)=f( 00 21 10 )= 0.13322081000687D-12
f(181)=f( 00 21 01 )=-0.24050556036498D-13
f(182)=f( 00 20 20 )= 0.45911287467622D-01
f(183)=f( 00 20 11 )=-0.17484947380742D+02
f(184)=f( 00 20 02 )=-0.39878282615400D+00
f(185)=f( 00 13 00 )=-0.19935210012102D+03
f(186)=f( 00 12 10 )=-0.86440268094447D-14
f(187)=f( 00 12 01 )= 0.82130084971219D-13
f(188)=f( 00 11 20 )= 0.43263259560514D-01
f(189)=f( 00 11 11 )= 0.29449704923219D+00
f(190)=f( 00 11 02 )=-0.15708199862304D+01
f(191)=f( 00 10 30 )=-0.17342752562997D-13
f(192)=f( 00 10 21 )=-0.32687067290678D-14
f(193)=f( 00 10 12 )=-0.29825195346950D-13
f(194)=f( 00 10 03 )=-0.80511602524577D-15
f(195)=f( 00 04 00 )= 0.16100884288046D+02
f(196)=f( 00 03 10 )= 0.18642238791199D-12
f(197)=f( 00 03 01 )=-0.48128094689396D-13
f(198)=f( 00 02 20 )= 0.14478186615918D-01
f(199)=f( 00 02 11 )=-0.17410223098787D+02
```

```
f(200)=f( 00 02 02 )= 0.33839335207046D+00
f(201)=f( 00 01 30 )= 0.45114594483085D-14
f(202)=f( 00 01 21 )=-0.16264690824082D-13
f(203)=f( 00 01 12 )= 0.40966025123915D-14
f(204)=f( 00 01 03 )=-0.16660736068576D-14
f(205)=f( 00 00 40 )=-0.89667175754049D-03
f(206)=f( 00 00 31 )= 0.61215249623609D+02
f(207)=f( 00 00 22 )= 0.33779619515527D-03
f(208)=f( 00 00 13 )= 0.36555246487907D+02
f(209)=f( 00 00 04 )= 0.78407302582081D-03
```

Figures 2.6.6, 2.6.7, and 2.6.8 show the X,P_x and Y,P_y and τ,P_τ projections of the transformed phase-space data produced by plotting the final condition file of this second MaryLie run. Remarkably, all 3 plots are nearly perfect circles! Correspondingly, the transformed phase-space data lies on a nearly perfect torus. Thus, in this case, the thickening of the original ellipses in figures 2.6.3, 2.6.4, and 2.6.5 is due to apparently harmless nonlinear effects that can be understood in terms of a Lie algebraically computable map. This example illustrates that the ability to remove harmless nonlinear effects from phase-space data is an important tool for tracking studies, for one can then examine on a much finer scale whatever chaotic behavior might remain.

Figure 2.6.6: Horizontal projection of transformed phase-space data.

Figure 2.6.7: Vertical projection of transformed phase-space data.

Figure 2.6.8: Temporal projection of transformed phase-space data.

2.7 Large Storage Ring

The previous examples dealt with small systems having relatively few elements. The purpose of this section is to give an example of the application of MaryLie to a large system having a large number of elements. In particular, an illustration will be given of the use of the *#lump* and *#loop* features of MaryLie.

Consider the storage ring shown schematically in figure 2.7.1. It is a preliminary design of a ring for a Superconducting Super Collider, and is intended to store 20 TeV protons. As illustrated, the design calls for 6 equally spaced interaction points. For simplicity, it will be assumed that all 6 portions of the ring between interaction points are identical. That is, the ring will be assumed to have 6 identical superperiods.

Figure 2.7.1: Schematic of Superconducting Super Collider storage ring showing 6 equally spaced interaction points separated by superperiods.

The contents of a superperiod are shown schematically in figure 2.7.2. It is assumed that both horizontal and vertical beta functions at the interaction point have the value $\beta^* = 1$ meter. The section labeled *lob1* is a leading low beta matching section that matches the low beta values at the interaction point to the larger beta values in the rest of the ring. Similarly, *lob2* is a trailing low beta section that matches the larger beta values in the ring to the low values at the interaction point.

Figure 2.7.2: Schematic of superperiod for Superconducting Super Collider showing its various sections.

The sections labeled *tro1* and *tro2* are phase trombones. They consist of collections of quadrupoles and drifts, and are used to adjust the tunes of the ring.

The sections labeled *dis1* and *dis2* are leading and trailing dispersion matching sections. They match the nonzero dispersion values in the section labeled *arc* to zero values at the interaction points, in the low beta matching sections, and in the phase trombones.

The arc itself, as shown in figure 2.7.3, consists of two matching sections labeled *arcin* and *arcout*, and 60 cells composed of drifts, dipoles, quadrupoles, and sextupoles. The arcs provide the major bending for the ring, and all chromaticity corrections.

Figure 2.7.3: Schematic of arc for superperiod of Superconducting Super Collider.

Shown below is a MaryLie run designed to study various features of this Superconducting Super Collider design. Note that *#menu* has now taken on Chinese proportions. The contents of a superperiod are listed in the *#lines* component under the line *sixth*, and are as shown in figure 2.7.2. The contents of the various sections can be seen from the lines *lob1*, *tro1*, etc., and the various lines and elements to which they refer. Altogether, a superperiod consists of 851 elements, and the entire ring consists of 5106 elements.

```
#comment
 Exhibit 2.7.
 This is a study of a large storage ring.
 This MARYLIE run builds up the transfer map for one superperiod (one
 sixth of the ring), prints it out, and computes its tunes and
 chromaticities.
 It also traces rays through one superperiod.
 Then the run tracks for 3000 superperiods (500 turns) using the lump and
 loop features of MARYLIE.  In this case the superperiod is broken up into
 three lumps.
 Each lump contains approximately 40 sextupoles.
#beam
  66715.94873007700
  21315.60784226800
  1.000000000000000
  1.000000000000000
#menu
 drc      drft
   5.75000000000000
 drs      drft
   1.87500000000000
 drts     drft
  0.750000000000000
 drtl     drft
   94.7500000000000
 dr56     drft
   18.0000000000000
 dr34     drft
   120.000000000000
 dr1      drft
   1.00000000000000
```

```
dr45     drft
  18.7000000000000
drex     drft
  20.0000000000000
drds     drft
  95.0000000000000
bend     nbnd
 0.444444444444444      0.000000000000000E+00  0.500000000000000E+00
  6.19780700000000       1.00000000000000       1.00000000000000
qfa      quad
  2.50000000000000       150.844700000000       1.00000000000000
  1.00000000000000
qda      quad
  2.50000000000000      -150.844700000000       1.00000000000000
  1.00000000000000
qfb      quad
  5.00000000000000       149.062900000000       1.00000000000000
  1.00000000000000
qdb      quad
  5.00000000000000      -149.062900000000       1.00000000000000
  1.00000000000000
qfc      quad
  5.00000000000000       154.497200000000       1.00000000000000
  1.00000000000000
qdc      quad
  5.00000000000000      -154.497200000000       1.00000000000000
  1.00000000000000
qfd      quad
  5.00000000000000       153.572400000000       1.00000000000000
  1.00000000000000
qdd      quad
  5.00000000000000      -153.572400000000       1.00000000000000
  1.00000000000000
qf1      quad
  6.80000000000000       294.181400000000       1.00000000000000
  1.00000000000000
qd1      quad
  6.80000000000000      -294.181400000000       1.00000000000000
  1.00000000000000
qf2      quad
  10.9000000000000       246.370400000000       1.00000000000000
  1.00000000000000
qd2      quad
  10.9000000000000      -246.370400000000       1.00000000000000
  1.00000000000000
qf3      quad
  12.1000000000000       221.969400000000       1.00000000000000
  1.00000000000000
qd3      quad
  12.1000000000000      -221.969400000000       1.00000000000000
  1.00000000000000
qf4      quad
  5.80000000000000       175.429200000000       1.00000000000000
```

```
  1.00000000000000
 qd4      quad
  5.80000000000000      -175.429200000000       1.00000000000000
  1.00000000000000
 qf5      quad
  12.0000000000000       202.941600000000       1.00000000000000
  1.00000000000000
 qd5      quad
  12.0000000000000      -202.941600000000       1.00000000000000
  1.00000000000000
 qf6      quad
  4.00000000000000       48.0381000000000       1.00000000000000
  1.00000000000000
 qd6      quad
  4.00000000000000      -48.0381000000000       1.00000000000000
  1.00000000000000
 qfh      quad
  2.50000000000000       135.701600000000       1.00000000000000
  1.00000000000000
 qdh      quad
  2.50000000000000      -135.710300000000       1.00000000000000
  1.00000000000000
 qf       quad
  5.00000000000000       135.701600000000       1.00000000000000
  1.00000000000000
 qd       quad
  5.00000000000000      -135.710300000000       1.00000000000000
  1.00000000000000
 sf       sext
  2.00000000000000       146.192300000000
 sd       sext
  2.00000000000000      -232.556300000000
 chrom    tasm
  2.00000000000000      1.000000000000000E-03   1.00000000000000
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 fin      end
 circ     circ
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  3000.00000000000       1.00000000000000      0.000000000000000E+00
 raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
 0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
 iden     iden
#lines
 lob1
```

```
     1*drex        1*qf1         1*dr1         1*qf1         1*dr1      &
     1*qd2         1*dr1         1*qd2         1*dr1         1*qf3      &
     1*dr34        1*qf4         1*dr45        1*qd5         1*dr56     &
     1*qf6
 tro1
     1*drts        1*qfa         1*drtl        1*qdb         1*drtl     &
     1*qfc         1*drtl        1*qdd         1*drtl        1*qfc      &
     1*drtl        1*qdb         1*drtl        1*qfa         1*drts
 dis1
     1*qfh         1*drc         1*bend        1*drc         1*qd       &
     1*drc         1*bend        1*drc         1*qf          1*drds     &
     1*qd          1*drds
 nsex
     1*qf          1*drc         1*bend        1*drc         1*qd       &
     1*drc         1*bend        1*drc
 arcin
     3*nsex
 cell
     1*qf          1*drs         1*sf          1*drs         1*bend     &
     1*drs         1*sd          1*drs         1*qd          1*drc      &
     1*bend        1*drc
 arcout
     2*nsex        1*qf          1*drc         1*bend        1*drc      &
     1*qd
 dis2
     1*drds        1*qf          1*drds        1*qd          1*drc      &
     1*bend        1*drc         1*qf          1*drc         1*bend     &
     1*drc         1*qdh
 tro2
     1*drts        1*qda         1*drtl        1*qfb         1*drtl     &
     1*qdc         1*drtl        1*qfd         1*drtl        1*qdc      &
     1*drtl        1*qfb         1*drtl        1*qda         1*drts
 lob2
     1*qd6         1*dr56        1*qf5         1*dr45        1*qd4      &
     1*dr34        1*qd3         1*dr1         1*qf2         1*dr1      &
     1*qf2         1*dr1         1*qd1         1*dr1         1*qd1      &
     1*drex
 arc
     1*arcin      60*cell        1*arcout
 sixth
     1*lob1        1*tro1        1*dis1        1*arc         1*dis2     &
     1*tro2        1*lob2
#lumps
 lu1
     1*lob1        1*tro1        1*dis1        1*arcin      20*cell
 lu2
    20*cell
 lu3
    20*cell        1*arcout      1*dis2        1*tro2        1*lob2
#loops
 loop1
     1*lu1         1*lu2         1*lu3
#labor
```

```
    1*fileout
    1*sixth
    1*mapout
    1*chrom
    1*raysin
    1*track
    1*loop1
    1*circ
    1*fin

 matrix for map is :

-2.20484E-01 -9.76044E-01  0.00000E+00  0.00000E+00 0.00000E+00  3.31843E-05
 9.75258E-01 -2.18180E-01  0.00000E+00  0.00000E+00 0.00000E+00  9.38706E-05
 0.00000E+00  0.00000E+00 -2.17810E-01 -9.75919E-01 0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  9.75487E-01 -2.20391E-01 0.00000E+00  0.00000E+00
 5.30602E-05  8.43817E-05  0.00000E+00  0.00000E+00 1.00000E+00 -3.08744E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00  1.00000E+00

 nonzero elements in generating polynomial are :

  f( 28)=f( 30 00 00 )=-0.45311841613799D-03
  f( 29)=f( 21 00 00 )=-0.42051084540769D-03
  f( 33)=f( 20 00 01 )= 0.29648412108137D+02
  f( 34)=f( 12 00 00 )= 0.10911021904278D-02
  f( 38)=f( 11 00 01 )=-0.95249092944067D+01
  f( 39)=f( 10 20 00 )= 0.11502630453174D-02
  f( 40)=f( 10 11 00 )=-0.18054566777158D-03
  f( 43)=f( 10 02 00 )=-0.15326259212955D-02
  f( 48)=f( 10 00 02 )=-0.55062715297709D+00
  f( 49)=f( 03 00 00 )= 0.24483150102761D-03
  f( 53)=f( 02 00 01 )=-0.29683518290291D+02
  f( 54)=f( 01 20 00 )= 0.54776622918203D-03
  f( 55)=f( 01 11 00 )=-0.18847354118399D-02
  f( 58)=f( 01 02 00 )=-0.57848883740275D-03
  f( 63)=f( 01 00 02 )= 0.23001005099699D+00
  f( 67)=f( 00 20 01 )=-0.24757716825927D+02
  f( 70)=f( 00 11 01 )=-0.33941869234081D+02
  f( 76)=f( 00 02 01 )= 0.24813595098500D+02
  f( 83)=f( 00 00 03 )= 0.12041613026027D+01
  f( 84)=f( 40 00 00 )= 0.53719143669651D+03
  f( 85)=f( 31 00 00 )=-0.65032625055806D+02
  f( 89)=f( 30 00 01 )= 0.13709185308574D+03
  f( 90)=f( 22 00 00 )= 0.11876136002758D+04
  f( 94)=f( 21 00 01 )=-0.15643146269412D+04
  f( 95)=f( 20 20 00 )= 0.45029248023302D+04
  f( 96)=f( 20 11 00 )=-0.46566047360800D+04
  f( 99)=f( 20 02 00 )= 0.32549479751298D+04
  f(104)=f( 20 00 02 )= 0.12224260475573D+04
  f(105)=f( 13 00 00 )= 0.55183722231071D+02
  f(109)=f( 12 00 01 )= 0.12811016337292D+04
  f(110)=f( 11 20 00 )= 0.39631314514662D+04
  f(111)=f( 11 11 00 )= 0.39099667109689D+04
```

```
 f(114)=f( 11 02 00 )=-0.42920577299781D+04
 f(119)=f( 11 00 02 )=-0.80112587128335D+04
 f(123)=f( 10 20 01 )= 0.19710683960939D+04
 f(126)=f( 10 11 01 )=-0.35423279001297D+04
 f(132)=f( 10 02 01 )= 0.43932597295168D+03
 f(139)=f( 10 00 03 )= 0.93187862459852D+02
 f(140)=f( 04 00 00 )= 0.94129796938971D+02
 f(144)=f( 03 00 01 )=-0.12848761161860D+03
 f(145)=f( 02 20 00 )= 0.32433986263860D+04
 f(146)=f( 02 11 00 )= 0.43007468516398D+04
 f(149)=f( 02 02 00 )= 0.44206956850141D+04
 f(154)=f( 02 00 02 )=-0.11847696000829D+02
 f(158)=f( 01 20 01 )=-0.63696419838679D+03
 f(161)=f( 01 11 01 )=-0.27452997974719D+03
 f(167)=f( 01 02 01 )= 0.84212462331467D+03
 f(174)=f( 01 00 03 )=-0.52565377323814D+02
 f(175)=f( 00 40 00 )=-0.72806787999036D+02
 f(176)=f( 00 31 00 )=-0.37282112837702D+03
 f(179)=f( 00 22 00 )= 0.69948623624571D+03
 f(184)=f( 00 20 02 )= 0.18516383109418D+04
 f(185)=f( 00 13 00 )=-0.21045575538010D+02
 f(190)=f( 00 11 02 )=-0.77762707818965D+04
 f(195)=f( 00 04 00 )= 0.32523737181351D+03
 f(200)=f( 00 02 02 )=-0.45676194736033D+03
 f(209)=f( 00 00 04 )= 0.46703244054266D+01

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:
horizontal tune =  0.7148060664739763
first order horizontal chromaticity =  2.2120285662994830E-06
second order horizontal chromaticity =  -517.6580207488710
horizontal tune when delta =  9.9999999999999999E-04
 0.7142884106652559

vertical tune =  0.7148438795599776
first order vertical chromaticity =  1.1365906773273283E-06
second order vertical chromaticity =  -569.8755596054544
vertical tune when delta =  9.9999999999999999E-04
 0.7142740051369628

tune separation when delta=  9.9999999999999999E-04
 1.4405528293126579E-05

normalized anharmonicities
 hhn=  -245.3126910141522
 vvn=  -115.7606385786890
 hvn=  -1227.239884296586
lump lu1      constructed and stored.( 1)
lump lu2      constructed and stored.( 2)
lump lu3      constructed and stored.( 3)
circulating thru loop1:
     1    2    3
```

As indicated in *#comments*, the MaryLie run first builds up the transfer map for one superperiod, prints it out, and computes its tunes and chromaticities. This is accomplished by the entries *sixth*, *mapout*, and *chrom* in the *#labor* component. Inspection of the generating polynomial for the nonlinear portion of the superperiod transfer map shows that it has noticeable nonlinear content as one might expect from the presence of numerous sextupoles. Also, inspection of the result of the chromaticity calculation shows that the sextupoles have been set to achieve negligible first-order chromaticities.

The second function of this MaryLie run is to track trajectories through multipole turns around the ring. The first appearance in *#labor* of the command *rays* causes the writing of the initial conditions into the final condition file. The second appearance of *rays*, since it follows the word *sixth*, causes MaryLie to apply the superperiod transfer map to the initial conditions and to write the resulting final conditions in the final condition file.

Subsequently, the word *loop1* and the command *circ* cause MaryLie to circulate 3000 times through the contents of the loop *loop1*, and to write tracking results into the final condition file after each circulation. (Observe that the parameters associated with the definition of *circ* in *#menu* have the values 3000 and 1.)

Reference to the *#lump* component of the MaryLie Master Input File shows that it contains the lumps *lu1*, *lu2*, and *lu3*. A lump is a collection of elements combined together and treated by a single transfer map. For example, as listed in its contents, *lu1* refers to a single transfer map obtained by combining all the maps for all the elements in *lob1*, *tro1*, *dis1*, *arcin*, and *20*cell*. Observe from the output that as MaryLie runs, it announces the construction and storage of lumps by name and assigns them identification numbers for future reference.

Now look at the contents of *loop1* as listed under *#loops*. It contains the lumps *lu1*, *lu2*, and *lu3*. When MaryLie encounters the name of a loop in #labor followed by a circulation command such as *circ*, it applies the transfer maps contained in the loop over and over again in succession. (Look at the MaryLie output, and observe that it announces the name of the loop and the identification numbers of the transfer maps through which circulation is occurring.) In this case, the first map *lu1* is applied to the initial conditions. Next *lu2* is applied to the resulting phase-space data. Then *lu3* is applied to the phase-space data resulting from the previous application of *lu2*. This completes one circulation around the loop. The sequence now repeats. The transfer map *lu1* acts on the results obtained from the previous application of *lu3*, etc., until the prescribed number of circulations (3000 in this case) have been completed.

Examination of the combined contents of *lu1*, *lu2*, and *lu3* shows that it is identical to the contents of the superperiod *sixth*. As a result, each circulation in this case amounts to tracking through a superperiod.

Now comes a subtle but important point. If MaryLie 3.0 were exact to arbitrary order, the result of applying the transfer map for one superperiod and the result of performing one circulation would be exactly the same. However, because MaryLie 3.0 only works to third order, higher order results are lost when a series of transfer maps are combined to produce a net transfer map. By contrast, when successive transfer maps are allowed to act numerically on phase-space data, these so-called *cross terms* are retained. Consequently, the result of using 3 lumps for a superperiod should be more accurate (should retain more high-order

terms) than the result of using a single map. Indeed, the accuracy should become better and better the more and smaller the lumps. Of course, the cost of computation increases accordingly because the time consuming part of a tracking calculation is the numerical evaluation of the action of transfer maps on phase-space data.

In any given case, it is necessary to experiment to find the optimum lump size that minimizes computation time without sacrificing too much accuracy. Shown below are the first few lines of the final condition file resulting from the present MaryLie run. The first two lines are two initial conditions. The next two lines are the result of applying the one superperiod transfer map to these two initial conditions. Lines 5 and 6 are the result of the first circulation through the lumps *lu1*, *lu2*, and *lu3* in *loop1*. The remaining lines are the result of subsequent circulations.

First few lines of final condition file from tracking run

X	P_x	Y	P_y	τ	P_τ
0.30000E-03	0.00000E+00	0.30000E-03	0.00000E+00	0.00000E+00	0.00000E+00
0.10000E-02	0.00000E+00	0.10000E-02	0.00000E+00	0.00000E+00	0.00000E+00
-0.66366E-04	0.29239E-03	-0.65638E-04	0.29269E-03	-0.43071E-06	0.00000E+00
-0.22862E-03	0.96816E-03	-0.22878E-03	0.97688E-03	-0.52978E-05	0.00000E+00
-0.66366E-04	0.29239E-03	-0.65636E-04	0.29269E-03	-0.43599E-06	0.00000E+00
-0.22867E-03	0.96853E-03	-0.22844E-03	0.97723E-03	-0.59345E-05	0.00000E+00
-0.27052E-03	-0.12874E-03	-0.27133E-03	-0.12879E-03	0.83488E-06	0.00000E+00
-0.88631E-03	-0.44215E-03	-0.90325E-03	-0.44732E-03	0.95155E-05	0.00000E+00

Lines 3 and 5, which are the respective results of applying to the initial condition in line 1 the one superperiod transfer map and one circulation, are nearly identical. Lines 4 and 6, which result from applying the one superperiod transfer map and one circulation to the initial condition in line 2, differ in the fourth place. A difference in accuracy is to be expected for these two initial conditions because the initial condition in line 2 has larger betatron amplitudes, and consequently higher order nonlinear effects should then be more important.

Evidently superperiod by superperiod tracking is sufficiently accurate for the smaller betatron amplitude case. Further investigation shows that breaking up a superperiod into 3 lumps gives sufficient accuracy in the larger betatron amplitude case. In actual operation of a Superconducting Super Collider, the betatron amplitudes would be expected to be an order of magnitude smaller than even that of the first initial condition. Thus MaryLie 3.0 superperiod by superperiod tracking of the error-free design lattice gives excellent accuracy for orbits of physical interest.

Figures 2.7.4 and 2.7.5 show phase-space plots resulting from the tracking portion of the MaryLie run. The orbits appear to have long-term stability. Indeed, much longer tracking runs give similar results. It should also be observed that there is considerable braiding and thickening of the phase-space projections. Again, as with the examples of the small proton and small electron rings, this is a nonlinear and presumably harmless effect due to sextupoles.

Figure 2.7.4: Horizontal phase-space projection for trajectories of two particles in a Superconducting Super Collider.

Figure 2.7.5: Vertical phase-space projection for trajectories of two particles in a Superconducting Super Collider.

Chapter 3

Brief Summary of Theoretical Tools

3.1 Introduction

The mathematical formalism of MaryLie is based on the theory of Lie algebras and symplectic mappings. Since these methods are new, and probably unfamiliar, the purpose of this section is to introduce these methods and describe how they are used in beam transport calculations.

This chapter is organized in the following manner:

3.2 Hamiltonian Dynamics 76
3.3 Poisson Brackets 79
3.4 Transfer Maps 80
3.5 The Symplectic Condition 81
3.6 Taylor Expansions 82
3.7 Overview of the Lie Algebraic Approach 82
3.8 Lie Operators and Lie Transformations 82
3.9 The Factorization Theorem 84
3.10 Polynomial Labeling Scheme 85
3.11 Ingredients of a Lie Algebraic Code 87

In the first few subsections we show how the motion of dynamical systems may be described in terms of certain mappings. Then we introduce certain Lie algebraic tools that can be used to represent these mappings. Finally, we show how these methods are applied to beam transport calculations.

3.2 Hamiltonian Dynamics

Consider a particle in an electromagnetic field. We can describe its motion using Hamilton's Equations. Let $(\mathbf{q}, \mathbf{p}) = (q_1, q_2, q_3, p_1, p_2, p_3)$ denote the generalized coordinates and momenta of the particle. Then, for a given set of initial conditions, the particle's subsequent motion is uniquely determined by *Hamilton's Equations*:

$$\begin{aligned} \frac{dq_i}{dt} &= \frac{\partial H}{\partial p_i} \quad (i = 1, 2, 3), \\ \frac{dp_i}{dt} &= -\frac{\partial H}{\partial q_i} \quad (i = 1, 2, 3), \end{aligned} \tag{3.2.1}$$

where $H(\mathbf{q}, \mathbf{p}, t)$ is the Hamiltonian of the system. In cartesian coordinates (x, y, z), for example, the Hamiltonian is given by the expression

$$\begin{aligned} &H(x, y, z, p_x, p_y, p_z; t) = \\ &\quad \sqrt{m^2c^4 + c^2[(p_x - qA_x)^2 + (p_y - qA_y)^2 + (p_z - qA_z)^2]} + q\phi. \end{aligned} \tag{3.2.2}$$

Here $\mathbf{A}$ and ϕ are the vector and scalar potentials, respectively, and m and q are the rest mass and charge of the particle, respectively.

Now consider the passage of a particle through a beam-line element, as shown in figure 3.2.1. Here, the design orbit is along the z-axis of a cartesian coordinate system. When designing an accelerator, it is more important to know when a particle will reach a particular location, and what its transverse displacement and momentum will be there, rather than to know its coordinates and momenta at an arbitrary time. Referring to figure 3.2.1, one would like to know the quantities (x, y, p_x, p_y, t) as functions of z.

In many instances it is possible to choose a coordinate as an independent variable in place of the time. The time, in turn, is treated as a coordinate. Finally, one introduces a momentum p_t conjugate to the time. When this is done, say for the beam-line element of figure 3.2.1, one automatically obtains the quantities (x, y, t, p_x, p_y, p_t) as functions of z.

In particular, one can find a new Hamiltonian, $K(x, y, t, p_x, p_y, p_t; z)$, such that the equations of motion with z as an independent variable have the form

$$\begin{aligned} x' &= \frac{\partial K}{\partial p_x} \quad , \quad p'_x = -\frac{\partial K}{\partial x}, \\ y' &= \frac{\partial K}{\partial p_y} \quad , \quad p'_y = -\frac{\partial K}{\partial y}, \\ t' &= \frac{\partial K}{\partial p_t} \quad , \quad p'_t = -\frac{\partial K}{\partial t}. \end{aligned} \tag{3.2.3}$$

Here, a prime denotes differentiation with respect to z. Proof that such a procedure is possible and the method for finding K may be found in papers listed in Chapter 11. There it is also shown that p_t is the negative of the total energy,

$$p_t = -E. \tag{3.2.4}$$

In most of what follows, we shall, out of convenience, speak in terms of "ordinary" coordinates and momenta that are functions of time. It should be remembered, however, that these results are equally valid (with the appropriate change in notation) when a *coordinate* is the independent variable. Thus, when we speak of "following a particle from some initial

Figure 3.2.1: Passage Through a Beamline Element

time t^{in} to some final time t^{fin} ", the application of MaryLie will actually be in terms of "following a particle from some initial coordinate z^{in} to some final coordinate z^{fin} ". This will be stated explicitly in important results. For now, the reader should keep this at the back of his or her mind, but not worry about it.

3.3 Poisson Brackets

A key tool in the Lie algebraic method is the *Poisson bracket*. In order to describe the Poisson bracket, and to establish terminology, it is useful to pause for a few definitions.

> **Definition**: The six dimensional space of vectors $(\mathbf{q}, \mathbf{p})$ is called *phase space*. The variables q_i and $p_i, (i = 1, 2, 3)$ are called *phase-space variables*. A *dynamical variable*, $f(\mathbf{q}, \mathbf{p})$, is any smooth function of the phase space variables. (Note that H, q_i and p_i are all dynamical variables.) Since the only functions we shall be concerned with are also dynamical variables, we shall use "function" and "dynamical variable" interchangeably.

The Poisson bracket arises naturally when we consider the time rate of change of a dynamical variable f along a trajectory. By the chain rule, we have the relation

$$(\frac{d}{dt})f(\mathbf{q}, \mathbf{p}) = \sum_{i=1}^{3} \frac{\partial f}{\partial q_i}\dot{q}_i + \frac{\partial f}{\partial p_i}\dot{p}_i. \tag{3.3.1}$$

Using Hamilton's equations (3.2.1), this becomes

$$\frac{df}{dt} = \sum_{i=1}^{3} \frac{\partial f}{\partial q_i}\frac{\partial H}{\partial p_i} - \frac{\partial f}{\partial p_i}\frac{\partial H}{\partial q_i}. \tag{3.3.2}$$

Now introduce the Poisson bracket $[f, g]$ of any two functions f and g. It is defined by the rule

$$[f, g] \stackrel{\text{def}}{=} \sum_{i=1}^{3} \frac{\partial f}{\partial q_i}\frac{\partial g}{\partial p_i} - \frac{\partial f}{\partial p_i}\frac{\partial g}{\partial q_i}. \tag{3.3.3}$$

Then (3.3.2) can be written in the more compact form

$$\frac{df}{dt} = [f, H]. \tag{3.3.4}$$

Note that Hamilton's equations (3.2.1) can be written as a special case of (3.3.4),

$$\frac{dq_i}{dt} = [q_i, H] \tag{3.3.5}$$

$$\frac{dp_i}{dt} = [p_i, H]. \tag{3.3.6}$$

3.4 Transfer Maps

For convenience of discussion, it is useful to employ a notation that treats coordinates and momenta on an equal footing. To this end, let $\mathbf{z}$ denote the vector with six entries given by the rule

$$\mathbf{z} = (\mathbf{q}, \mathbf{p}) = (q_1, q_2, q_3, p_1, p_2, p_3). \tag{3.4.1}$$

(Note that here, and generally from now on, the symbol z has a different meaning than in section 3.2.) Hamilton's equations can now be written in the compact form

$$\frac{d\mathbf{z}}{dt} = -[H, \mathbf{z}]. \tag{3.4.2}$$

Next, consider the 7-dimensional space consisting of $\mathbf{z}$ and t. We call this *augmented phase space*. Suppose a particle has initial phase-space coordinates $\mathbf{z}^{\text{in}}$ at some initial time t^{in}. This state of affairs is represented by the point $(\mathbf{z}^{\text{in}}, t^{\text{in}})$ in augmented phase space. As the particle moves (in accord with Hamilton's equations), the vector $(\mathbf{z}, t)$ will trace out a trajectory in the augmented phase space. See figure 3.4.1.

Figure 3.4.1: Augmented Phase Space

Now let the time t^{in} be fixed. By specifying initial conditions $\mathbf{z}^{\text{in}}$, we also specify a unique trajectory since every point in augmented phase space has exactly one trajectory passing through it. We can follow this trajectory to some final time, t^{fin} . We can also do this for every other trajectory starting at t^{in} . The set of all trajectories so obtained is called the *Hamiltonian flow* (between t^{in} and t^{fin}).

We can, of course, also reverse this procedure. Any point ($\mathbf{z}^{\text{fin}}$, t^{fin}) may be followed backward in time to some point ($\mathbf{z}^{\text{in}}$, t^{in}). Since the correspondence between $\mathbf{z}^{\text{in}}$ and $\mathbf{z}^{\text{fin}}$ is one to one, we may regard them as being related by some invertible (generally nonlinear) map $\mathcal{M}$. We write, symbolically,

$$\mathbf{z}^{\text{fin}} = \mathcal{M}\mathbf{z}^{\text{in}}. \tag{3.4.3}$$

We say that $\mathcal{M}$ is generated by "following" the Hamiltonian flow from t^{in} to t^{fin}. Since $\mathcal{M}$ relates initial and final conditions, it will often be referred to as a *transfer map*. In our applications, we shall utilize transfer maps corresponding to various beamline elements. Such mappings relate coordinates and momenta at the exit face of an element to those at the entrance face.

3.5 The Symplectic Condition

The fact that $\mathcal{M}$ arises from following a Hamiltonian flow imposes strong restrictions on its nature. Indeed, $\mathcal{M}$ belongs to a class of mappings known as *symplectic* mappings. Consider the Jacobian matrix M that characterizes the *local linear* behavior of $\mathcal{M}$. It is defined by the equations

$$M_{ab}(z^{\text{in}}) = \frac{\partial z_a^{\text{fin}}}{\partial z_b^{\text{in}}}. \tag{3.5.1}$$

That is, M describes how small changes in the initial conditions produce small changes in the final conditions. Note that in general M depends on the trajectory in question, and thus we write $M(z^{\text{in}})$. Next, define a matrix J, called the *Poisson* matrix, by the equation

$$J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}. \tag{3.5.2}$$

Here each entry in J is a 3×3 block, and I denotes the 3×3 identity matrix. Then it can be shown that M obeys the relation

$$M^T(z^{\text{in}})JM(z^{\text{in}}) = J \text{ for all } z^{\text{in}}. \tag{3.5.3}$$

Here M^T denotes the transpose of M.

Equation (3.5.3) is the condition that M be a *symplectic* matrix. Correspondingly, a map $\mathcal{M}$ whose Jacobian matrix M is symplectic is said to be a *symplectic map*.

Note that the left hand side of (3.5.3) should generally depend on z^{in} since M depends on z^{in}. By contrast, the right hand side of (3.5.3) is independent of z^{in}. Consequently, $\mathcal{M}$ is strongly restricted by the symplectic condition.

A proof of the symplectic condition and a discussion of some of its consequences may be found in some of the papers listed in Chapter 11. There it is also shown that a symplectic map is a canonical transformation, and vice versa. It is sufficient for present purposes to say that a major reason for using Lie algebraic methods is to exploit the symplectic condition.

Finally, we observe that for purposes of Accelerator Physics it is more convenient to employ the phase-space coordinate ordering (6.18.1) rather than (4.1). In this case the matrix J takes the form (6.18.2).

3.6 Taylor Expansions

For most beam line elements, there is a well defined reference or "design" trajectory. For example, the design trajectory for a quadrupole magnet is the trajectory running along its axis. Suppose we choose our phase-space coordinate systems at t^{in} and t^{fin} in such a way that the design trajectory has $\mathbf{z}^{\text{in}} = 0$ and $\mathbf{z}^{\text{fin}} = 0$. Then, for orbits near the design orbit, $\mathbf{z}^{\text{in}}$ and $\mathbf{z}^{\text{fin}}$ are small. In this case we may expand the relation between $\mathbf{z}^{\text{fin}}$ and $\mathbf{z}^{\text{in}}$, given symbolically by (3.4.3), in a more concrete series representation. We write

$$z_a^{\text{fin}} = \sum_b R_{ab} z_b^{\text{in}} + \sum_{b,c} T_{abc} z_b^{\text{in}} z_c^{\text{in}} + \sum_{b,c,d} U_{abcd} z_b^{\text{in}} z_c^{\text{in}} z_d^{\text{in}} + \cdots . \tag{3.6.1}$$

Here R is the usual transfer matrix of linear matrix theory, and T is the second order transfer matrix, as calculated in the commonly used code TRANSPORT. The quantities U describe third order corrections, and are not usually treated by present codes. Third order contributions are, however, handled routinely by MaryLie 3.0.

Because $\mathcal{M}$ is a symplectic map, the various quantities $R, T, U, \ldots$ are not all independent. Instead, they are interrelated as a result of (3.5.3) by rather complicated linear and nonlinear equations. Moreover, it can be shown that in general the series (3.6.1) cannot be truncated without violating the symplectic condition. Consequently, the use of truncated Taylor series does not lead to an approximation scheme that is consistent with the symplectic condition. For these reasons, (3.6.1) is usually not the best way to parameterize a symplectic transfer map.

3.7 Overview of the Lie Algebraic Approach

We have seen that the motion of a charged particle through a beam line element can be described by a symplectic transfer map $\mathcal{M}$. The Lie algebraic approach provides tools for representing such maps, and for combining symplectic maps for individual elements to find grand maps for collections of beam line elements.

First, we will show that given any dynamical variable f, we can use the Poisson bracket to obtain an associated operator, which we shall denote by the symbols $:f:$. These operators are called *Lie operators*. (We will also refer to f as a *Lie generator*.) Next we will describe how symplectic maps $\mathcal{M}$ can be written down and manipulated in terms of Lie operators. Finally, we will indicate how the symplectic maps for various beam line elements and collections of beam line elements can be described in terms of various polynomials.

3.8 Lie Operators and Lie Transformations

Suppose f is some function of the phase-space variables z. Associated with every such function will be a *Lie operator* denoted by $:f:$. The Lie operator $:f:$ is a *differential* operator defined by the rule

$$:f: \stackrel{\text{def}}{=} \sum_i (\partial f/\partial q_i)(\partial/\partial p_i) - (\partial f/\partial p_i)(\partial/\partial q_i). \tag{3.8.1}$$

In particular, if $:f:$ acts on any phase-space function g, one finds the result

$$:f: g = \sum_i (\partial f/\partial q_i)(\partial g/\partial p_i) - (\partial f/\partial p_i)(\partial g/\partial q_i) = [f, g]. \tag{3.8.2}$$

Thus, one may heuristically view a Lie operator as a Poisson bracket waiting to happen.

Now that Lie operators have been defined, it is a simple matter to define powers of Lie operators. Powers are defined in terms of multiple Poisson brackets:

$$\begin{aligned} :f:^0 g &= g, \\ :f:^1 g &= [f, g], \\ :f:^2 g &= :f: :f: g = [f, [f, g]], \\ :f:^3 g &= [f, [f, [f, g]]], \text{etc.} \end{aligned} \tag{3.8.3}$$

Next we define the *exponential* of a Lie operator $:f:$ by writing the series

$$e^{:f:} = \exp(: f :) \stackrel{\text{def}}{=} \sum_{m=0}^{\infty} \frac{:f:^m}{m!}. \tag{3.8.4}$$

The quantity $e^{:f:}$ or $\exp(: f :)$ will be called the *Lie transformation* associated with f. We shall be concerned primarily with the operation of $e^{:f:}$ on each component z_a. In accord with the definitions (3.8.3) and (3.8.4), we have the relation

$$e^{:f:} z_a = z_a + [f, z_a] + \frac{[f, [f, z_a]]}{2!} + \cdots. \tag{3.8.5}$$

Lie operators and Lie transformations have a number of important properties that are described in detail in some of the papers listed in Chapter 11. One basic property worth noting here is that, in general, Lie operators do not commute,

$$:f: :g: \neq :g: :f: . \tag{3.8.6}$$

Indeed, one can prove the relation

$$:f: :g: - :g: :f: = :[f, g]: . \tag{3.8.7}$$

Consequently, the commutator of two Lie operators can be computed in terms of Poisson brackets.

As a result of the noncommutativity of Lie operators, Lie transformations also do not commute in general; nor do their exponents generally add:

$$\begin{aligned} e^{:f:} e^{:g:} &\neq e^{:g:} e^{:f:}, \\ e^{:f:} e^{:g:} &\neq e^{(:f: + :g:)}. \end{aligned} \tag{3.8.8}$$

3.9 Factorization Theorem

The importance of Lie operators and Lie transformations for our present purposes is a consequence of the *factorization theorem*. This theorem states that symplectic maps can be written in terms of products of Lie transformations, and these Lie transformations are of a very simple kind.

Suppose $\mathcal{M}$ is a symplectic map of the form (3.6.1). Then $\mathcal{M}$ can be represented as an infinite product of Lie transformations,

$$\mathcal{M} = e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdots . \tag{3.9.1}$$

Here each function f_n is a *homogeneous polynomial* of degree n in the phase-space variables $\mathbf{z}^{\text{in}}$.

There are a number of important features of the factored product representation. Those of particular interest for our present concerns are listed below. More detailed discussion may be found in some of the papers listed in Chapter 11.

- Any set of homogeneous polynomials f_n, when employed in (3.9.1), will produce a symplectic map. Thus, the restrictions imposed on $\mathcal{M}$ by the symplectic condition are automatically taken care of by the use of Lie transformations.

- As a corollary of the previous statement, it follows that the infinite product (3.9.1) can be truncated at any stage, and the result is still a symplectic map.

- Suppose we compare the representations (3.6.1) and (3.9.1). Then, as will be shown shortly, one finds that R is completely specified in terms of f_2. Also, T is completely specified in terms of f_3. Finally, U is completely specified in terms of f_3 and f_4. Thus if one is content with results that are correct through third order, as explicitly displayed in (3.6.1), then one only need work with polynomials f_2, f_3, and f_4 in (3.9.1). A count shows that there are 203 (linearly independent) such polynomials in six variables. By contrast, if we count the number of parameters involved in R, T, and U (when the symplectic condition is overlooked), we find that this number is much larger. Thus, the use of Lie algebraic methods requires considerably less computer storage.

  In summary, the factor $\exp(: f_2 :)$ describes the *linear* part of $\mathcal{M}$, and the remaining factors $\exp(: f_3 :) \exp(: f_4 :) \cdots$ describe the *nonlinear* part

- As may be inferred from the previous discussion, each f_n has a direct physical interpretation. For example, second order effects due to sextupole magnets are described by f_3. Third order effects due to octupole magnets and iterated sextupoles are described by f_4. Were one to include fourth and fifth order effects, as is envisioned for example for MaryLie 5.0, then one only need adjoin the homogeneous polynomials f_5 and f_6.

As an illustration for the reader of some aspects of Lie algebraic calculations, we will verify the statements made in the third item above. Consider the evaluation of

$$e^{:f_2:} z_a \quad = \quad z_a + :f_2: z_a + \frac{:f_2:^2 z_a}{2!} + \cdots$$

$$= \quad z_a + [f_2, z_a] + \frac{[f_2, [f_2, z_a]]}{2!} + \tag{3.9.2}$$

We know that, by definition, f_2 is of degree 2. What are the degrees of the terms $[f_2, z_a]$, $[f_2, [f_2, z_a]]$, etc.? Inspection of the Poisson bracket operation (3.3.3) shows that it contains two differentiations. Consequently, $[f_2, z_a]$ must be of degree one. Since $[f_2, z_a]$ is of degree one, it follows that $[f_2, [f_2, z_a]]$ must also be of degree one, etc. We conclude that all the terms on the right hand side of (3.9.2) are of degree one. Therefore (3.9.2) is a linear transformation, and there exist coefficients R_{ab} such that

$$e^{:f_2:} z_a^{\text{in}} = \sum_b R_{ab} z_b^{\text{in}}. \tag{3.9.3}$$

Next consider the evaluation of

$$\begin{aligned} e^{:f_n:} z_a^{\text{in}} &= z_a^{\text{in}} + :f_n: z_a^{\text{in}} + \frac{:f_n:^2}{2!} z_a^{\text{in}} + \cdots \\ &= z_a^{\text{in}} + [f_n, z_a^{\text{in}}] + \frac{[f_n, [f_n, z_a^{\text{in}}]]}{2!} + \cdots, \end{aligned} \tag{3.9.4}$$

where $n \geq 3$. Suppose that g_m and g_n are two homogeneous polynomials of degrees m and n respectively. Then inspection of (3.3.3) shows that $[g_m, g_n]$ is a homogeneous polynomial of degree $m+n-2$. It follows that the terms $[f_n, z_a], [f_n, [f_n, z_a]], \ldots$ are of degrees $n-1, 2n-3$, etc. We see that, provided $n \geq 3$, the terms on the right hand side of (3.9.4) are of successively higher degree. Comparison with (3.6.1) shows that only $[f_3, z_a]$ contributes to the second order terms, and only $[f_3, [f_3, z_a]]$ and $[f_4, z_a]$ contribute to the third order terms. We also conclude that polynomials f_n of degree five and higher only contribute to terms in (3.6.1) of degree four and higher.

3.10 Polynomial Labeling Scheme

As indicated in the previous section, symplectic maps $\mathcal{M}$ that send the origin into itself may be specified through third order in terms of polynomials f_2, f_3, and f_4. In 6 variables, there are a fixed number of different monomials of any given degree. The numbers of such monomials for each degree through degree 9 are listed below:

Degree n	Number of Monomials N(n) of Degree n in 6 variables	Running Total
1	6	6
2	21	27
3	56	83
4	126	209
5	252	461
6	462	923
7	792	1715
8	1287	3002
9	2002	5004

These monomials may be listed sequentially, and consequently each monomial may be labeled by a number called its *index*. One way of doing this is illustrated below for the first few monomials. The variables used are those of section 4.1.2.

		Exponents		
Index	Monomial	X P_x	Y P_y	τ P_τ
1	X	1 0	0 0	0 0
2	P_x	0 1	0 0	0 0
3	Y	0 0	1 0	0 0
4	P_y	0 0	0 1	0 0
5	τ	0 0	0 0	1 0
6	P_τ	0 0	0 0	0 1
7	X^2	2 0	0 0	0 0
8	XP_x	1 1	0 0	0 0
$\vdots$	$\vdots$		$\vdots$	
26	τP_τ	0 0	0 0	1 1
27	P_τ^2	0 0	0 0	0 2
28	X^3	3 0	0 0	0 0
29	X^2P_x	2 1	0 0	0 0
$\vdots$	$\vdots$		$\vdots$	
82	τP_τ^2	0 0	0 0	1 2
83	P_τ^3	0 0	0 0	0 3
84	X^4	4 0	0 0	0 0
85	X^3P_x	3 1	0 0	0 0
$\vdots$	$\vdots$		$\vdots$	
208	τP_τ^3	0 0	0 0	1 3
209	P_τ^4	0 0	0 0	0 4

This labeling is given in detail in Chapter 14 for all monomials through degree 4.

The indexing scheme just described is used in MaryLie 3.0 to label the monomial coefficients occurring in f_3 and f_4 . Thus, for example, when MaryLie 3.0 is asked to print the nonlinear part of the transfer map as in section 2.2, the quantities $f(i)$ are the nonzero coefficients of the monomials with indices i appearing in f_3 and f_4 . Note that for the convenience of the reader, the corresponding exponents are also displayed.

The indices 1 through 6 corresponding to f_1 and the indices 7 through 27 corresponding to f_2 are not used by MaryLie 3.0 for the specification of transfer maps. (They are used, however, for other purposes. They will also be used by MaryLie 3.1 for the specification of transfer maps that do not send the origin into itself.) In view of (3.9.2) and (3.9.3), f_2 could be used to specify the matrix part R of a transfer map. However, according to (3.9.2), the computation of R from f_2 requires the summation of an infinite series. It is therefore more convenient for most purposes to store and manipulate the matrices R directly.

3.11 Ingredients of a Lie Algebraic Code

The Lie algebraic concepts just sketched, and their consequences, can be used to make a Lie algebraic charged particle beam transport code. The special ingredients of such a code are described briefly below.

3.11.1 Element Library

We have seen that the effect of a beamline element is described by a symplectic map, and that a symplectic map (through terms of degree three) is described in turn by certain polynomials f_2, f_3, and f_4. The first ingredient of a Lie algebraic code, therefore, is a library that contains the various polynomials for the various kinds of beamline elements.

MaryLie has such a library, by way of various subroutines, that includes the most common beamline elements. Given a common beamline element, say a sextupole magnet of a certain length and strength, there is an associated subroutine that supplies the appropriate f_2, f_3, and f_4. [Actually, as mentioned earlier, the subroutine provides the matrix R instead of f_2 because the series (3.9.2) generally contains an infinite number of terms and is therefore computationally awkward.] The way in which this library was prepared is described in some of the papers listed in Chapter 11.

Exhibit 3.11.1 below shows, for example, the maps for a drift and a sextupole, each of length .5 meters. As expected, they have the same linear (matrix) parts, but their nonlinear generators (f_3 and f_4) are different. Note also that even a simple drift has nonlinear generators. They arise from the use of canonical coordinates and the choice of z as the independent variable. See section 4.1.2.

```
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 3.11.1.
 This MaryLie run produces and displays the transfer maps for a drift
 and a sextupole.  Both have a length of .5 meters, and the sextupole
 has a strength of 2 Tesla/m^2.  The beam parameters are those for
 50 MeV protons.

#beam
  1.03528744085195
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
```

```
  clear    iden
  dr       drft
   0.500000000000000
  sex      sext
   0.500000000000000       2.00000000000000
  end      end
 #labor
     1*fileout
     1*dr
     1*mapout
     1*clear
     1*sex
     1*mapout
     1*end

**************************
* Transfer Map for Drift *
**************************

 matrix for map is :

  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.56964E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

 nonzero elements in generating polynomial are :

  f( 53)=f( 02 00 01 )=-0.79605606944381
  f( 76)=f( 00 02 01 )=-0.79605606944381
  f( 83)=f( 00 00 03 )= -7.2753826985187
  f(140)=f( 04 00 00 )=-6.25000000000000E-02
  f(149)=f( 02 02 00 )=-0.12500000000000
  f(154)=f( 02 00 02 )= -3.6772315941900
  f(195)=f( 00 04 00 )=-6.25000000000000E-02
  f(200)=f( 00 02 02 )= -3.6772315941900
  f(209)=f( 00 00 04 )= -28.386857507713

******************************
* Transfer Map for Sextupole *
******************************

 matrix for map is :

  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.56964E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=-0.32197177342268
 f( 29)=f( 21 00 00 )= 0.24147883006701
 f( 34)=f( 12 00 00 )=-8.04929433556708E-02
 f( 39)=f( 10 20 00 )= 0.96591532026805
 f( 40)=f( 10 11 00 )=-0.48295766013402
 f( 43)=f( 10 02 00 )= 8.04929433556708E-02
 f( 49)=f( 03 00 00 )= 1.00616179194588E-02
 f( 53)=f( 02 00 01 )=-0.79605606944381
 f( 54)=f( 01 20 00 )=-0.24147883006701
 f( 55)=f( 01 11 00 )= 0.16098588671134
 f( 58)=f( 01 02 00 )=-3.01848537583765E-02
 f( 76)=f( 00 02 01 )=-0.79605606944381
 f( 83)=f( 00 00 03 )= -7.2753826985187
 f( 84)=f( 40 00 00 )= 3.88746835803553E-02
 f( 85)=f( 31 00 00 )=-3.88746835803553E-02
 f( 90)=f( 22 00 00 )= 1.45780063426332E-02
 f( 95)=f( 20 20 00 )= 7.77493671607107E-02
 f( 96)=f( 20 11 00 )=-3.88746835803553E-02
 f( 99)=f( 20 02 00 )= 9.71867089508883E-04
 f(105)=f( 13 00 00 )=-2.42966772377221E-03
 f(109)=f( 12 00 01 )=-0.12815379221136
 f(110)=f( 11 20 00 )=-3.88746835803553E-02
 f(111)=f( 11 11 00 )= 2.72122785062487E-02
 f(114)=f( 11 02 00 )=-2.42966772377221E-03
 f(132)=f( 10 02 01 )= 0.12815379221136
 f(140)=f( 04 00 00 )=-6.23264523054448E-02
 f(144)=f( 03 00 01 )= 3.20384480528393E-02
 f(145)=f( 02 20 00 )= 9.71867089508883E-04
 f(146)=f( 02 11 00 )=-2.42966772377221E-03
 f(149)=f( 02 02 00 )=-0.12465290461089
 f(154)=f( 02 00 02 )= -3.6772315941900
 f(161)=f( 01 11 01 )= 0.25630758442271
 f(167)=f( 01 02 01 )=-9.61153441585178E-02
 f(175)=f( 00 40 00 )= 3.88746835803553E-02
 f(176)=f( 00 31 00 )=-3.88746835803553E-02
 f(179)=f( 00 22 00 )= 1.45780063426332E-02
 f(185)=f( 00 13 00 )=-2.42966772377221E-03
 f(195)=f( 00 04 00 )=-6.23264523054448E-02
 f(200)=f( 00 02 02 )= -3.6772315941900
 f(209)=f( 00 00 04 )= -28.386857507713

end of MARYLIE run
```

3.11.2 Concatenation Subroutines

Second, there must be a procedure for combining two symplectic maps to find a net resultant symplectic map. Suppose $\mathcal{M}_f$ and $\mathcal{M}_g$ are two symplectic maps with the representations

$$\mathcal{M}_f \;=\; e^{:f_2:}e^{:f_3:}e^{:f_4:}, \tag{3.11.1}$$

$$\mathcal{M}_g = e^{:g_2:}e^{:g_3:}e^{:g_4:}. \tag{3.11.2}$$

Then one needs a method for finding h_2, h_3, and h_4 such that

$$\mathcal{M}_f\mathcal{M}_g = \mathcal{M}_h = e^{:h_2:}e^{:h_3:}e^{:h_4:}. \tag{3.11.3}$$

There are standard Lie algebraic tools for this purpose, and they are embodied in concatenation subroutines. A description of these tools may be found in some of the papers listed in Chapter 11. Path length is also accumulated in the concatenation process.

In actual operation, MaryLie goes through a collection of beam elements sequentially. Each time it comes to an element, it finds the map (matrix and polynomials) for that element in its library, and then combines that map with the net map corresponding to all the preceding elements. See section 5.11.

3.11.3 Ray Trace Subroutines

Third, there must be a procedure for using the symplectic map (up to some point in a beam line) to perform a ray trace. That is, we may wish to carry out the operation (3.4.3) for a variety of initial conditions. Suppose $\mathcal{M}$ has the factorization

$$\mathcal{M} = e^{:f_2:}e^{:f_3:}e^{:f_4:}. \tag{3.11.4}$$

Then the factor $e^{:f_2:}$ is evaluated exactly using an appropriate transfer matrix R. However, the handling of the factors $e^{:f_3:}e^{:f_4:}$ is somewhat more complicated since their complete evaluation would appear to require the summation of an infinite series. Presently, there are essentially two available alternatives.

First, one may make only an approximate evaluation using the truncated series

$$e^{:f_3:}e^{:f_4:} \simeq (1+ :f_3: +\frac{:f_3:^2}{2!} + \frac{:f_3:^3}{3!})(1+ :f_4:) \tag{3.11.5}$$

$$\simeq 1+ :f_3: +(\frac{:f_3:^2}{2!}+ : f_4 :) + (\frac{:f_3:^3}{3!}+ :f_3: :f_4:). \tag{3.11.6}$$

Because the series has been broken off, the net result is no longer a true symplectic map. Observe that the last set of terms on the right hand side of (3.11.6) produce, when acting on z^{in} , terms of degree 4. As indicated earlier, we only expect the series to be correct through terms of degree 3. This is because terms such as $:f_5: z^{\text{in}}$, which are also of degree 4, have not been included. However, (3.11.6) is easily computed using the machinery of MaryLie 3.0, and has the virtue of being symplectic through terms of degree 4.

As variants on (3.11.6), it is possible to truncate the expansion at earlier stages. For example, if only the first 3 terms are kept, the result is correct through terms of degree 3 and symplectic through terms of degree 3. If only the first 2 terms are kept, the result is correct through terms of degree 2 and symplectic through terms of degree 2.

Alternatively, one may carry out a completely symplectic ray trace by means of a transformation function of mixed variables, $F(q, P)$. This transformation function, which is expressible in terms of f_3 and f_4, is used to obtain new variables (Q, P) from the old variables

(q, p). [Here (Q, P) are the final conditions of the ray trace; (q, p) are the initial conditions.] The equations for P are implicit, but can be solved to machine precision using a rapidly converging iterative procedure. Then Q can be calculated explicitly from q and P. See section 7.2.

This transformation function procedure is equivalent to introducing higher polynomials g_5,g_6,... and exactly evaluating the image of z^{in} under the mapping

$$\mathcal{M} = e^{:f_2:} e^{:f_3:} e^{:f_4:} e^{:g_5:} \ldots . \tag{3.11.7}$$

The higher order polynomials are not physical, being various combinations of f_2, f_3, and f_4. However, since MaryLie is only accurate through third order (use of Lie generators through f_4), the above mapping is as "correct" as if we had set $g_5 = g_6 = \cdots = 0$, and has the virtue that it can be evaluated easily and exactly to machine precision.

Thus, this "symplectic" ray trace is correct through third order, and is symplectic to all orders. However, because of the iterations involved, it requires somewhat more machine time than the various truncated Lie series ray traces.

3.11.4 Analysis Subroutines

A fourth ingredient in a Lie algebraic code is a collection of computational tools for evaluating various properties of transfer maps and collections of transfer maps. Many of these tools are still under development. Some of the quantities or properties that MaryLie 3.0 can compute are listed below:

- First, second, and third order dispersion. (That is, the off-momentum closed orbit through third order in momentum deviation.)
- First, second, and third order "phase-slip factors" (specifically, the time of flight on the off-momentum closed orbit through third order in momentum deviation).
- Linear and nonlinear properties of the transfer map about the off-momentum closed orbit.
- Tunes and first and second order chromaticities and anharmonicities.
- Linear lattice functions and their dependence on momentum deviation through second order.
- Resonant and nonresonant normal forms $\mathcal{N}$ for a transfer map $\mathcal{M}$, and the normal form transformations $\mathcal{A}$. (These quantities are computed through third order.) A normal form $\mathcal{N}$ gives information about chromaticities and the dependence of tunes on betatron amplitudes. The normal form transformation $\mathcal{A}$ is useful in analyzing tracking studies, finding higher order invariants, and computing invariant manifolds.
- Nonlinear lattice functions through third order.
- Nonlinear resonance data.

- Moment transport and emittance analysis.

More information about these computational tools is given in Chapters 8 and 10 and some of the papers listed in Chapter 11.

3.11.5 Bookkeeping Subroutines

A fifth ingredient of a Lie algebraic code is a set of routines for storing and manipulating large polynomials. This task is facilitated in MaryLie by means of a suitable hash function and look-up tables that make optimum use of memory space. Their exact nature need not concern most MaryLie users.

3.11.6 Procedures and Fitting and Optimization Subroutines

Finally, the last ingredient of a Lie algebraic code, as with many other codes, is a set of routines for carrying out various computational procedures such as fitting and optimization. MaryLie has an extensive collection of such routines. More information is given in Chapters 9 and 10.

Chapter 4

Input/Output Preparation and Formats

4.1 General Considerations

4.1.1 Units and Constants

MaryLie uses MKSA units, unless noted otherwise. On input, beam-line element parameters should be specified as follows:

angle	:	degrees
length	:	meters
frequency	:	Hertz
magnetic induction	:	Tesla
voltage	:	Volts

Fundamental constants used in MaryLie and PREP are taken from the references listed in Chapter 11, and are shown below:

MaryLie and PREP Constants

π = 4.d0 $*$ atan(1.d0)
c = 2.99 792 458 x 10^8 m/s
e = 1.60 217 733 $\times$ 10^{-19} coulomb

m_e = 0.510 999 06 MeV/c^2 m_p = 938.272 31 MeV/c^2 m_d = 1875.613 39 MeV/c^2
$m_{h-} = m_p + 2m_e - m_{bind}$ = 939.294 29 MeV/c^2 [m_{bind} = (13.59811 + .7542) eV/c^2]
m_μ = 105.658 387 MeV/c^2 m_π = 139.567 5 MeV/c^2

4.1.2 Scaling and Dimensionless Variables for Phase Space

Let x, y, z be the Cartesian coordinates of a particle having rest mass m, and let p_x, p_y, p_z be the corresponding relativistic mechanical momenta. They are defined by the relations

$$p_x = \gamma m v_x \ , \ p_y = \gamma m v_y \ , \ p_z = \gamma m v_z, \tag{4.1.1}$$

where γ is the usual relativistic factor,

$$\gamma = [1 - (v^2/c^2)]^{-1/2}. \tag{4.1.2}$$

The magnitude p of the mechanical momentum is defined by the relation

$$p = (p_x^2 + p_y^2 + p_z^2)^{1/2} = \gamma m v. \tag{4.1.3}$$

Suppose the design momentum is denoted by the quantity p^0. (See Section 5.6.3). Define a momentum deviation variable δ by the relation

$$p = (1 + \delta)p^0. \tag{4.1.4}$$

Also, let p_t be the *negative* of the total energy. It is defined by the relation

$$p_t = -(m^2c^4 + p^2c^2)^{1/2} = -\gamma m c^2. \tag{4.1.5}$$

Corresponding to the design momentum p^0, there is a design value p_t^0 for p_t given by the relation

$$p_t^0 = -[m^2c^4 + (p^0c)^2]^{1/2}. \tag{4.1.6}$$

As explained in section 3.6, it is useful to work with deviation variables. For the transverse coordinates, the quantities x, y, p_x, p_y can themselves be viewed as deviation variables since they all have the value zero on the design orbit. The case of longitudinal deviation variables is more complicated. For one of them we select $\tilde{p}_t$, the deviation in p_t. It is defined by the relation

$$\tilde{p}_t = p_t - p_t^0. \tag{4.1.7}$$

For the second we chose $\tilde{t}$, the difference in the time of flight between an actual particle and a particle on the design orbit with the design momentum (energy). It can be shown that the variables $(x, p_x, y, p_y, \tilde{t}, \tilde{p}_t)$ form a canonical set.

Rather than working with the phase-space variables $(x, p_x, y, p_y, \tilde{t}, \tilde{p}_t)$, MaryLie uses dimensionless variables given by

$$\begin{aligned} X &= x/l \quad , \quad P_x = p_x/p^0, \\ Y &= y/l \quad , \quad P_y = p_y/p^0, \\ \tau &= c\tilde{t}/l \quad , \quad P_\tau = \tilde{p}_t/(p^0c). \end{aligned} \tag{4.1.8}$$

The scale length l is chosen by the user and given as input. See section 5.6.3. The differential transit time $\tilde{t}$ is scaled by l/c, the light travel time across the scale length. Momenta are scaled by the design momentum, p^0, and $\tilde{p}_t$ is scaled by p^0c. The variables $(X, P_x, Y, P_y, \tau, P_\tau)$ also form a canonical set.

As might be imagined, there is a relation between P_τ and δ. One finds from (4.1.4), (4.1.6), (4.1.7), and (4.1.8) the relations

$$\begin{aligned} P_\tau &= -(1/\beta_0)\{[1 + (2\delta + \delta^2)\beta_0^2]^{1/2} - 1\} \\ &= -\beta_0\delta + (\delta^2/2)(\beta_0^3 - \beta_0) - (\delta^3/2)(\beta_0^5 - \beta_0^3) + \cdots, \end{aligned} \tag{4.1.9}$$

$$\delta = [1 - 2P_\tau/\beta_0 + P_\tau^2]^{1/2} - 1 = -P_\tau/\beta_0 + (P_\tau^2/2)(1 - \beta_0^{-2}) + \cdots. \tag{4.1.10}$$

Here β_0 is the usual relativistic factor evaluated on the design orbit,

$$\beta_0 = v^0/c = -p^0 c/p_t^0. \tag{4.1.11}$$

Many codes provide expansions of various quantities as power series in δ. These quantities include closed orbit location (dispersion), tunes, and Twiss parameters. To facilitate easy comparison, MaryLie also provides such expansions. However, from a Lie algebraic perspective, expansions in P_τ are more natural. One might argue that such expansions are also of greater physical interest since it is energy deviations that are produced by RF cavities. Moreover, beams are often characterized by their energy spread. For these reasons, MaryLie also provides expansions of the usual quantities as a power series in P_τ. Since it is natural to do so in a Lie algebraic code, MaryLie also provides expansions of several additional quantities as power series in P_τ. Of course, given any expansion in P_τ, one can always find a related expansion in δ, and vice versa, by use of the relations (4.1.9) and (4.1.10). Indeed, the MaryLie δ expansions are found in this way.

Consider particle motion in a drift space. The Hamiltonian K for such motion with z as the independent variable and the quantities (x, p_x, y, p_y, t, p_t) as dependent variables is given by the relation

$$K = -(p_t^2/c^2 - m^2c^2 - p_x^2 - p_y^2)^{1/2}. \tag{4.1.12}$$

Correspondingly, the Hamiltonian H for such motion with z as the independent variable and the deviation variables (4.1.2.8) as dependent variables is given by the relation

$$H = -(1/\ell)\{[1 - (2P_\tau/\beta_0) + P_\tau^2 - P_x^2 - P_y^2]^{1/2} + (P_\tau/\beta_0) - (1/\beta_0^2)\}. \tag{4.1.13}$$

By using the Hamiltonian H we find the result

$$\begin{aligned} x' &= dx/dz = \ell\ dX/dz = \ell\ \partial H/\partial P_x \\ &= P_x[1 - (2P_\tau/\beta_0) + P_\tau^2 - P_x^2 - P_y^2]^{-1/2} \\ &= P_x + P_x P_\tau/\beta_0 + (1/2)P_x[P_\tau^2(3\beta_0^{-2} - 1) + P_x^2 + P_y^2] + \cdots. \end{aligned} \tag{4.1.14}$$

We see that x' (the usual TRANSPORT type variable) agrees with P_x in lowest order; but there are second-order chromatic differences, and third- and higher-order geometric and chromatic differences. Also, X and x' are *not* canonically conjugate, $[X, x'] \neq 1$.

4.2 Input and Output Files

Because MaryLie processes and produces a large amount of data, it employs several input and output files (tapes). Most of these files may be chosen at will by the user. However, a few of these files have fixed designations. Such files, and their use, are listed below:

Input Files

Name	Purpose
file 5	Terminal input to MaryLie. Used primarily in

fitting and optimization. See Chapter 9.

file 11 This is the Master Input File. It describes a collection of beam-line elements and commands and the calculations to be performed. See sections 4.3.1, 4.4.1, and Chapter 5.

file 13, l5, etc. These files may be used at will. Generally, it is convenient to use odd numbered files for input and even numbered files for output.

Output Files

Name	Purpose
file 6	This file is sent to the user's terminal or printer. It contains information about the progress made by MaryLie during the course of computation, and results of various calculations requested by the user. See sections 7.2, 7.3, 7.4, and 7.7.
file 12	If MaryLie is requested to provide extensive output, this output may be sent to file 12 rather than file 6. The results of a MaryLie run can then be examined using an Editor. See sections 7.2, 7.3, 7.4, and 7.7. If desired, file 12 can be replaced by some other file. See section 7.30.
file 14, 16, etc.	These files may be used at will. Generally, it is convenient to use even numbered files for output, and odd numbered files for input.

4.3 Preparation of Files

4.3.1 The Program PREP

File 11 is the Master Input File for MaryLie. It may be prepared directly by the user with the aid of a local Editor. However, for a beginner, it may be much more easily prepared with the aid of the interactive program PREP. The use of this program is described in Chapter 5.

4.3.2 Free Format Input

Unless as noted elsewhere, both MaryLie and PREP read all input in "free field" format, for which the following rules apply:

- Numbers may be input anywhere on a line.

- Sequences of numbers are read in from left to right; they must be separated by a comma and/or blanks.

- A sequence of numbers may continue from one line to the next, but an individual number may not.

- Blanks are *not* read in as zeroes.

- Real numbers do not need a decimal point if the number is a whole number.

- Floating point numbers are in double precision.

The following are all valid representations of the real number 23.41:

```
23.41
23.41D0
0.2341D+2
```

Warning: At least with some operating systems, there is a format problem with free format writes on the CRAY when running in double (128 bit) precision. Suppose one wishes to write various numbers on a file, and then reread these numbers for subsequent use in the same or later MaryLie run. (This may be done, for example, in connection with the *rt*, *tmi*, and *tmo* commands. See sections 7.2, 7.5, and 7.6.) The problem is that when the CRAY writes numbers out to files, all significant figures appear as desired, but with an E format rather than a D format. Then, when and if these numbers are read back in, the E format causes them to be treated as single precision. Consequently their double precision accuracy is lost. There is an awkward solution to this problem if these numbers are to be used in a subsequent MaryLie run: one uses an editor to change the E's to D's in such files before the numbers are read back in. There does not seem to be any simple solution to this problem if the numbers are to be read back in during the same MaryLie run.

4.4 Input File Structures

4.4.1 File 11 (Master Input File)

File 11 contains 7 major components. Each component begins with the (sharp or number or pound) symbol # followed by the name of the component. The components are #comment, #beam, #menu, #lines, #lumps, #loops, and #labor. Subsequent lines in a given component describe its contents. Thus, the general structure of file 11 is as shown below:

#comment
⋮
#beam
⋮
#menu
⋮
#lines
⋮
#lumps
⋮
#loops
⋮
#labor
⋮

The #comment, #lines, #lumps, and #loops components are optional. The #beam, #menu, and #labor components are required.

Complete detail about each of these components and its contents is given in Chapter 5. For the present the brief summary below, which states the purpose of each component and the section reference for further information, is sufficient. At this point, the reader is also encouraged to examine or re-examine some of the examples in Chapter 2.

Component	Purpose/Section
#comment	Allows the user to write comments about the system under study and the computations to be made. Section 5.5.
#beam	Specifies magnetic rigidity and relativistic β and γ factors and charge of the beam. Also specifies scale length. Section 5.6.
#menu	Contains a list of beam-line elements, commands, and procedures. Section 5.7.
#lines	Contains a list of names for collections of elements and/or commands and/or procedures drawn from the menu. Section 5.8.
#lumps	Contains a list of items from the menu and/or lines that are to be combined together to form individual maps. Section 5.9.
#loops	Contains information about special tracking or other operations. Section 5.10.
#labor	Specifies the system to be studied, and the actual operations and calculations to be performed. Section 5.11.

4.4.2 Initial Conditions File for Ray Traces or Tracking

Initial conditions are read in prior to ray tracing or tracking. See sections 7.2 and 7.3. As usually configured, MaryLie can store up to 10,000 rays (i.e. sets of initial conditions). The order of the input variables is:

$$(X^{in}, P_x^{\text{in}}, Y^{\text{in}}, P_y^{\text{in}}, \tau^{\text{in}}, P_\tau^{\text{in}})$$

The input variables must be in *dimensionless form* (as described in section 4.1.2), and may be entered in free format as described in section 4.3.2.

Example: .1,0,0,0,0,0
0,.6,0,0,0,0
0,0,.4,.1,0,0

The above specifies initial conditions for 3 rays.

Note that because MaryLie reads the Initial Conditions File in free format, the format of one number per line is acceptable as well as the format described above. See sections 7.2 and 7.3. If desired, final and initial conditions may be written in full precision. Thus it is possible to restart a tracking calculation. See section 4.5.2.

4.4.3 Matrix and Polynomial Input File

The use of this file is explained in sections 7.5 and 7.6 under type codes *tmi* and *tmo*. Each line of the file may be in free format as illustrated below:

$$\left.\begin{array}{lll} I_1, & J_1, & R(I_1,J_1) \\ I_2, & J_2, & R(I_2,J_2) \\ & \vdots & \\ 6, & 6, & R(6,6) \end{array}\right\} \text{nonzero matrix elements}$$

$$\left.\begin{array}{ll} N_1, & P(N_1) \\ N_2, & P(N_2) \\ & \vdots \\ 209, & P(209) \end{array}\right\} \text{nonzero polynomial coefficients}$$

Here the N_i are integers (called MaryLie indices) between 0 and 209. The quantity $P(0)$ is the path length or zeroth order moment. See sections 7.43 and 8.37. The remaining $P(N_i)$ correspond to 1st through 4th order polynomial coefficients. See section 3.10 and Chapter 14.

The above sequence may be repeated as many times as desired. The (6,6) matrix element must always be the last element listed, even if it is zero. Also, the last polynomial entry {209,$P(209)$} must be listed, even if it is zero. An example is shown below:

$$\begin{array}{ll} 1,1,1.346 & \\ 1,2,5.135D-02 & \\ 1,6,-6.084D+01 & \\ 2,1,-18.36 & \\ 2,2,.04246 & \\ 2,6,-.3988 & \\ 3,3,6.203 & \\ 3,4,4.991D-02 & \\ 5,5,1.0 & \\ 6,6,1.0 & \\ 28,-.3916D+03 & (X^3) \\ 29,.8192D-02 & (X^2P_x) \\ 95,-.2660D+03 & (X^2P_xP_\tau) \\ 104,-.6151D+03 & (X^2Y^2) \\ 140,.9753 & (P_x^4) \\ 144,4.361 & (P_x^3P_\tau) \\ 209,0. & (P_\tau^4) \end{array}$$

Note: The quantities shown in parentheses are not actually entered. They are displayed only for the reader's convenience in this example. See section 3.10 and Chapter 14.

4.4.4 Input Data for Random Elements

See section 6.19 for use of files in connection with inputing data for random elements.

4.4.5 Input Data for Procedures and Fitting and Optimization Commands

See sections 9.5 and 9.6.

4.5 Output File Structures

4.5.1 Files 6 and 12

The exact formats of these files are not expected to be of interest to the general reader. They may be inferred either by looking at examples or from the FORTRAN listing of MaryLie.

4.5.2 Final Conditions from a Ray Trace or Tracking

At the user's option, MaryLie writes the results of a ray trace on file 6 (the user's terminal or printer) or file 12. See sections 7.2 and 7.3. It also writes the final conditions of every ray trace on a user assigned final conditions file. (Recall that in a single ray trace MaryLie will trace all the initial rays.) This data may then be used by plotting routines, or various other analysis routines, including the current or subsequent runs of MaryLie itself. The format of the final conditions file is 6(1x,1pe12.5). This gives a 6 column format for the 6 phase-space variables (see section 4.4.2), and each entry has 6 digit precision. If desired, final conditions can also be written out in full precision. See sections 7.2 and 7.3.

4.5.3 Matrix and Polynomial Output File

The use of this file is explained in section 7.6 under type code *tmo*. The output format is the same as the input format for the Matrix and Polynomial Input File described in section 4.4.3. Output is in full precision.

Chapter 5

Format of MaryLie Master Input File and Use of PREP

5.1 General Instructions

As indicated in section 4.3.1, file 11 may be prepared with the aid of the program PREP. The use of PREP is not required, but for the beginner it will ease the labor of initially inputing a large lattice. Slight modifications of an existing lattice are more easily accomplished by directly editing the file (usually file 11) in which it is stored. Once the reader has observed the Master Input File as prepared by PREP, she or he will be able to infer its format, and if desired, will be able to make up such a file directly.

5.1.1 PREP Input

PREP runs interactively, and receives input and communicates with the user through the terminal (files 5 and 6). Also, if desired, PREP can receive information from file 9. See section 5.3. As indicated in section 4.3.2, PREP uses free format input.

5.1.2 PREP Output

The output of PREP is written on file 10. The contents of file 10 must subsequently be copied into file 11 to be used for a MaryLie run, or into file 9 to restart a PREP run.

5.2 Use of Help Feature

For the convenience of the user, PREP contains help information that can be accessed at any time through appropriate help commands.

When PREP is executed, the first thing it replies is

```
**PREP 3.0** Master Input File preparation for MARYLIE 3.0
Copyright 1987 Alex J. Dragt
All rights reserved
```

It then provides opportunity to receive brief instructions about the use of the help feature:

```
Do you want instructions? (y,n) <n> y
```

PREP runs interactively, and help is available at any time. Help may be obtained in any of three ways depending on the circumstances:

- If PREP expects an answer or requests input in the form of a character string, you may type ? or ?help instead of the expected response. After the help session, PREP will return to where it was and re-ask the same question or re-request the same input.

- If PREP is expecting a number, you may get help by responding with a number that equals 1.e+30 or more. Again, after the help session, PREP will return to where it was.

- At various places in a PREP run, as will be evident, you will be able to ask for help directly.

There are several levels in help. To return directly from help, type 'exit'. To backtrack through the various levels, type 'back'. To re-view a current level, type 'see'.

The further operation of the help feature is meant to be self explanatory. The user is urged to try this feature whenever a question arises.

5.3 Input From External File

At the user's option, PREP can receive information from file 9. The structure of file 9 is the same as that of files 10 and 11. This feature makes it possible for a user to begin a PREP run with the results of a previous PREP run, or a user prepared file 11. For an example of the use of file 9, see section 5.14. PREP gives opportunity to use file 9 by asking the following question at the beginning of a run:

```
Read a PREP data set from file 9? (y,n,?) <n>
```

5.4 Modes, Editors, and Related Commands

After offering the options of instructions and the use of a data set at the beginning of a run, PREP goes into *control* mode as indicated by the lines below:

```
 Control mode. Type
cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
```

PREP runs in several modes. These modes enable the user to specify information concerning the comment, beam, menu, lines, lumps, loops, and labor components of file 11. These components have already been described briefly in section 4.4.1. Listed below are the associated modes, their purpose, and the sections describing these modes in more detail.

Mode	Purpose/Section
comment (cmts)	To allow the user to write comments about the system under study and the computations to be made. Section 5.5.
beam	To input the #beam component which describes the magnetic rigidity and relativistic β and γ factors of the beam, and its charge, and to specify the scale length to be used. Section 5.6.
menu	To input the #menu component which contains a list of beam-line elements and commands. Section 5.7.
line	To input the #lines component which contains a list of names for collections of elements and/or commands drawn from the menu. Section 5.8.
lump	To input the #lumps component which contains a list of items from the menu and/or lines that are to be combined together to form individual maps. Section 5.9.
loop	To input the #loops component which contains information about special tracking or other operations. Section 5.10.
labor	To input the #labor component which specifies the system to be studied, and the operations and calculations to be performed. Section 5.11.

In addition to the modes listed above, PREP has a *control* mode. It is activated by typing *ctrl*. Its purpose is to give access to the other modes as well as the special commands *summ*, *file*, *exit*, *abort*, and *?help*. The *?help* command has already been described in section 5.2. The purpose of the other commands, and the sections in which they are described in fuller detail, are listed below.

Command	Purpose/Section
summ	To obtain a listing (summary) of what has been accomplished to date in a PREP session. Section 5.12.
file	To write on file 10 whatever has been specified to date about the comments, beam, menu, lines, lumps, loops, and labor components. Section 5.13.
exit	To cause normal termination of a PREP run. Section 5.14
abort	To cause abnormal termination of a PREP run. Section 5.14.

The use of beam mode is relatively simple, and any mistakes made while in this mode are easily rectified by rerunning the mode. See section 5.6. The use of the comments, menu, lines, lumps, loops, and labor modes is more complex. Each of these modes has a special editor that enables the user to see what has been accomplished, make changes, and return to the control mode. These special editors are described in the same sections as their respective modes.

5.5 Comment (cmts) Mode

The *comment* mode allows the user to write a comments section about the system under study and the calculations to be performed. The use of the #comments component of the Master Input File is optional.

5.5.1 Format of #comment Component

As presently configured, the #comment component may contain up to 100 lines. Each line may contain up to 72 characters.

5.5.2 Sample Conversation in Comments Mode

Below is a sample conversation with PREP while in *comment* mode. The user has put in a one line comment, returned to *control* mode by way of the *comment* editor, and issued a *summ* command to see the proposed contents of the Master Input File up to this point.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
cmts
comments input mode:
input next comment line (type '#' to edit)
This is a demonstration of the use of PREP.
input next comment line (type '#' to edit)
#
```

```
Comment edit mode
type p,pc,pl,u,n,d,dl,ud,i,ib,?, or ctrl :
ctrl
Control mode.
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort
or ?:
summ
#comment
 This is a demonstration of the use of PREP.
#beam
 0.0000000000000000E+00
 0.0000000000000000E+00
 0.0000000000000000E+00
 0.0000000000000000E+00
#menu
Control mode.
```

5.5.3 Comment Editor

PREP has a simple line editor for correcting mistakes in the #comment component. To assist the user, the editor has a "viewing window" that sits over what is referred to as the "current" line. The editor for #comment is activated by typing '#'. The available commands for the #comment editor are listed below. Here (CR) denotes "Carriage Return".

To leave the *comments* editor and return to *control* mode, type *ctrl*. The proposed contents of the Master Input File as prepared by PREP up to this point may then be examined by entering the command *summ*. Modifications in the #comment component may be made at any time by returning to the *comments* mode by way of the *control* mode.

If extensive changes in the #comment component are required, it may be easier to make them by using a local Editor on files 10 or 11 after completion of a PREP run.

5.5.4 Interspersed Comments

Any line in the MaryLie Master Input File beginning with an ! (explanation mark) is ignored by MaryLie. This feature may be used to place comment lines (almost) anywhere in the Master Input File. It may also be used to temporarily "comment out" various portions of the Master Input File. This application is particularly useful for making temporary changes in the #labor component of the Master Input File.

There is one caution with the use of interspersed comments. They must not be inserted between the name line for an element/command (in the #menu component) and the parameter line or parameter lines associated with that name line. See section 5.7.2. Correspondingly, if a name line is to be "commented out", its associated parameter line or parameter lines must also be "commented out".

5.6 Beam Mode

The #beam component is a required component for every Master Input File. PREP automatically sets all the entries of the #beam component equal to zero at the beginning of a

Comments Editor Commands		
Command	Usage	Result
p	p(CR)	print all of #comment component with line numbers. After this command is executed the current line is the last line.
pc	pc(CR)	print current line.
pl	pl(CR) m,n(CR)	print lines m through n. After this command is executed, the current line is line n. Thus, this command may be used with m=n to place the viewing window over line n.
u	u(CR)	Move viewing window up one line.
n	n(CR)	Move viewing window to the next line.
d	d(CR)	delete the current line. Viewing window now sits over the next line.
dl	dl(CR) m,n(CR)	delete lines m through n. Viewing window now sits over the old line n+1.
ud	ud(CR) m(CR)	undelete before line m. Inputs before line m the most recently deleted set of lines (by the command *d* or *dl*). Viewing window now sits over the old line m.
i	i(CR)	input after the current line. PREP goes into *comments* input mode. To return to the *comments* editor, type #.
ib	ib(CR) m(CR)	input before line m. PREP goes into *comments* input mode. To return to the *comments* editor, type #.
?	?(CR)	Get help from PREP.
ctrl	ctrl(CR)	Return to *control* mode.

PREP run. (This is a permissible set of values for certain kinds of MaryLie runs.) The user can modify the contents of the #beam component by using PREP in *beam* mode.

When in beam mode, PREP first asks for a Particle Type Code, as shown below:

Particle Type Code	Particle
e	electron or positron
p	proton or antiproton
h	H minus ion
d	deuteron
mu	muon
pi	pion
o	other

When type code *o* is used, the user is asked to specify charge (in units of e) and mass (in MeV/c^2). The charge should be input as a positive number. (MaryLie works with the absolute value of the input charge in any case.)

Next PREP will request that the user input the design momentum (in MeV/c) or the design *kinetic* energy (in MeV) of the beam. Using these values, PREP computes the magnetic rigidity and design relativistic beta and gamma factors of the beamline. There is also provision for altering these values.

Finally, PREP requests that the scale length be input (see section 4.1.2). It then returns to *control* mode.

5.6.1 Beam Editor

There is no editor for the #beam component. The proposed content of the Master Input File as prepared by PREP up to any given point may be examined by entering the command *summ*. The #beam component can be changed at any time, if desired, by entering the command *beam*, and then repeating the procedure of section 5.6 above.

If one wishes to bypass the use of PREP, as most experienced MaryLie users do, the contents of the #beam component can be set directly within a MaryLie run by using a command with type code *cbm*. See section 7.3.9.

5.6.2 Sample Conversation in Beam Mode

A sample conversation with PREP, specifying a proton beam with an energy of 800 MeV, and specifying a scale length of 2 meters, is shown below:

```
Typecmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
beam
beam input mode.
choose particle from listing:
     e (electron, positron)
     p (proton, anti-proton)
     h (H- ion)
     d (deuteron)
```

```
     mu (muon)
     pi (pion)
     o (other)
p

Absolute change (in units of e) is    1.000000000000000
Mass (in MeV/c*c) is   938.2723100000000
input design momentum in MeV/c
(note: if < 0, this will be assumed to be the design
         kinetic energy in MeV) :
-800

design energy and momentum:
kinetic energy (MeV) =     800.0000000000000
momentum (MeV/c) =     1463.296175078716
prep results:
values for brho,beta,gamma,(gamma-1), and abs(q/e) are
    4.881030646470486
   0.8418106683634142
    1.852630938240094
   0.8526309382400936
    1.000000000000000
type 1 if these are ok; type 0 to input your own values
1
lastly, input the scale length in meters :
2
Control mode.
```

5.6.3 Format of #beam Component

If the *summ* command is issued at this point, one obtains the result shown below for the proposed contents of the Master Input File:

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
  This is a demonstration of the use of PREP.
#beam
   4.881030646470486
  0.8526309382400936
   1.000000000000000
   2.000000000000000
#menu
Control mode.
```

Observe that the #beam component consists of 4 numbers:

First number	- Brho, the magnetic rigidity, in Tesla meters.
Second number	- the quantity (gamma-1) where gamma is the relativistic gamma factor.
Third number	- the quantity |q/e| , the absolute value of the charge in units of the electron charge.
Fourth number	- the scale length in meters.

Since MaryLie reads file 11 in free format, these numbers may be put on separate lines if space for full precision is required, or may be truncated and put on a single line if compactness is desired. PREP puts one number on each line.

The reader may be curious why the quantity (gamma-1) is specified rather than simply gamma. The answer has to do with numerical precision and round off. MaryLie computations require both the relativistic beta and the relativistic gamma factors. Both these quantities can be computed to machine precision from a knowledge of (gamma-1) even in extreme cases of very small or very large particle energy.

Finally, note that the quantity Brho is related to the particle charge q and the design momentum p^0 (see Section 4.1.2) by the equation

$$\text{Brho} = p^0/|q|. \tag{5.6.1}$$

5.7 Menu Mode

Much of PREP is devoted to the specification of the contents of a beam line or accelerator lattice, and the commands to be executed. Elements of a lattice (as well as of lines, lumps, and loops) and commands are drawn from a user specified *menu*. A *menu* is a table of elements, commands, and procedures. Each element, command, or procedure is given a user specified name, a type code mnemonic, and various required parameters. The user specified name may be composed of one to eight alpha-numeric characters. (It must, however, not begin with a number or with the symbols , + − # ! * .) Subsequent references to each element, command, or procedure are made by the user specified name. Thus, the user may refer to elements, commands, or procedures in a way that identifies their particular purpose.

User specified names must be unique. That is, *menu* entries, *lines*, *lumps*, and *loops* must all have different names. (See sections 5.8, 5.9, and 5.10 for discussion of *lines*, *lumps*, and *loops*, respectively.)

Every Master Input File must have a #menu component. It can be prepared by using PREP in *menu* mode. As presently configured, the total number of elements, commands, and procedures placed in the #menu component should not exceed 400.

When in the *menu* mode, PREP asks for the user specified name, then the type code mnemonic, and finally the required parameters. A list of currently available type codes is given below. Element type codes are described in detail in section 6, and command type codes are described in detail in sections 7 and 8. The type codes for procedures and fitting and optimization commands are described in section 9. Note that all type codes are given in lower case. If entries are made in upper case, they are automatically converted to lower case by PREP and MaryLie.

Currently available type codes:

Mnemonics for elements (see Sect. 6):

drft	:	drift
nbnd	:	normal entry and exit bending magnet
pbnd	:	parallel faced bending magnet
gbnd	:	general bending magnet
prot	:	rotation of reference plane
gbdy	:	body of a general bending magnet
frng	:	hard edge dipole fringe fields
cfbd	:	combined function bend
cfrn	:	combined function dipole fringe fields
sol	:	solenoid
quad	:	quadrupole
cfqd	:	combined function quadrupole
recm	:	REC quadrupole multiplet
sext	:	sextupole
octm	:	mag. octupole
octe	:	elec. octupole
srfc	:	short rf cavity
arot	:	axial rotation
thlm	:	thin lens low order multipole
cplm	:	"compressed" low order multipole
twsm	:	linear matrix via twiss parameters
dism	:	dispersion matrix
jmap	:	j mapping
rmap	:	random map
mark	:	marker
dp	:	data point
spce	:	space
arc	:	arc
usr1 ... usr10	:	user subroutines that act on phase space data
usr11 ... usr20	:	user subroutines that produce or act on maps
r****	:	random counterpart of element ****

Mnemonics for simple commands (see Sect. 7):

end	:	halt execution
of	:	open files
cf	:	close files
rt	:	ray trace
num	:	number lines in a file
circ	:	circulate
wcl	:	write contents of a loop
rapt	:	aperture particle distribution with rectangular aperture
eapt	:	aperture particle distribution with elliptic aperture
wnd	:	window a beam
wnda	:	window a beam in all planes
whst	:	write history of beam loss
pmif	:	print contents of master input file
ptm	:	print transfer map
tmi	:	input transfer map from an external file
tmo	:	output transfer map to an external file
ps1 ... ps9	:	parameter sets
rps1 ... rps9	:	random parameter sets
wps	:	write parameter set
stm	:	store the existing transfer map
gtm	:	get transfer map from storage
mask	:	mask off selected portions of transfer map
ftm	:	filter transfer map
sqr	:	square the existing map
symp	:	symplectify matrix in transfer map
iden	:	identity mapping
inv	:	invert
rev	:	reverse map
revf	:	reverse factorize
tran	:	transpose
tpol	:	twiss polynomial
dpol	:	dispersion polynomial
time	:	write time
cdf	:	change output drop file
bell	:	ring bell at terminal
wmrt	:	write out value of merit function
paws	:	pause
inf	:	change or write out values of infinities
zer	:	change or write out values of zeroes
cbm	:	change or write out beam parameters
dims	:	dimensions
wuca	:	write out contents of ucalc array
pli	:	path length information
shoa	:	show contents of arrays

Mnemonics for advanced commands (see Sect. 8):

cod	:	compute off-momentum closed orbit data
tasm	:	twiss analyze static map
tadm	:	twiss analyze dynamic map
ctr	:	change tune range
snor	:	static normal form analysis
dnor	:	dynamic normal form analysis
asni	:	apply script N inverse
rasm	:	resonance analyze static map
radm	:	resonance analyze dynamic map
sia	:	static invariant analysis
dia	:	dynamic invariant analysis
psnf	:	compute power of static normal form
pdnf	:	compute power of dynamic normal form
pnlp	:	compute power of nonlinear part
fasm	:	fourier analyze static map
fadm	:	fourier analyze dynamic map
pold	:	polar decompose matrix portion of transfer map
ppa	:	principal planes analysis
tbas	:	translate basis
exp	:	compute exponential
gbuf	:	get buffer contents
amap	:	apply map to a function or moments
moma	:	moment analysis
bgen	:	generate beam
tic	:	translate initial conditions
smul	:	multiply polynomial by a scalar
padd	:	add two polynomials
pmul	:	multiply two polynomials
pb	:	Poisson bracket two polynomials
mn	:	compute matrix norm
psp	:	polynomial scalar product
pval	:	evaluate a polynomial
sq	:	select quantities
wsq	:	write selected quantities
csym	:	check symplectic condition
geom	:	compute geometry of a loop
fwa	:	copy file to a working array
lnf	:	compute logarithm of normal form
merf	:	merge files

Mnemonics for procedures and fitting and optimization commands (see Sect. 9):

bip	:	begin inner procedure
bop	:	begin outer procedure
tip	:	terminate inner procedure
top	:	terminate outer procedure
aim	:	specify aims
vary	:	specify quantities to be varied
fit	:	fit to achieve aims
mss	:	minimize sum of squares optimization
opt	:	general optimization
mrt0	:	merit function (least squares)
mrt1 ... mrt5	:	merit functions (user written)
con1 ... con5	:	constraints
grad	:	compute gradient matrix
scan	:	scan parameter space
cps1 ... cps9	:	capture parameter set
fps	:	free parameter set
flag	:	change or write out values of flags and defaults
rset	:	reset menu entries

5.7.1 Sample Conversation in Menu Mode

Below is a sample conversation with PREP while in menu mode. The user has prepared a menu consisting of elements and commands.

The elements are a long and short drift (user names *drl* and *drs*), a horizontally focusing and a horizontally defocusing quadrupole (user names *hfq* and *hdq*), and a parallel faced dipole (user name *bend*).

The commands are to print the Master Input File (user name *fileout*), to print the transfer map (user name *mapout*), to write the transfer map on an external file (user name *wrtmap*), to replace the transfer map by the identity map (user name *clear*), and an end command (user name *bye*).

Also shown is the return to *control* mode by way of the *menu* editor.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
menu
menu input mode
input name and mnemonic for menu item#   1
(or type# to go to menu edit mode)
drl drft
input length of drift in meters
2.28
 drl      drft      parameters:
  2.280000000000000
input name and mnemonic for menu item#   2
(or type # to go to menu edit mode)
drs drft
input length 1 of drift in meters
.45
 drs      drft      parameters:
 0.4500000000000000
input name and mnemonic for menu item #   3
(or type # to go to menu edit mode)
hfq quad
input length(m), field grad.(t/m), lfrn, tfrn
where: field grad .ge. 0 for hor focussing quad,
       field grad .lt. 0 for hor defocussing quad
       lfrn=1 for leading edge fr fields, =0 otherwise
       tfrn=1 for trailing edge fr fields,=0 otherwise
.5  2.7  1  1
 hfq     quad    parameters:
 0.5000000000000000     2.700000000000000     1.000000000000000
  1.000000000000000
input name and mnemonic for menu item #   4
(or type # to go to menu edit mode)
hdq quad
input length(m), field grad.(t/m), lfrn, tfrn
where: field grad .ge. 0 for hor focussing quad,
       field grad .lt. 0 for hor defocussing quad
       lfrn=1 for leading edge fr fields, =0 otherwise
       tfrn=1 for trailing edge fr fields,=0 otherwise
.5  -1.9 1 1
 hdq    quad    parameters:
```

```
 0.5000000000000000      -1.900000000000000      1.000000000000000
  1.000000000000000
input name and mnemonic for menu item #   5
(or type # to go to menu edit mode)
bend  pbnd
input bend angle(deg), gap size (m), normalized field integral, b(tesla)
36  0  .5  1.2
 bend    pbnd   parameters:
  36.00000000000000   0.0000000000000000E+00   0.5000000000000000
  1.200000000000000
input name and mnemonic for menu item #   6
(or type # to go to menu edit mode)
fileout pmif
input itype,ifile,isend where:
     itype=0 to print contents as they are
          =1 to print as interpreted by MARYLIE
          =2 to print (interpreted) #labor component
      ifile= external output file number
      isend=1 to print on file6 (tty)
           =2 to print on file ifile
           =3 to do both
1 12  3
 fileout    pmif    parameters:
  1.000000000000000    12.00000000000000   3.000000000000000
input name and mnemonic for menu item #   7
(or type # to go to menu edit mode)
mapout ptm
input nm,nf,nt,nu,ibasis for ptm
1 1 0 0 1
 mapout     ptm     parameters:
  1.000000000000000         1.000000000000000   0.0000000000000000E+00
 0.0000000000000000E+00     1.000000000000000
input name and mnemonic for menu item #   8
(or type # to go to menu edit mode)
trace rt
input icfile,nfcfile,norder,ntrace,nwrite,ibrief:
13  14  5  1000  5  0
 rays     rt     parameters:
  13.00000000000000     14.00000000000000     5.000000000000000
  1000.000000000000     5.000000000000000     0.0000000000000000E+00
input name and mnemonic for menu item #   9
(or type # to go to menu edit mode)
wrtmap tmo
there are  1 parameters for type tmo
input ifile, the file number on which map is to be written.
16
 wrtmap   tmo   parameters:
  16.00000000000000
input name and mnemonic for menu item #   10
(or type # to go to menu edit mode)
clear iden
there are  0 parameters for type iden
no parameters needed for reset to identity
```

```
 clear   iden   parameters:
 none
input name and mnemonic for menu item #   11
(or type # to go to menu edit mode)
bye end
there are 0 parameters for type end
no parameters for end command
 bye     end     parameters:
 none
input name and mnemonic for menu item #   12
(or type # to go to menu edit mode)
#
menu edit mode
type p, pl, i, ? or ctrl :
ctrl
Control mode.
```

5.7.2 Format of #menu Component

If the *summ* command is issued at this point one obtains the results shown below for the proposed contents of the Master Input File:

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
 This is a demonstration of the use of PREP.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  2.000000000000000
#menu
 drl     drft
   2.28000000000000
 drs    drft
  0.450000000000000
 hfq    quad
  0.500000000000000       2.70000000000000          1.00000000000000
   1.00000000000000
  hdq   quad
   0.500000000000000     -1.90000000000000          1.00000000000000
    1.00000000000000
  bend    pbnd
   36.00000000000000     0.0000000000000000E+00     0.5000000000000000
   1.200000000000000
  fileout    pmif
    1.00000000000000     12.0000000000000          3.00000000000000
  mapout    ptm
    1.00000000000000        1.00000000000000     0.000000000000000E+00
   0.000000000000000E+00    1.00000000000000
  trace    rt
    13.0000000000000     14.0000000000000           5.00000000000000
    1000.00000000000     5.00000000000000          0.000000000000000E+00
```

```
  wrtmap    tmo
    16.0000000000000
  clear    iden
  bye     end
Control mode.
```

From the contents of the #menu component, one sees that each *menu* item has the following format:

> A "name" line giving the user specified name and type code mnemonic for an element or command. Names may contain up to 8 characters.
>
> One or more associated "parameter" lines following the "name" line. They specify the parameters required by the particular type code. Since MaryLie reads file 11 in free format, the parameters may occur on separate lines, or may be grouped together. For convenience, PREP puts at most 3 parameters on a line. If a particular type code requires no parameters, then there is no associated parameter line.

5.7.3 Menu Editor

When in menu input mode, the contents of any menu item may be changed simply be re-entering its name and then following instructions. Thus, if a mistake is made while entering some menu item, one should

- complete entering that particular menu item and its parameters,
- then re-enter the name of that item and follow instructions.

There is also a simple *menu* editor to facilitate examination of the #menu component. It is activated by typing '#'. The available commands for the #menu editor are listed below.

Menu Editor Commands		
Command	Usage	Result
p	p(CR)	print the entire menu.
pl	pl(CR) m,n(CR)	print menu lines (entries) m through n.
i	i(CR)	Return to menu input mode.
?	?(CR)	Get help from PREP.
ctrl	ctrl(CR)	Return to control mode.

Modifications in the #menu component may be made at any time by returning to the *menu* mode by way of the *control* mode. If extensive modifications are required, it may be easier to make them by using a local Editor on files 10 or 11 after completion of a PREP run.

5.8 Line Mode

A *line* is a collection of elements and/or commands referenced by a single user specified name. All possible #menu items, save for commands with type codes *bip*, *bop*, *tip*, and *top*, may appear in a line. See sections 9.1, 9.2, 9.3, and 9.4 Line names may contain up to 8 characters. (They must, however, not begin wih a number or with the symbols , + − # ! * .) All names must be unique. That is, *menu* entries, *lines*, *lumps*, and *loops* must all have different names. (See sections 5.7, 5.9, and 5.10 for a discussion of *menu* entries, *lumps*, and *loops*, respectively.)

The use of the lines feature of MaryLie makes it possible to define superelements and supercommands.

Lines may contain other lines up to a logical depth of 20. Lines may also contain *lumps*. (See section 5.9 for a discussion of *lumps*.) The contents of a line must meet the requirement that upon translation into a string of elements and/or commands, all items in the string must appear in the *menu* (or be *lumps*). (See section 5.7 for a description of the #menu component of the Master Input File.)

The user may define up to 100 lines, with up to 100 entries in each. (Actually, the total number of lines, lumps, and loops must not exceed 100.)

The #lines component is an optional component of the Master Input File. For examples of the use of *lines*, see chapters 2 and 10.

5.8.1 Sample Conversation in Line Mode

Below is a sample conversation with PREP while in *line* mode. The user has defined three lines with the user specified names *cell*, *ring*, and *study*. The line *cell* contains a collection of elements drawn from the menu. The line *ring* is defined in terms of the line *cell*, and contains 10 cells. The line *study* is a collection of elements and commands. Note that PREP first asks for the name of a *line*, and then for its contents. Return to *control* mode is made by way of the *line* editor.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
line
line input mode
input name for the next line (or type # to edit)
cell
input contents of cell    (use & to cont on next line):
drl hdq drs bend drs hfq drl
input name for the next line (or type # to edit)
ring
input contents of ring     (use & to cont on next line):
10*cell
input name for the next line (or type # to edit)
study
input contents of study    (use & to cont on next line):
clear drl mapout hdq mapout clear
input name for the next line (or type # to edit)
#
line edit mode
```

```
type p, pl, i, ? or ctrl :
ctrl
Control mode.
```

5.8.2 Format of #lines Component

If the *summ* command is issued at this point, one gets the results listed below for the proposed contents of the Master Input File.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
 This is a demonstration of the use of PREP.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  2.000000000000000
#menu
 drl     drft
   2.28000000000000
 drs     drft
  0.450000000000000
 hfq    quad
  0.500000000000000     2.70000000000000     1.00000000000000
   1.00000000000000
 hdq    quad
  0.500000000000000    -1.90000000000000     1.00000000000000
   1.00000000000000
 bend    pbnd
  36.00000000000000    0.0000000000000000E+00  0.5000000000000000
   1.200000000000000
 fileout    pmif
   1.00000000000000     12.0000000000000     3.00000000000000
 mapout    ptm
   1.00000000000000     1.00000000000000     0.000000000000000E+00
  0.000000000000000E+00     1.00000000000000
 trace    rt
  13.0000000000000     14.0000000000000     5.00000000000000
  1000.00000000000     5.00000000000000      0.000000000000000E+00
 wrtmap    tmo
   16.0000000000000
 clear    iden
 bye    end
#lines
 cell
     1*drl     1*hdq     1*drs     1*bend     1*drs     &
     1*hfq     1*drl
 ring
     10*cell
 study
     1*clear     1*drl     1*mapout     1*hdq     1*mapout     &
     1*clear
```

```
#lumps
#loops
Control mode.
```

From the content of the #lines component, one sees that each line has the following format:

> A "name" entry giving the user specified name of the line. Names may contain up to 8 characters.
>
> One or more associated "contents" entries on lines following the name entry line. These entries specify the contents of the *line*. Each such "contents" entry line must end with an '&' sign if it is to be followed by another "contents" entry line. Since MaryLie reads file 11 in free format, the contents of a *line* may occur on separate entry lines, or may be grouped together. For convenience, PREP puts at most 5 entries on a line.

5.8.3 Multiplicative Notation

If a line contains several of the same items in a row, this fact may be represented by the use of a multiplicative (repetition number) notation. For example, if *hfq* is the user specified name for a particular quadrupole, then two such quadrupoles in a row may be specified by writing either hfq hfq or 2*hfq. By default, both in PREP and MaryLie, hfq means the same as 1*hfq.

Multiplicative notation may also be used with lines and lumps. For example if *name* is the user specified name of a line or a lump, then m*name (with m> 0) has the same effect as m successive occurrences of name.

If name is the user specified name for an element, command, line, or lump (within a line/lump), then 0*name has the same effect as omitting name. This feature is useful for temporarily "commenting out" an item.

Negative repetition numbers may also be used. Their effect requires a more extensive description. First, by default in both PREP and MaryLie , −name means the same thing as −1*name. Second, if name is an element, command, or *previously constructed* lump, then m*name and −m*name have the same effect. (See section 5.9 for a description of how and when lumps are constructed.)

The case of lines and unmade lumps is somewhat more complicated. Suppose, for example, that *name* is a line with entries a,b,c where a,b,c are elements and/or commands. Then, as described above, 2*name is equivalent to the sequence a b c a b c. However, −1*name is equivalent to the sequence c b a, and −2*name is equivalent to the sequence c b a c b a. That is, a negative repetition number for a line produces a *reversed* sequence.

The situation with unmade lumps is similar. Suppose, for example, that name is a lump that has not been previously constructed, and that name has entries a,b,c where a,b,c are elements and/or commands. Then 1*name constructs and stores (and also concatenates) the map associated with the sequence a b c. Similarly 2*name produces the same map, but concatenates it twice. This, of course, is equivalent to concatenating the square of the map. By contrast, −1*name constructs, stores, and concatenates the map associated with the *reversed* sequence c b a.

The use of negative repetition numbers with lines/lumps within other lines/lumps is still more complicated. Consider, for example, the case when the lines *line1*, *line2*, and *line3* are defined by the following #line entries:

```
#lines
     line1
          a b c
     line2
          -1*line1
     line3
          -1*line2
```

Here a,b,c are elements and/or commands as before. Then, as described above, *line2* is equivalent to the reversed sequence c b a. Correspondingly, *line3* is equivalent to the "reversed reversed" sequence a b c. That is, lines/(unmade) lumps nested within other lines/(unmade) lumps "inherit" negative signs from the repetition numbers of the lines/(unmade) lumps they are nested within. The general rule for lines and unmade lumps is that nested repetition numbers (including their signs) behave as if they were simply multiplied together. One consequence of this rule is that the actual map that is produced and stored when a lump is constructed depends on the context in which the lump is found. In general, negative repetition numbers should be used with care.

5.8.4 Line Editor

When in *line* input mode, the name and contents of any *line* item may be changed simply by re-entering its name and then following instructions. Thus, if a mistake is made while entering a *line* item, one should

- complete entering the name of the line and at least one item in its contents,
- then re-enter the name of that line and follow instructions.

There is also a simple *line* editor to facilitate examination of the #lines component. It is activated by typing '#'. The available commands for the #lines editor are listed below.

Modifications in the #lines component may be made at any time by returning to the *line* mode by way of the *control* mode. If extensive modifications are required, it may be easier to make them by using a local Editor on files 10 or 11 after completion of a PREP run.

5.9 Lump Mode

A *lump* is a collection of elements and/or commands whose ultimate result is the construction by MaryLie of a map. This map is stored by MaryLie, and is available for later use. Each lump is given a user specified name, and each time the name of a particular lump is invoked, the corresponding map is recalled from memory.

Lump names may contain up to 8 characters. (They must, however, not begin with a number or with the symbols , + − # ! * .) All names must be unique. That is, *menu*

Line Editor Commands		
Command	Usage	Result
p	p(CR)	print the entire contents of the #lines component.
pl	pl(CR) m,n(CR)	print lines m through n.
i	i(CR)	Return to menu input mode.
?	?(CR)	Get help from PREP.
ctrl	ctrl(CR)	Return to *control* mode.

entries, *lines*, *lumps*, and *loops* must all have different names. (See sections 5.7, 5.8, and 5.10 for a discussion of *menu* entries, *lines*, and *loops*, respectively.)

Lumps may contain lines and other lumps up to a logical depth of 20. (See section 5.8 for a description of the #lines component of the Master Input File.) A lump must meet the requirement that upon translation into a string of elements and/or commands, all items in the string must appear in the *menu* or be other lumps. (See section 5.7 for a description of the #menu component of the Master Input File.)

MaryLie constructs the map associated with a lump as follows:

- A storage area is set aside for the map to be associated with the lump, and the identity map is initially stored in this area.
- The contents of the lump are translated into a string of elements and/or commands. Each time an element is encountered in the string, the map for that element is computed and concatenated with whatever map is in the storage area. The map in the storage area is then replaced by the result of this concatenation. (See section 3.11.2 for a description of the concatenation of maps.) Whenever another lump is encountered in the string, the map for that lump is concatenated with whatever map is in the storage area. The map in the storage area is then replaced by the result of this concatenation. Whenever a command is encountered in the string, it is examined to see whether or not it is of the type that acts on a map. If it is not, the command is ignored, and a message to this effect is written out by MaryLie at the terminal. (A ray trace command is an example of a command that does not act on a map.) If the command is of the type that acts on a map, the command is applied to the map in the storage area. The map in the storage area is then replaced by the map resulting from the application of this command. [A square command (type code mnemonic *sqr*) is an example of a command that acts on a map.]

Thus after all the items in the string associated with the contents of the lump have been encountered, the result is a stored map.

The user may define up to 100 lumps with up to 100 entries in each. (Actually, the total number of lines, lumps, and loops must not exceed 100.) In actual operation, MaryLie does not construct and store the map for a lump unless and until it is required as a result of some entry in the #labor component of the Master Input File. See section 5.11. As presently configured, at any given moment the total allowed number of stored maps associated with lumps is 20. If the user tries to construct more than 20 lumps, some of the existing (already constructed) lumps are destroyed on a first made first destroyed basis. The assumption is that recently constructed lumps are more likely to be required for subsequent calculations than ancient lumps. Section 5.9.3 describes how the user may control the destruction of lumps in a completely flexible way if required.

The #lumps component is an optional component of the Master Input File. For examples of the use of *lumps*, see sections 2.7 and 10.4.

5.9.1 Sample Conversation in Lump Mode

Below is a sample conversation with PREP while in *lump* mode. The user has defined two lumps called *lmpbend* and *lmpcell*. The lump *lmpbend* consists of the map for the element *bend* listed in #menu. The lump *lmpcell* consists of the map corresponding to all the elements in the line *cell* when their individual maps are concatenated together. The line *cell* is assumed to be defined in the #lines component of the Master Input File. Note that PREP first asks for the name of a *lump*, and then for its contents. Return to *control* mode is made by the way of the *lump* editor.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
lump
lump input mode
input name for the next lump (or type # to edit)
lmpbend
input contents of lmpbend (use & to cont on next line):
bend
input name for the next lump (or type # to edit)
lmpcell
input contents of lmpcell (use & to cont on next line):
cell
input name for the next lump (or type # to edit)
#
lump edit mode
type p, pl, i, ? or ctrl :
ctrl
Control Mode.
```

5.9.2 Format of the #lumps Component

If the *summ* command is issued at this point, one gets the results listed below for the proposed contents of the Master Input File.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
```

```
 This is a demonstration of the use of PREP.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  2.000000000000000
#menu
 drl    drft
   2.28000000000000
 drs drft
  0.450000000000000
 hfq   quad
  0.500000000000000     2.70000000000000     1.00000000000000
   1.00000000000000
 hdq   quad
  0.500000000000000    -1.90000000000000     1.00000000000000
   1.00000000000000
 bend    pbn
  36.00000000000000     0.0000000000000000E+00     0.5000000000000000
  1.200000000000000
 fileout    pmif
   1.00000000000000     12.0000000000000     3.00000000000000
 mapout    ptm
   1.00000000000000    1.00000000000000    0.000000000000000E+00
  0.000000000000000E+00     1.00000000000000
 trace    rt
   13.0000000000000     14.0000000000000     5.00000000000000
   1000.00000000000     5.00000000000000      0.000000000000000E+00
 wrtmap tmo
   16.0000000000000
 clear   iden
 bye end
#lines
 cell
     1*drl     1*hdq     1*drs     1*bend     1*drs     &
     1*hfq     1*drl
 ring
    10*cell
 study
     1*clear     1*drl     1*mapout     1*hdq     1*mapout     &
     1*clear
#lumps
 lmpbend
     1*bend
 lmpcell
     1*cell
#loops
Control mode.
```

From the content of the #lumps component one sees that each lump has the following format:

> A "name" line giving the user specified name of the *lump*. Names may contain up to 8 characters.

One or more associated "contents" lines following the name line. These lines specify the contents of the *lump*. Each such "contents" line must end with an '&' sign if it is to be followed by another "contents" line. Since MaryLie reads file 11 in free format, the contents of a *lump* may occur on separate entry lines, or may be grouped together. For convenience, PREP puts at most 5 entries on a line.

5.9.3 Multiplicative Notation

If a lump appears repeatedly, then it may either be written repeatedly, or a repetition number may be used. For example, if *name* is a lump, then 2*name is equivalent to name name. By default, both in PREP and in MaryLie , name means the same as 1*name.

The case of 0*name, when *name* is a lump, is more complicated. If 0*name appears within a line, lump, or a loop, then it is ignored. The same is true if 0*name appears in #labor (see section 5.11) and *name* has *not* actually been constructed. (See section 5.9 for a description of how and when lumps are constructed.) This feature is useful for temporarily "commenting out" a lump.

By contrast, if *name* has been constructed previously and 0*name appears in #labor, then the map for this lump is destroyed at that point in the MaryLie run, and its associated storage allocation is made available for the construction and storage of the map for another lump. This MaryLie feature places the construction and destruction of lumps (actually their associated maps) under the direct control of the user.

Negative repetition may also be used (with care) for lumps. See section 5.8.3.

5.9.4 Lump Editor

When in *lump* input mode, the name and contents of any *lump* item may be changed simply by re-entering its name and then following instructions. Thus, if a mistake is made while entering a *lump* item, one should

- complete entering the name of the lump and at least one item in its contents,
- then re-enter the name of that lump and follow instructions.

There is also a simple *lump* editor to facilitate examination of the #lumps component. It is activated by typing '#'. The available commands for the #lumps editor are listed below.

Modifications in the #lumps component may be made at any time by returning to the *lump* mode by way of the *control* mode. If extensive modifications are required, it may be easier to make them by using a local Editor on files 10 or 11 after completion of a PREP run.

Lump Editor Commands		
Command	Usage	Result
p	p(CR)	print the entire contents of the #lumps component.
pl	pl(CR) m,n(CR)	print lines m through n.
i	i(CR)	Return to lumps input mode.
?	?(CR)	Get help from PREP.
ctrl	ctrl(CR)	Return to *control* mode.

5.10 Loop Mode

A *loop* is a collection of elements, lines, and/or lumps to be used for tracking or other studies. (See section 5.8 for a description of *lines*, and section 5.9 for a description of *lumps*.) The loop feature of MaryLie permits tracking using more than one transfer map. See sections 2.7, 10.3, 10.4, and 10.8. The loop feature also is used in the computation of the geometry of a beamline or circular machine. See sections 10.9 and 10.11 through 10.14.

Each loop is given a user specified name. Names may contain up to 8 characters. (They must, however, not begin with a number or with the symbols , + − # ! * .) All names must be unique. That is, *menu* entries, *lines*, *lumps*, and *loops* must all have different names. (See sections 5.7, 5.8, and 5.9 for a discussion of *menu* entries, *lines*, and *lumps*, respectively.)

A loop must meet the requirement that upon translation into a string of elements and/or commands and/or lumps, all items in the string must either appear in the *menu* or be lumps. (See section 5.7 for a description of #menu.) Although commands may appear either implicitly or explicitly in a loop, they are, in fact, all ignored except for those that act on phase-space data and those that occur within the definitions of lumps. Loops may contain other loops up to a logical depth of 20.

As presently configured, the user may define up to 100 loops with up to 100 entries in each. (Actually, the total number of lines, lumps, and loops must not exceed 100.) When a loop is completely translated into a string as described above, the total number of resulting items must be less than 1000.

The #loops component is an optional component of the Master Input File. Loops are used in conjunction with commands having the type code mnemonics *circ*, *wcl*, and *geom*. See sections 7.3, 7.33, and 8.38. For examples of the use of *loops*, see Chapters 2 and 10.

5.10.1 Sample Conversation in Loop Mode

Below is a sample conversation with PREP while in *loop* mode. The user has defined two loops with the names *turtle* and *hare*. The loop *turtle* contains the line *cell*, and the loop *hare*

contains the lump *lmpcell*. The line *cell* is assumed to be defined in the #lines component of the Master Input File, and the lump *lmpcell* is assumed to be defined in the #lumps component of the Master Input File. Note that PREP first asks for the name of a *loop*, and then for its contents. Return to *control* mode is made by way of the *loop* editor.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
loop
loop input mode
input name for the next loop (or type # to edit)
turtle
input contents of turtle   (use & to cont on next line):
cell
input name for the next loop (or type # to edit)
hare
input contents of hare     (use & to cont on next line):
lmpcell
input name for the next loop (or type # to edit)
#
loop edit mode
type p, pl, i, ? or ctrl :
ctrl
control mode.
```

5.10.2 Format of the #loops Component

If the *summ* command is issued at this point, one gets the results listed below for the proposed contents of the Master Input File.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
 This is a demonstration of the use of PREP.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  2.000000000000000
#menu
 drl    drft
   2.28000000000000
 drs    drft
  0.450000000000000
 hfq    quad
  0.500000000000000     2.70000000000000     1.00000000000000
   1.00000000000000
 hdq    quad
  0.500000000000000     -1.90000000000000     1.00000000000000
   1.00000000000000
 bend    pbnd
   36.00000000000000     0.0000000000000000E+00     0.5000000000000000
   1.200000000000000
 fileout    pmif
```

```
   1.00000000000000     12.0000000000000    3.00000000000000
 mapout    ptm
   1.00000000000000     1.00000000000000    0.000000000000000E+00
  0.000000000000000E+00     1.00000000000000
 trace    rt
   13.0000000000000     14.0000000000000     5.00000000000000
   1000.00000000000    5.00000000000000     0.000000000000000E+00
 wrtmap tmo
   16.0000000000000
 clear    iden
 bye      end
#lines
 cell
     1*drl     1*hdq     1*drs     1*bend     1*drs     &
     1*hfq    1*drl
 ring
    10*cell
 study
     1*clear     1*drl    1*mapout     1*hdq     1*mapout     &
     1*clear
#lumps
 lmpbend
     1*bend
 lmpcell
     1*cell
#loops
 turtle
     1*cell
  hare
     1*lmpcell
Control mode.
```

From the content of the #loops component, one sees that each loop has the following format:

> A "name" line giving the user specified name of the *loop*. Names may contain up to 8 characters.
>
> One or more associated "contents" lines following the name line. These lines specify the contents of the *loop*. Each such "contents" line must end with an '&' sign if it is to be followed by another "contents" line. Since MaryLie reads file 11 in free format, the contents of a *loop* may occur on separate entry lines, or may be grouped together. For convenience, PREP puts at most 5 entries on a line.

5.10.3 Multiplicative Notation

There is no reason for a loop to appear repeatedly in the #labor component of the Master Input File. See section 5.11. Consequently, if *name* is a loop and m is a positive integer, then m*name has the same effect as 1*name when appearing in #labor. The only exception is that an entry of the form 0*name is ignored. Finally, by default both in PREP and MaryLie , name means the same as 1*name.

Negative repetition numbers may be used with loops with care. By default, −name has the same effect as −1*name.

If *name* is a loop and −name (or −1*name) appears in #labor, the effect is that *name* is constructed in reverse order. Also, consistent with the description above, −m*name has the same effect as −1*name. Note that if a loop has a negative repetition number, then lines, *unmade* lumps, and loops inside this loop will inherit the negative sign associated with the negative repetition number. See section 5.8.3.

It is possible to have repeated loops inside other loops. In this case, −m*name gives the same results as taking the contents of m*name in reversed order (providing that whatever lumps are referenced have not been constructed previously). Specifically, repetition numbers for loops within other loops behave just as they do for nested lines. See section 5.8.3.

5.10.4 Loop Editor

When in *loop* input mode, the name and contents of any *loop* item may be changed simply by re-entering its name and then following instructions. Thus, if a mistake is made while entering a *loop* item, one should

- complete entering the name of the *loop* and at least one item in its contents,
- then re-enter the name of that *loop* and follow instructions.

There is also a simple *loop* editor to facilitate examination of the #loops component. It is activated by typing '#'. The available commands for the #loops editor are listed below.

Loop Editor Commands		
Command	Usage	Result
p	p(CR)	print the entire contents of the #loops component.
pl	pl(CR) m,n(CR)	print lines m through n.
i	i(CR)	Return to loops input mode.
?	?(CR)	Get help from PREP.
ctrl	ctrl(CR)	Return to *control* mode.

Modifications in the #loops component may be made at any time by returning to the *loop* mode by way of the *control* mode. If extensive modifications are required, it may be easier to make them by using a local Editor on files 10 or 11 after completion of a PREP run.

5.11 Labor Mode

As indicated earlier, the #labor component specifies the actual system to be studied and what operations are actually to be carried out. Thus, the #labor component is a list of elements, commands, procedures, lines, lumps, and loops. As presently configured, this list may contain up to 1000 entries.

The entries in the #labor component are drawn from the #menu, #lines, #lumps, and #loops components. That is, all entries in the #labor component must also occur elsewhere as user specified names in at least one of the #menu, #lines, #lumps, and #loops components. PREP checks to make sure that this is the case. *Moreover, the user specified names of all entries in the #menu, #lines, #lumps, and #loops components must be unique.*

During program execution, MaryLie reads the entries in the labor list sequentially. It is the entries of the #labor component that determine what will actually be done in a given MaryLie run. For this reason, the #labor component is a required component for every Master Input File.

Throughout a run, MaryLie keeps in memory a transfer map that will be referred to as the *total transfer map*. The map consists of matrix and Lie generator parts plus path length. At the beginning of a run, the total transfer map is set equal to the identity map. When MaryLie encounters the name of an *element* as it reads the labor list, it calculates the transfer map for that element (based on the parameters specified for that element in #menu). Specifically, it computes the numerical values for the entries of the matrix R and the polynomials f_3 and f_4 that represent the transfer map. It also computes path length and stores the result in $f(0)$. Next, it concatenates this map with the *total transfer map*, and stores the result as the *new total transfer map*. Path length is also accumulated. (See section 3.11.2 for a description of the concatenation of maps.) Finally, MaryLie discards both the transfer map for the element it has just encountered and the previous total transfer map, and retains only the new total transfer map. Thus,

> **MaryLie concatenates as it steps through the labor list, maintains the new (current) total transfer map in storage, and discards all previous maps.**

Following concatenation, the next entry will be read from the labor list.

Similarly, when MaryLie reads the name of a lump in the labor list, it computes (or recalls from memory if previously computed) the transfer map for that lump. The transfer map for the lump is then concatenated with the total transfer map to give a new transfer map.

When MaryLie reads an entry that is a *command*, it will do whatever operation is requested (e.g., a ray trace or a map manipulation). It then again reads the next entry from the labor list.

If MaryLie encounters the name of a line in the labor list, it carries out all commands and concatenates all successive maps associated with this line.

This whole process stops when MaryLie encounters an entry in the labor list with the type code *end*. (See section 7.1.) Thus, each labor list must have somewhere an entry with the type code *end*. Usually this is the last entry in the list. However, it may occur earlier. In that case, all subsequent entries in the labor list are ignored.

At the beginning of this section, it was pointed out that it is the entries of the #labor component that determine what will actually be done in a given MaryLie run. This feature of MaryLie makes it possible for the #menu, #lines, #lumps, and #loops components to have entries that are *not* used in a given MaryLie run. Indeed, MaryLie is organized in such a way that maps for elements in #menu are not computed unless they are referenced either explicitly or implicitly in #labor. Similarly, lumps are not constructed, lines are not translated, etc., unless they are actually needed. Thus, the user may make a variety of MaryLie runs simply by changing the contents of the #labor component.

For examples of various MaryLie runs, and how they are specified by the contents of #labor, see chapters 2 and 10.

5.11.1 Sample Conversation in Labor Mode

Below is a sample conversation with PREP while in *labor* mode. The user has specified a MaryLie run that computes a variety of maps, some of which are written on an external file for later use, and then repeatedly traces rays through a simple cell.

Note that PREP asks for the entries in the labor list one at a time. Return to *control* mode is made by way of the *labor* editor.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
labor
labor input mode:
input next labor entry (type '#' to edit)
fileout
input next labor entry (type '#' to edit)
study
input next labor entry (type '#' to edit)
ring
input next labor entry (type '#' to edit)
wrtmap
input next labor entry (type '#' to edit)
clear
input next labor entry (type '#' to edit)
lmpcell
input next labor entry (type '#' to edit)
wrtmap
input next labor entry (type '#' to edit)
trace
input next labor entry (type '#' to edit)
bye
input next labor entry (type '#' to edit)
#
labor edit mode
type p,pc,pl,u,n,d,dl,i,ib,?, or ctrl :
ctrl
Control mode.
```

5.11.2 Format of the #labor Component

If the *summ* command is issued at this point, one gets the results listed below for the proposed contents of the Master Input File.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
 This is a demonstration of the use of PREP.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  2.000000000000000
#menu
 drl    drft
   2.28000000000000
 drs    drft
  0.450000000000000
 hfq    quad
  0.500000000000000      2.70000000000000      1.00000000000000
   1.00000000000000
 hdq    quad
  0.500000000000000     -1.90000000000000     1.00000000000000
   1.00000000000000
 bend    pbnd
  36.00000000000000     0.0000000000000000E+00     0.5000000000000000
  1.200000000000000
 fileout    pmif
   1.00000000000000      12.0000000000000      3.00000000000000
 mapout    ptm
   1.00000000000000      1.00000000000000      0.000000000000000E+00
  0.000000000000000E+00      1.00000000000000
 trace    rt
   13.0000000000000      14.0000000000000      5.00000000000000
   1000.00000000000      5.00000000000000      0.000000000000000E+00
 wrtmap    tmo
   16.0000000000000
 clear   iden
 bye     end
#lines
 cell
     1*drl     1*hdq     1*drs     1*bend     1*drs     &
     1*hfq      1*drl
 ring
    10*cell
 study
     1*clear     1*drl     1*mapout     1*hdq     1*mapout     &
     1*clear
#lumps
 lmpbend
      1*bend
 lmpcell
```

```
      1*cell
#loops
 turtle
     1*cell
 hare
     1*lmpcell
#labor
    1*fileout
    1*study
    1*ring
    1*wrtmap
    1*clear
    1*lmpcell
    1*wrtmap
    1*trace
    1*bye
Control mode.
```

From the contents of the #labor component, one sees that it consists of a simple list with *one* entry on each line. Indeed, MaryLie reads the #labor component one line at a time, and expects only one entry on each line. Thus, unlike other components of the Master Input File, the *#labor component must have only one entry on each line.* Since MaryLie reads each line in free format, the entry may begin anywhere on the line.

5.11.3 Multiplicative Notation

As described previously, menu entries, lines, lumps, and (by mistake) loops may appear repeatedly in #labor. This can be achieved more compactly by writing m*name where m is an integer and *name* is a menu entry, line, lump, or loop. See sections 5.8.3, 5.9.3, and 5.10.3 for more detail. Finally, name means the same as 1*name.

5.11.4 Labor Editor

PREP has a simple line editor for correcting mistakes in the #labor component. To assist the user, the editor has a "viewing window" that sits over what is referred to as the "current" line. The editor for #labor is activated by typing '#'. The available commands for the #labor editor are listed below.

To leave the *labor* editor and return to *control* mode, type *ctrl*. Modifications in the #labor component may be made at any time by returning to the *labor* mode by way of the *control* mode.

If extensive changes in the #labor component are required, it may be easier to make them by using a local Editor on files 10 or 11 after completion of a PREP run.

5.12 Use of the Summary Command in PREP

While in *control* mode, the user may type *summ* to obtain a listing of what has been accomplished to date in a PREP session. Illustrations of the use of the *summ* command are given

Labor Editor Commands		
Command	Usage	Result
p	p(CR)	print all of #labor component with line numbers. After this command is executed the current line is the last line.
pc	pc(CR)	print current line.
pl	pl(CR) m,n(CR)	print lines m through n. After this command is executed, the current line is line n. Thus, this command may be used with m=n to place the viewing window over line n.
u	u(CR)	Move viewing window up one line.
n	n(CR)	Move viewing window to the next line.
d	d(CR)	delete the current line. Viewing window now sits over the next line.
dl	dl(CR) m,n(CR)	delete lines m through n. Viewing window now sits over the old line n+1.
i	i(CR)	input after the current line. PREP goes into *labor* input mode. To return to the *labor* editor, type #.
ib	ib(CR) m(CR)	input before line m. PREP goes into *labor* input mode. To return to the *labor* editor, type #.
?	?(CR)	Get help from PREP.
ctrl	ctrl(CR)	Return to *control* mode.

in sections 5.6.3, 5.7.2, 5.8.2, 5.9.2, 5.10.2, and 5.11.2. In essence, the *summ* command gives a listing of what PREP has prepared to date to put on file 10.

5.13 Use of the File Command in PREP

While in *control* mode, the user may type the command *file* at any time during a PREP run. This command causes file 10 to be rewound, and then causes PREP to write on file 10 whatever has been specified to date about the #comment, #beam, #menu, #lines, #lumps, #loops, and #labor components of the Master Input File. When a PREP run is terminated by means of the *exit* command, the final results of the PREP run are automatically written on file 10. If a PREP run is terminated with the *abort* command, this final writing on file 10 is bypassed.

It is recommended that the *file* command be entered from time to time in the course of a PREP run. Then, should the computer crash, all that might be lost are the entries put in after the last *file* command.

An example of the use of the *file* command is shown below.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
file
master input file written on file 10
Control mode.
```

5.14 Terminating and Resuming a PREP Run

A PREP run may be terminated by entering the commands *exit* or *abort* while in *control* mode. Use of the *exit* command is shown below.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
exit
master input file written on file 10
End of PREP.
```

As explained in section 5.13, use of the *exit* command automatically results in the issuing of a final *file* command before termination. Use of the *abort* command causes termination without the issue of a final *file* command.

A partially satisfactory or partially complete file 10 or 11 can be used as input at the beginning of a PREP run. In this way, PREP can be used to edit or complete a file 10. The user simply copies the existing file 10 or 11 into file 9, and then reads file 9 at the beginning of a PREP run. See section 5.3. Shown below is a sample PREP conversation in which use has been made of file 9.

```
Do you want instructions? (y,n) <n>
n
Read a PREP data set from file 9? (y,n,?) <n>
y
file 9 read in.
Control mode.
```

If the *summ* command is issued at this point, one obtains the output shown below thus indicating that PREP has been informed of the current state of the Master Input File, and is ready to receive corrections and/or additions.

```
Type cmts,beam,menu,line,lump,loop,labor,summ,file,exit,abort or ?:
summ
#comment
 This is an example of the use of PREP.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  2.000000000000000
#menu
 drl      drft
   2.28000000000000
 drs      drft
  0.450000000000000
 hfq      quad
  0.500000000000000       2.70000000000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.500000000000000      -1.90000000000000       1.00000000000000
   1.00000000000000
 bend     pbnd
  36.00000000000000      0.0000000000000000E+00  0.5000000000000000
  1.200000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   1.00000000000000       1.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 trace    rt
   13.0000000000000       14.0000000000000       5.00000000000000
   1000.00000000000       5.00000000000000      0.000000000000000E+00
 wrtmap   tmo
   16.0000000000000
 clear    iden
 bye      end
#lines
 cell
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 ring
    10*cell
 study
     1*clear       1*drl         1*mapout      1*hdq         1*mapout   &
     1*clear
#lumps
 lmpbend
     1*bend
 lmpcell
     1*cell
```

```
#loops
 turtle
     1*cell
 hare
     1*lmpcell
#labor
    1*fileout
    1*study
    1*ring
    1*wrtmap
    1*clear
    1*lmpcell
    1*wrtmap
    1*trace
    1*bye
Control mode.
```

Chapter 6

Catalog of Beam-Line Elements

The beam-line elements and their type code mnemonics, as currently available in MaryLie 3.0, are listed below. Also listed are the subsections that describe them in detail.

Type Code	Element	Subsection
drft	Drift Space	6.1
	Dipole Bend Magnets	
nbnd	a) Normal Entry Bending Magnet, with or without Fringe Fields.	6.2
pbnd	b) Parallel Faced Bending Magnet, with Fringe Fields and equal entry and exit angles.	6.3
gbnd	c) General Bending Magnet.	6.4
prot	d) Used for Leading and Trailing Pole Face Rotations.	6.5
gbdy	e) Used for the Body of a General Bending Magnet.	6.6
frng	f) Used for Hard-Edge Dipole Fringe Fields.	6.7
cfbd	g) Combined Function Bend.	6.8
quad	Magnetic Quadrupole.	6.9
sext	Magnetic Sextupole.	6.10
octm	Magnetic Octupole.	6.11
octe	Electric Octupole.	6.12

Type Code	Element	Subsection
srfc	Short RF Cavity.	6.13
arot	Axial Rotation.	6.14
twsm	Linear matrix transformation specified in terms of twiss parameters.	6.15
thlm	"Thin lens" approximation to low order multipoles.	6.16
cplm	"Compressed" approximation to low order multipoles.	6.17
jmap	Map with matrix part J.	6.18
r****	Random counterpart of the element with type-code mnemonic ****.	6.19
usr1 ⋮ usr10	User Specified Subroutines that act on phase space data.	6.20
usr11 ⋮ usr20	User Specified Subroutines that produce or act on maps.	6.21
dism	Dispersion matrix.	6.22
sol	Solenoid.	6.23
cfqd	Combined Function Magnetic Quadrupole.	6.24
mark	Marker.	6.25
dp	Data Point.	6.26
recm	REC Quadrupole Multiplet.	6.27
spce	Space for accounting purposes.	6.28
cfrn	Change or write out values of fringe field parameters for combined function dipole.	6.29
rmap	Random map.	6.30
arc	Angular space for accounting purposes.	6.31

Note that the type codes are given in lower case. If entries are made in upper case, they are automatically converted to lower case by PREP and MaryLie.

The purpose of this section is to outline the use of these elements by MaryLie, and to describe the parameters required to specify these elements in the #menu component of the Master Input File.

Calculations of f_2 (or equivalently the matrix R), f_3 , and f_4 for the beam-line elements listed in this section have been carried out by Diamond, Douglas, Dragt, Forest, Healy, Neri, Ryne, and van Zeijts. Expressions for the coefficients of the various polynomials may be found in the papers listed in Chapter 11. It is expected that other elements, some of which

may be described through 4th and 5th order in terms of polynomials f_5 and f_6, will be added to MaryLie's library in the future.

6.1 Drift

Type Code: drft

Required Parameters:

1. Length of drift (m).

Example:

```
dr5   drft
0.3D+01
```

The above specifies a drift space with user given name *dr5*, and length 3 meters. For an example of a MaryLie run that computes and displays the transfer map for a drift, see section 3.11.1.

6.2 Normal Entry and Exit (Sector) Bending Magnet

Type Code: nbnd or sbnd

Required Parameters:

1. Bend angle in degrees.
2. Gap size (pole to pole distance) in meters.
3. Normalized field integral (dimensionless).
4. Magnetic field, B (Tesla).
5. LFRN
   = 0 for no fringe field on leading edge (entry) of dipole.
   = 1 for hard-edge fringe field on leading edge of dipole.
6. TFRN
   = 0 for no fringe field on trailing edge (exit) of dipole.
   = 1 for hard-edge fringe field on trailing edge of dipole.

Example:

```
bend   nbnd
36., .03, .5, 1.2, 1, 1
```

This specifies a normal entry bend with user given name *bend* . It subtends an angle of 36°, has a field of 1.2 Tesla, and has leading and trailing hard-edge fringe fields. The full gap size (pole to pole distance) is .03 meters, and the normalized field integral is .5.

Vector potential: $A_\phi = -\dfrac{\rho B}{2}$.

Description:

The type code *nbnd* produces a normal entry (sector) dipole that bends to the right. See figures 6.2.1 through 6.2.3 and Exhibit 6.2.1. Normal entry dipoles that bend in other directions may be obtained by using *nbnd* in conjunction with *arot*. See figures 6.2.4 through 6.2.7 and Exhibits 6.2.2 through 6.2.6 for various kinds of normal entry bends and their associated coordinate systems. For the benefit of those accustomed to referring to such magnets as *sector bends*, MaryLie also recognizes the equivalent type code mnemonic *sbnd*.

For analytic calculations, trajectories in figure 6.2.1 can be conveniently computed using the right handed cylindrical triad $\mathbf{e}_\rho, \mathbf{e}_y, \mathbf{e}_\phi$. Note that this choice differs from the more familiar right-handed triad $\mathbf{e}_\rho, \mathbf{e}_\phi, \mathbf{e}_z$. For example, one finds the results

$$
\begin{aligned}
B_\rho &= (\nabla \times \mathbf{A})_\rho &&= \frac{\partial A_\phi}{\partial y} - \frac{1}{\rho}\frac{\partial A_y}{\partial \phi} = 0, \\
B_y &= (\nabla \times \mathbf{A})_y &&= \frac{1}{\rho}\frac{\partial A_\rho}{\partial \phi} - \frac{1}{\rho}\frac{\partial}{\partial \rho}(\rho A_\phi) \\
&= \frac{1}{\rho}\frac{\partial}{\partial \rho}(\frac{\rho^2 B}{2}) &&= B, \\
B_\phi &= (\nabla \times \mathbf{A})_\phi &&= \frac{\partial A_y}{\partial \rho} - \frac{\partial A_\rho}{\partial y} = 0.
\end{aligned}
$$

Leading and trailing fringe fields, when employed, are treated in the finite gap approximation. In the absence of better information, the normalized field integral parameter should be set equal to .5. For a further discussion of the finite gap approximation, see section 6.7.

Figure 6.2.1: Geometry of a Normal Entry Bend, top view. A positive particle is bent to the right (the $-x$ direction). The bend angle is θ. The bending radius ρ, path length s, and cord length d are given by the relations

$$\rho = \frac{B\rho}{B},\ s = \rho\theta,\ d = 2\rho\sin(\frac{\theta}{2}).$$

Figure 6.2.2: Geometry of a Normal Entry Bend, end view. The magnetic field B is along the $+y$ direction, and the magnetic force F on a positive particle is in the $-x$ direction. If the magnetic field is viewed as being produced by poles, then the south pole is above and the north pole is below as shown. (Recall that magnetic field lines come out of north poles and go into south poles.)

Figure 6.2.3: Illustrative off energy orbits (for a positive particle) in a Normal Entry Bend (top view). A normal entry particle exits with $x > 0$ if the energy exceeds the design energy ($P_\tau < 0$), and exits with $x < 0$ if the energy is below the design energy ($P_\tau > 0$). See Exhibit 6.2.1.

Figure 6.2.4: Geometry of a Normal Entry Bend that bends a positive particle to the left (top view). It is obtained by using the type code *nbnd* preceded and followed by axial rotations of $+180°$ and $-180°$, respectively. See Exhibit 6.2.2. Alternatively, the same effect can be achieved by using *nbnd* alone with $\theta < 0$ and $B < 0$. See Exhibit 6.2.3.

Figure 6.2.5: Geometry of a vertical Normal Entry Bend (side view) that bends a positive particle up. It is obtained by using the type code *nbnd* preceded and followed by axial rotations of $+90°$ and $-90°$, respectively. See Exhibit 6.2.4.

Figure 6.2.6: Geometry of a vertical Normal Entry Bend (side view) that bends a positive particle down. It is obtained by using the type code *nbnd* preceded and followed by axial rotations of $-90°$ and $+90°$, respectively. See Exhibit 6.2.5.

Figure 6.2.7: Geometry (side view) of a vertical upward "step" (for positive particles) employing two Normal Entry Bends separated by a Drift. See Exhibit 6.2.6.

```
#comment
 Exhibit 6.2.1.
 This is a MARYLIE run illustrating the sign conventions for a
 Normal Entry Bend rbend that bends to the right.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 end      end
#labor
    1*fileout
    1*rbend
    1*matout
    1*raysin
    1*trace
    1*end

matrix for map is :

 8.09017E-01  2.39083E+00  0.00000E+00  0.00000E+00  0.00000E+00 -9.22806E-01
-1.44507E-01  8.09017E-01  0.00000E+00  0.00000E+00  0.00000E+00 -6.98239E-01
 0.00000E+00  0.00000E+00  1.00000E+00  2.55570E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 6.98239E-01  9.22806E-01  0.00000E+00  0.00000E+00  1.00000E+00  8.18104E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (contents of file 13):

```
 0 0 0 0 0 -.01
 0 0 0 0 0 +.01
```

Final conditions for ray trace (contents of file 14):

```
 9.11465E-03  6.97045E-03  0.00000E+00  0.00000E+00 -8.00914E-03 -1.00000E-02
-9.34490E-03 -6.99462E-03  0.00000E+00  0.00000E+00  8.35945E-03  1.00000E-02
```

```
#comment
 Exhibit 6.2.2.
 This is a MARYLIE run illustrating the sign conventions for a
 Normal Entry Bend lbnd that bends to the left.
 The element lbnd is achieved using nbnd with preceding and
 following rotations of +180 and -180 degrees, respectively.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 rot+180  arot
   180.000000000000
 rot-180  arot
  -180.000000000000
 end      end
#lines
 lbend
     1*rot+180     1*rbend       1*rot-180
#lumps
#loops
#labor
    1*fileout
    1*lbend
    1*matout
    1*raysin
    1*trace
    1*end

matrix for map is :

 8.09017E-01  2.39083E+00  8.41639E-18  7.26563E-18  0.00000E+00  9.22806E-01
-1.44507E-01  8.09017E-01  6.36824E-18  8.41639E-18  0.00000E+00  6.98239E-01
 8.41639E-18  7.26563E-18  1.00000E+00  2.55570E+00  0.00000E+00 -4.06669E-17
 6.36824E-18  8.41639E-18 -2.80640E-34  1.00000E+00  0.00000E+00 -3.07705E-17
-6.98239E-01 -9.22806E-01  3.07705E-17  4.06669E-17  1.00000E+00  8.18104E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (contents of file 13):

```
 0 0 0 0 0 -.01
 0 0 0 0 0 +.01
```

Final conditions for ray trace (contents of file 14):

```
-9.11465E-03 -6.97045E-03  4.01671E-19  3.07179E-19 -8.00914E-03 -1.00000E-02
 9.34490E-03  6.99462E-03 -4.11818E-19 -3.08244E-19  8.35945E-03  1.00000E-02
```

```
#comment
Exhibit 6.2.3.
This is a MARYLIE run illustrating that a Normal Entry Bend (lbnd or
negbend) that bends to the left can be achieved in two ways: lbnd uses
nbnd with preceding and following rotations of +180 and -180 degrees,
respectively; negbend uses nbnd with a negative bend angle and a negative
field strength.  The map (lbnd inverse)*negbend is computed, and shown to
be (to machine precision) the identity map.
The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 rbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 negbend  nbnd
  -36.0000000000000      0.000000000000000E+00  0.500000000000000
  -1.20000000000000       1.00000000000000       1.00000000000000
 rot+180  arot
   180.000000000000
 rot-180  arot
  -180.000000000000
 zero     zer
  0.000000000000000E+00  1.000000000000000E-15  0.000000000000000E+00
 inv      inv
 end      end
#lines
 lbend
     1*rot+180     1*rbend       1*rot-180
#lumps
#loops
#labor
    1*fileout
    1*zero
    1*lbend
    1*inv
    1*negbend
    1*mapout
    1*end

matrix for map is :

 1.00000E+00  0.00000E+00 -8.41639E-18  1.42441E-17  0.00000E+00 -4.16334E-17
 0.00000E+00  1.00000E+00 -6.36824E-18  7.85893E-18  0.00000E+00  0.00000E+00
-7.85893E-18  1.42441E-17  1.00000E+00  0.00000E+00  0.00000E+00  3.79734E-17
```

```
-6.36824E-18  8.41639E-18  2.80640E-34  1.00000E+00  0.00000E+00  3.07705E-17
 0.00000E+00  4.16334E-17 -3.07705E-17  3.79734E-17  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

```
#comment
 Exhibit 6.2.4.
 This is a MARYLIE run illustrating the sign conventions for a
 Normal Entry Bend upbend that bends up. The element upbend is
 achieved using nbnd with preceding and following rotations of
 +90 and -90 degrees, respectively.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 rot+90   arot
   90.0000000000000
 rot-90   arot
  -90.0000000000000
 end      end
#lines
 upbend
     1*rot+90      1*rbend       1*rot-90
#lumps
#loops
#labor
    1*fileout
    1*upbend
    1*matout
    1*raysin
    1*trace
    1*end

matrix for map is :

 1.00000E+00  2.55570E+00 -4.20819E-18 -3.63281E-18  0.00000E+00  2.03335E-17
-7.01601E-35  1.00000E+00 -3.18412E-18 -4.20819E-18  0.00000E+00  1.53853E-17
-4.20819E-18 -3.63281E-18  8.09017E-01  2.39083E+00  0.00000E+00  9.22806E-01
-3.18412E-18 -4.20819E-18 -1.44507E-01  8.09017E-01  0.00000E+00  6.98239E-01
-1.53853E-17 -2.03335E-17 -6.98239E-01 -9.22806E-01  1.00000E+00  8.18104E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (contents of file 13):

```
 .01 0 0 0 0  0
 0   0 0 0 0 -.01
 0   0 0 0 0 +.01
```

Final conditions for ray trace (contents of file 14):

```
 1.00000E-02 -4.36715E-09  2.34769E-06  1.77635E-06  8.58309E-06  0.00000E+00
-2.00836E-19 -1.53590E-19 -9.11465E-03 -6.97045E-03 -8.00914E-03 -1.00000E-02
 2.05909E-19  1.54122E-19  9.34490E-03  6.99462E-03  8.35945E-03  1.00000E-02
```

```
#comment
 Exhibit 6.2.5.
 This is a MARYLIE run illustrating the sign conventions for a
 Normal Entry Bend dwnbend that bends down. The element dwnbend
 is achieved using nbnd with preceding and following rotations of
 -90 and +90 degrees, respectively.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 rot+90   arot
   90.0000000000000
 rot-90   arot
  -90.0000000000000
 end      end
#lines
 dwnbend
     1*rot-90      1*rbend       1*rot+90
#lumps
#loops
#labor
    1*fileout
    1*dwnbend
    1*matout
    1*raysin
    1*trace
    1*end

matrix for map is :

 1.00000E+00  2.55570E+00  4.20819E-18  3.63281E-18  0.00000E+00  2.03335E-17
-7.01601E-35  1.00000E+00  3.18412E-18  4.20819E-18  0.00000E+00  1.53853E-17
 4.20819E-18  3.63281E-18  8.09017E-01  2.39083E+00  0.00000E+00 -9.22806E-01
 3.18412E-18  4.20819E-18 -1.44507E-01  8.09017E-01  0.00000E+00 -6.98239E-01
-1.53853E-17 -2.03335E-17  6.98239E-01  9.22806E-01  1.00000E+00  8.18104E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (contents of file 13):

```
 .01 0 0 0 0  0
 0   0 0 0 0 -.01
 0   0 0 0 0 +.01
```

Final conditions for ray trace (contents of file 14):

```
 1.00000E-02 -4.36715E-09 -2.34769E-06 -1.77635E-06  8.58309E-06  0.00000E+00
-2.00836E-19 -1.53590E-19  9.11465E-03  6.97045E-03 -8.00914E-03 -1.00000E-02
 2.05909E-19  1.54122E-19 -9.34490E-03 -6.99462E-03  8.35945E-03  1.00000E-02
```

```
#comment
 Exhibit 6.2.6.
 This is a MARYLIE run illustrating the sign conventions for a
 vertical "upward" step composed of normal entry up and down
 bends separated by a drift.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 rot+90   arot
   90.0000000000000
 rot-90   arot
  -90.0000000000000
 dr       drft
   2.00000000000000
 end      end
#lines
 upbend
     1*rot+90      1*rbend       1*rot-90
 dwnbend
     1*rot-90      1*rbend       1*rot+90
 upstep
     1*upbend      1*dr          1*dwnbend
#lumps
#loops
#labor
    1*fileout
    1*upstep
    1*matout
    1*raysin
    1*trace
    1*end

matrix for map is :

 1.00000E+00  7.11140E+00 -1.70508E-17 -2.99500E-18  0.00000E+00  1.23054E-16
```

```
-1.87396E-34  1.00000E+00 -2.13648E-18  1.19610E-17  0.00000E+00  4.10938E-17
-1.19610E-17  2.99500E-18  7.52000E-02  5.17746E+00  0.00000E+00  2.62291E+00
 2.13648E-18  1.70508E-17 -1.92053E-01  7.52000E-02  0.00000E+00 -4.68504E-01
-4.10938E-17 -1.23054E-16 -4.68504E-01  2.62291E+00  1.00000E+00  4.72225E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (file 13):

```
 .01 0 0 0 0  0
 0   0 0 0 0 -.01
 0   0 0 0 0 +.01
```

Final conditions for ray trace (file 14):

```
 9.99997E-03 -1.16646E-08  6.67276E-06 -1.19190E-06  2.29253E-05  0.00000E+00
-1.21237E-18 -4.12806E-19 -2.59374E-02  4.62526E-03 -4.63138E-02 -1.00000E-02
 1.24919E-18  4.08869E-19  2.65281E-02 -4.74673E-03  4.81669E-02  1.00000E-02
```

Note that both the height and direction (P_y) of the final rays depend on P_τ.

6.3 Parallel Faced (Rectangular) Bending Magnet (Symmetric)

Type Code: pbnd or rbnd

Required Parameters:

1. Bend angle in degrees.
2. Gap size (pole to pole distance) in meters.
3. Normalized field integral (dimensionless).
4. Magnetic field, B (Tesla).

Example:

```
bend    pbnd
36. , .03 , .5 , 1.2
```

This specifies a parallel faced bend with user given name *bend* . It subtends an angle of 36° and has a field of 1.2 Tesla. The entry and exit angles are equal, and have the value 18 (=36/2) degrees. See figure 6.3.1. The full gap size (pole to pole distance) is .03 meters, and the normalized field integral is .5. Leading and trailing fringe fields are treated in the finite gap approximation. In the absence of better information, the normalized field integral parameter should be set equal to .5. For a further discussion of the finite gap approximation, see section 6.7. Should the user wish to specify his or her own fringe fields, this can be done by employing the type codes *prot* and *gbdy*. See sections 6.5 and 6.6.

Vector Potential (in global Cartesian coordinates): $A_z = -xB$.

Description:

Only symmetric bends (equal entry and exit angles) are produced by this type code. See figure 6.3.1. For the benefit of those accustomed to referring to such magnets as *rectangular bends*, MaryLie also recognizes the equivalent type code mnemonic *rbnd*. An asymmetric parallel faced bend (entry and exit angles unequal) may be produced using the type code *gbnd* for the general bending magnet. See section 6.4.

As in the case of the Normal Entry Bend, a positive particle is bent to the right (the $-x$ direction). Particles may also be bent to the left, up, or down by using parallel faced bends in conjunction with appropriate axial rotations. The procedure is identical to that for Normal Entry Bends. See section 6.2 and Exhibit 6.3.1.

Figure 6.3.1: Geometry of a Symmetric Parallel Faced Bend, top view. A positive particle is bent to the right (the $-x$ direction). The bend angle is θ, and the entry and exit angles are $\theta/2$. The bending radius ρ, path length l, and cord length d are given by the same relations as for the Normal Entry Bend. See figure 6.2.1.

```
#comment
 Exhibit 6.3.1.
 This is a MARYLIE run illustrating the sign conventions for a
 vertical "upward" step composed of parallel faced  bends
 separated by a drift.  Comparison is also made between between
 two ways of forming parallel faced bends that bend to the left
 showing that the results are the same.  See section 6.2 for an
 analogous discussion in the case of normal entry bends.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero    zer
   0.000000000000000E+00  1.000000000000000E-15  0.000000000000000E+00
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 negbend  pbnd
  -36.0000000000000      0.000000000000000E+00  0.500000000000000
  -1.20000000000000
 rot+90   arot
   90.0000000000000
 rot-90   arot
  -90.0000000000000
 rot+180  arot
   180.000000000000
 rot-180  arot
  -180.000000000000
 dr       drft
   2.00000000000000
 clear    iden
 inv      inv
 end      end
#lines
 lbend
     1*rot+180     1*rbend       1*rot-180
```

```
 upbend
     1*rot+90      1*rbend       1*rot-90
 dwnbend
     1*rot-90      1*rbend       1*rot+90
 upstep
     1*upbend      1*dr          1*dwnbend
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*upstep
    1*matout
    1*raysin
    1*trace
    1*clear
    1*lbend
    1*inv
    1*negbend
    1*mapout
    1*end

matrix for map is :

 3.84086E-02  5.33464E+00  1.21913E-17 -3.53420E-19  0.00000E+00  9.87692E-17
-1.87178E-01  3.84086E-02 -2.19753E-18 -1.55670E-17  0.00000E+00  1.14797E-17
 1.55670E-17  3.53420E-19  1.00000E+00  6.78166E+00  0.00000E+00  3.38952E+00
 2.19753E-18 -1.21913E-17 -1.87720E-34  1.00000E+00  0.00000E+00  2.52948E-34
-1.14797E-17 -9.87692E-17  0.00000E+00  3.38952E+00  1.00000E+00  5.07505E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (contents of file 13):

```
 .01 0 0 0 0  .0
 .0  0 0 0 0 -.01
 .0  0 0 0 0 +.01
```

Final conditions for ray trace (contents of file 14):

```
 3.84075E-04 -1.87177E-03 -8.13872E-07 -5.69185E-07  1.86463E-05  0.00000E+00
-9.79363E-19 -1.18902E-19 -3.33628E-02 -6.93889E-24 -4.96961E-02 -1.00000E-02
 9.95846E-19  1.10489E-19  3.44479E-02  6.93889E-24  5.18501E-02  1.00000E-02
```

Note that the height but not the direction (P_y) of the final rays depends on P_τ. This is an advantage of using parallel faced bends. Compare with Exhibit 6.2.6.

```
matrix for map is :

 1.00000E+00  0.00000E+00  6.11777E-18 -2.87754E-17  0.00000E+00 -1.38778E-17
 2.78597E-34  1.00000E+00  6.32188E-18 -8.99678E-18  0.00000E+00  1.38778E-17
 8.99678E-18 -2.87754E-17  1.00000E+00  0.00000E+00  0.00000E+00  5.45780E-17
 6.32188E-18 -6.11777E-18  0.00000E+00  1.00000E+00  0.00000E+00  3.29078E-17
 1.38778E-17  1.38778E-17 -3.29078E-17  5.45780E-17  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.4 General Bending Magnet

Type Code: gbnd

Required Parameters:

1. Bend angle in degrees.
2. Entry angle in degrees.
3. Exit angle in degrees.
4. Gap size (pole to pole distance) in meters.
5. Normalized field integral (dimensionless).
6. Magnetic field, B (Tesla).

Example:

```
oddbend    gbnd
25.2 , 8.0 , 5.0 , .03 , .5 , .85
```

This specifies a general bending magnet with user given name *oddbend*. It produces a bend angle of 25.2°, and has entry and exit angles of 8° and 5° from the normal, respectively. The magnet has a strength of .85 Tesla. The full gap size (pole to pole distance) is .03 meters, and the normalized field integral is .5. Leading and trailing fringe fields are treated in the finite gap approximation. In the absence of better information, the normalized field integral parameter should be set equal to .5. For a further discussion of the finite gap approximation, see section 6.7. Should the user wish to specify his or her own fringe fields, this can be done by employing the type codes *prot* and *gbdy*. See sections 6.5 and 6.6.

Description:

A general bending magnet is a dipole magnet with arbitrary pole face geometry. The faces need not be perpendicular to the entry and exit design trajectory, as with a normal-entry magnet, nor be parallel to each other, as with the parallel-faced magnet.

The three required geometric parameters are illustrated in figure 6.4.1. The bend angle θ is the angle subtended by the design trajectory from the entry face to the exit face. The entry angle ϕ is the angle from the design trajectory to the normal to the entry face, measured positive in the clockwise direction. The exit angle ψ is the angle from the normal to the exit face to the design trajectory, again measured positive in the clockwise direction.

The user will note that both the normal entry and parallel-faced magnets are special cases of the general bending magnet, and the latter may be used as a substitute. See Exhibit 6.4.1.

The type code *gbnd* may also be used to model a wiggler. See Exhibit 6.4.2.

Figure 6.4.1: Geometry of a General Bending Magnet, top view. The entry angle ϕ is the angle from the design trajectory to the normal to the entry face, measured positive in the clockwise direction. By contrast, the exit angle ψ is the angle from the normal to the exit face to the design trajectory, again measured positive in the clockwise direction.

```
#comment
 Exhibit 6.4.1.
 This is a MARYLIE run illustrating the sign conventions for a
 General Bending Magnet.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-15  0.000000000000000E+00
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rbend    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 gbend    gbnd
   36.0000000000000       18.0000000000000       18.0000000000000
  0.000000000000000E+00  0.500000000000000       1.20000000000000
 negbend  gbnd
  -36.0000000000000      -18.0000000000000      -18.0000000000000
  0.000000000000000E+00  0.500000000000000      -1.20000000000000
 rot+180  arot
   180.000000000000
 rot-180  arot
  -180.000000000000
 inv      inv
 clear    iden
 end      end
#lines
 lbend
     1*rot+180     1*rbend       1*rot-180
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*rbend
    1*inv
```

```
    1*gbend
    1*mapout
    1*clear
    1*lbend
    1*inv
    1*negbend
    1*mapout
    1*end

matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.38778E-17
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.38778E-17
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
-1.38778E-17 -1.38778E-17  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00  0.00000E+00  6.11777E-18 -2.87754E-17  0.00000E+00 -1.38778E-17
 2.78597E-34  1.00000E+00  6.32188E-18 -8.99678E-18  0.00000E+00  1.38778E-17
 8.99678E-18 -2.87754E-17  1.00000E+00  0.00000E+00  0.00000E+00  5.45780E-17
 6.32188E-18 -6.11777E-18  0.00000E+00  1.00000E+00  0.00000E+00  3.29078E-17
 1.38778E-17  1.38778E-17 -3.29078E-17  5.45780E-17  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :




#comment
 Exhibit 6.4.2.
 This is a MaryLie run illustrating the use of the type code gbnd
 to model a wiggler.  The wiggler consists of the following:
 a) entry = a parallel faced bend with zero entry angle and
    nonzero exit angle;
 b) 25 pairs of parallel faced left and right bends;
 c) a final parallel faced left bend;
 d) exit = a parallel faced bend with nonzero entry angle and
    zero exit angle.
 With this construction, the wiggler produces no net bending, and
 a beam that enters normally at x=y=0 also exits normally at
 x=y=0.
#beam
  4.78740000023600
  2807.64374771200
  1.00000000000000
  1.00000000000000
#menu
```

```
 zer      zer
  0.000000000000000E+00  1.000000000000000E-10  0.000000000000000E+00
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 entry    gbnd
   5.00000000000000      0.000000000000000E+00   5.00000000000000
  0.000000000000000E+00  0.500000000000000       4.00000000000000
 exit     gbnd
   5.00000000000000       5.00000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.500000000000000       4.00000000000000
 pbnd     pbnd
   10.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000
 rot+180  arot
   180.000000000000
 rot-180  arot
  -180.000000000000
#lines
 rbend
     1*pbnd
 lbend
     1*rot+180     1*pbnd        1*rot-180
 pair
     1*lbend       1*rbend
 wig
     1*entry      25*pair        1*lbend       1*exit
#lumps
#loops
#labor
    1*fileout
    1*zer
    1*wig
    1*mapout
    1*fin

matrix for map is :

 1.00000E+00  1.08899E+01  1.19701E-16  6.40843E-16  0.00000E+00 -2.60209E-18
-2.92881E-32  1.00000E+00  1.64018E-17  1.18788E-16  0.00000E+00  0.00000E+00
 1.18788E-16  6.40843E-16 -9.47358E-01  3.81252E-01  0.00000E+00  1.42171E-16
 1.64018E-17  1.19701E-16 -2.68885E-01 -9.47358E-01  0.00000E+00  1.96305E-17
 0.00000E+00 -1.11022E-15 -1.96305E-17 -1.42171E-16  1.00000E+00  2.76593E-02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 53)=f( 02 00 01 )= -5.4866394975113
 f( 54)=f( 01 20 00 )=-1.33753901565392E-03
 f( 55)=f( 01 11 00 )= 9.42507217935450E-03
```

```
 f( 58)=f( 01 02 00 )= 1.89650071918470E-03
 f( 67)=f( 00 20 01 )= -3.8486239216226
 f( 70)=f( 00 11 01 )= 1.05453134698232E-03
 f( 76)=f( 00 02 01 )= -5.4606912148188
 f( 83)=f( 00 00 03 )=-1.38930890666728E-02
 f(140)=f( 04 00 00 )= -1.3821588366485
 f(145)=f( 02 20 00 )= -5.7367553349907
 f(146)=f( 02 11 00 )= 0.10653345142166
 f(149)=f( 02 02 00 )= -8.5095112889952
 f(154)=f( 02 00 02 )= -5.5496344962907
 f(158)=f( 01 20 01 )= 3.22126895230156E-02
 f(161)=f( 01 11 01 )=-0.22698920950513
 f(167)=f( 01 02 01 )=-4.56744723946634E-02
 f(175)=f( 00 40 00 )=-0.26292566764119
 f(176)=f( 00 31 00 )=-1.17384240876864E-03
 f(179)=f( 00 22 00 )=-0.73832467046815
 f(184)=f( 00 20 02 )= -3.9278645561837
 f(185)=f( 00 13 00 )=-2.38314000564671E-02
 f(190)=f( 00 11 02 )=-2.70099395539289E-02
 f(195)=f( 00 04 00 )=-0.48368417698322
 f(200)=f( 00 02 02 )= -5.4741675233709
 f(209)=f( 00 00 04 )=-1.39728446835345E-02

end of MARYLIE run
```

6.5 Plane Rotation

Type Code: prot

Required Parameters:

1. Rotation angle in degrees.
2. KIND
   = 1 for transition from the normal reference plane to a rotated reference plane.
   = 2 for transition from a rotated reference plane to the normal reference plane.

Example:

```
planerot    prot
    20   ,   1
```

This specifies a map with the user given name *planerot*. It is the map for the transition from a normal reference plane to a new reference plane rotated by 20° from the normal reference plane.

Description:

This type code is used internally by MaryLie, in the generation of transfer maps for parallel faced bending magnets and general bending magnets, to achieve pole face rotation. It is also available directly to the user under the type code *prot*. By itself, it describes the transition between the normal reference plane and a rotated reference plane, or between a rotated reference plane and the normal reference plane. See figures 6.5.1 and 6.5.2. For a description of how *prot* is used in connection with *gbdy* to produce maps for bending magnets with modified fringe fields, see section 6.6.

Figure 6.5.1: Geometry for a transition from the normal reference plane to a rotated reference plane. The x, y axes define the normal reference plane, and the $\overline{x}, \overline{y}$ axes define the rotated reference plane. The x and $\overline{x}$ axes are in the plane of the paper. The rotation angle ϕ from the x to the $\overline{x}$ axis is measured positive in the clockwise direction.

Figure 6.5.2: Geometry for a transition from a rotated reference plane to the normal reference plane. The x, y axes define the normal reference plane, and the $\overline{x}, \overline{y}$ define the rotated reference plane. The x and $\overline{x}$ axes are in the plane of the paper. The rotation angle ψ from the rotated reference plane to the normal reference plane is measured positive in the clockwise direction.

6.6 Body of a General Bending Magnet

Type Code: gbdy

Required Parameters:

1. Bend angle in degrees.
2. Entry angle in degrees.
3. Exit angle in degrees.
4. Magnetic field, B (Tesla).

Example:

```
gbod    gbdy
36. , 0. , 18. , .85
```

This specifies a general bending magnet with user given name *gbod*. It has a bend angle of 36°, normal entry, and an exit angle of 18° from the normal to the poleface.

Description:

The body of the general bending magnet is a portion of the general bending magnet described in section 6.4. The transfer map generated is that of the magnet itself, without fringe fields and without pole-face rotations at the entry and exit. The purpose of this map is to make it possible for the user to supply his or her own fringe field maps for a general bending magnet. Such maps would be inserted between a separately specified (entry) pole-face rotation (type code *prot*), and the general bend body, and between the general bend body and the (exit) pole-face rotation. See Exhibits 6.6.1 and 6.6.2.

The *gbdy* element can also be used to express a *pbnd* (or, for that matter, a *gbnd*) element in terms of an *nbnd* element and various leading and trailing maps. This fact makes it possible to "slice" a *pbnd* element since it is easy to slice an *nbnd* element. See Exhibit 6.6.3.

The geometry for the *gbdy* type code is identical to that of the general bending magnet. See figure 6.6.1.

Figure 6.6.1: Geometry for the type code *gbdy*, top view. The entry angle ϕ is the angle from the design trajectory to the normal to the entry face, measured positive in the clockwise direction. By contrast, the exit angle ψ is the angle from the normal to the exit face to the design trajectory, again measured positive in the clockwise direction.

```
#comment
 Exhibit 6.6.1.
 This is a MARYLIE run illustrating the use of gbdy in connection
 with prot and frng.
 It is shown that the map corresponding to pbnd is of the form

   pbnd = prot * frng * gbdy * frng * prot

 for suitable choices of the parameters associated with prot, frng, and gbdy.
 Thus, if a user wishes to employ his/her own fringe fields for a parallel
 faced bend instead of those provided by MARYLIE, it is only necessary to
 replace the maps associated with frng by user supplied maps.  Also, if the
 user wishes to make a finite gap correction, this can be done by
 suitably specifying the parameters associated with frng.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
   0.000000000000000E+00  1.000000000000000E-15  0.000000000000000E+00
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 rpbnd    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 gbod     gbdy
   36.0000000000000       18.0000000000000       18.0000000000000
   1.20000000000000
 inprot   prot
   18.0000000000000       1.00000000000000
 outprot  prot
   18.0000000000000       2.00000000000000
 infrng   frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       1.00000000000000
 outfrng  frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       2.00000000000000
 inv      inv
 clear    iden
 end      end
#lines
 conbend
     1*inprot      1*infrng      1*gbod        1*outfrng     1*outprot
```

```
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*rpbnd
    1*inv
    1*conbend
    1*mapout
    1*end

matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.31986E-17
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  6.93889E-18
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
-1.38778E-17  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  5.09434E-18
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

```
#comment
 Exhibit 6.6.2.
 This is a MARYLIE run illustrating the use of gbdy in connection
 with prot and frng.  It will be shown that the map corresponding
 to gbnd is of the form

   gbnd = prot * frng * gbdy * frng * prot

 for suitable choices of the parameters associated with prot, frng, and gbdy.
 Thus, if a user wishes to employ his/her own fringe fields for a general
 bend instead of those provided by MARYLIE, it is only necessary to
 replace the maps associated with frng by user supplied maps.
 The parameters used in gbnd are those of section 6.4.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-15  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 oddbend  gbnd
   25.2000000000000       8.00000000000000       5.00000000000000
  0.000000000000000E+00  0.500000000000000      0.850000000000000
 gbod     gbdy
   25.2000000000000       8.00000000000000       5.00000000000000
  0.850000000000000
 inprot   prot
   8.00000000000000       1.00000000000000
 outprot  prot
   5.00000000000000       2.00000000000000
 infrng   frng
   8.00000000000000      0.000000000000000E+00  0.500000000000000
  0.850000000000000       1.00000000000000
 outfrng  frng
   5.00000000000000      0.000000000000000E+00  0.500000000000000
  0.850000000000000       2.00000000000000
 inv      inv
 clear    iden
 end      end
#lines
 conbend
     1*inprot      1*infrng      1*gbod        1*outfrng     1*outprot
#lumps
#loops
#labor
    1*fileout
```

```
    1*setzero
    1*oddbend
    1*inv
    1*conbend
    1*mapout
    1*end

matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.72812E-17
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  2.08167E-17
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.33681E-19  1.00000E+00  0.00000E+00  0.00000E+00
-5.20417E-18  1.38778E-17  0.00000E+00  0.00000E+00  1.00000E+00  1.79602E-18
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

```
#comment
 Exhibit 6.6.3.
 This is a MARYLIE run illustrating that the gbdy element can be used
 to represent a parallel faced (rectangular) bend in terms of a normal
 entry (sector) bend and various leading and trailing maps.  This
 representation makes it possible to "slice" a pbnd since it is
 easy to slice an nbnd.  The map for a pbnd in terms of an nbnd is
 of the form

 pbnd = inprot * infrng * ingbdy * nbnd * outgbdy * outfrng * outprot

 for a suitable choice of parameters.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 rbend    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 inv      inv
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 inprot   prot
   18.0000000000000       1.00000000000000
 outprot  prot
   18.0000000000000       2.00000000000000
 infrng   frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       1.00000000000000
 outfrng  frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       2.00000000000000
 ingbdy   gbdy
  0.000000000000000E+00   18.0000000000000      0.000000000000000E+00
   1.20000000000000
 outgbdy  gbdy
  0.000000000000000E+00  0.000000000000000E+00   18.0000000000000
   1.20000000000000
 sbend    nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000      0.000000000000000E+00  0.000000000000000E+00
 fin      end
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-14  0.000000000000000E+00
#lines
 product
     1*inprot      1*infrng      1*ingbdy      1*sbend       1*outgbdy  &
     1*outfrng     1*outprot
```

```
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*rbend
    1*inv
    1*product
    1*mapout
    1*fin

matrix for map is :

 1.00000E+00  8.77311E-16  0.00000E+00  0.00000E+00  0.00000E+00 -3.01236E-16
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.11022E-16
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 5.55112E-17  4.14635E-16  0.00000E+00  0.00000E+00  1.00000E+00  5.24524E-16
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.7 Fringe Fields for Bending Magnets (Hard Edge)

Type Code: frng

Required Parameters:

1. Entry or exit angle in degrees.
2. Gap size (pole to pole distance) in meters.
3. Normalized field integral (dimensionless).
4. Magnetic field inside magnet, B (Tesla).
5. IEDGE
   = 1 for leading (entry) edge.
   = 2 for trailing (exit) edge.

Example:

```
infringe    frng
  18  ,  .01  ,  .5  ,  1.2  ,  1
```

This specifies a map with the user given name *infringe*. The map describes the effect of a leading (entry) hard-edge dipole fringe field for the case in which the entry angle is 18°, the full gap (pole to pole distance) is .01 meters, the normalized field integral is .5, and the interior field of the dipole magnet is 1.2 Tesla.

Description:

The type code *frng* is used internally in MaryLie to supply hard-edge fringe fields for the type codes *nbnd*, *pbnd*, *gbnd*, and *cfbd*. See Exhibits 6.7.1, 6.6.1, and 6.6.2. Also see sections 6.8 and 6.29. The entry angle is the angle ϕ in figures 6.4.1 or 6.6.1, and the exit angle is the angle ψ. In the absence of better information, the normalized field integral parameter should be set equal to .5.

Cautionary Note:

The proper treatment of dipole fringe field effects requires the use of GENMAP or similar procedures. See section 1.4.1. Strictly speaking, there is no physically consistent theory of third and higher order aberrations in the hard-edge limit. Indeed, one f_4 term diverges in the hard-edge limit, and one depends on how the limit is taken. The coefficient of the monomial X^4 diverges and that of Y^3P_y is model dependent. For lack of better information, MaryLie sets these f_4 terms to zero, and the remaining f_3 and f_4 terms to their well-defined hard-edge limits. Also, it is not possible to construct, even in lowest order, a truly satisfactory analytic theory of fringe-field effects. For convenience in comparison of codes, the approximations used in *frng* for the matrix part of the map are the same as those for TRANSPORT.

```
#comment
 Exhibit 6.7.1.
 This is a MARYLIE run illustrating the use of frng in connection
 with nbnd.
 It is shown that the map corresponding to nbnd is of the form

   nbnd(with fringe) = frng * nbnd(without fringe) * frng

 for suitable choices of the parameters associated with frng.
 Thus, if a user wishes to employ his/her own fringe fields for a
 normal entry bend instead of those provided by MARYLIE, it is only
 necessary to replace the maps associated with frng by user supplied maps.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
 0.000000000000000E+00    1.000000000000000E-15  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 nbndwf   nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 nbndwof  nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000      0.000000000000000E+00  0.000000000000000E+00
 infrng   frng
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       1.00000000000000
 outfrng  frng
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       2.00000000000000
 inv      inv
 clear    iden
 end      end
#lines
 conbend
     1*infrng      1*nbndwof     1*outfrng
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*nbndwf
    1*inv
    1*conbend
    1*mapout
```

```
    1*end

matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  4.16334E-17
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00 -4.16334E-17  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.8 Combined Function Bending Magnet (Normal Entry and Exit)

Type Code: cfbd

Required Parameters:

1. Bend angle in degrees.
2. Magnetic field, B_0 (Tesla).
3. LFRN
   = 0 for no fringe field on leading edge (entry) of dipole.
   = 1 for hard-edge dipole fringe field on leading edge of dipole.
   = 2 for hard-edge quadrupole fringe field on leading edge of dipole.
   = 3 for both hard-edge dipole and quadrupole fringe fields on leading edge of dipole.
4. TFRN
   = 0 for no fringe field on trailing edge (exit) of dipole.
   = 1 for hard-edge dipole fringe field on trailing edge of dipole.
   = 2 for hard-edge quadrupole fringe field on trailing edge of dipole.
   = 3 for both hard-edge dipole and quadrupole fringe fields on trailing edge of dipole.
5. IOPT (controls meaning of the parameters P1. . . P6 below)
   = 1 for multipole expansion case.
   = 2 for mixed Taylor and multipole expansion case.
   = 3 for scaled mixed Taylor and multipole expansion case.
   = 4 for normal field specified by an index.

   Note: if IOPT is a two-digit integer of the form IJ, then J is interpreted as above, and I is used to control printing of parameters as follows:

   =0 to not echo back multipole and Taylor coefficients.
   =1 to write multipole and Taylor coefficients at the terminal.
   =2 to write multipole and Taylor coefficients on file 12.
   =3 to write multipole and Taylor coefficients at the terminal and on file 12.
6. IPSET
   = 0 to set all higher multipole or Taylor coefficients to zero.
   = J (with J an integer from 1 to 9) to take multipole or Taylor coefficient values from the parameter set associated with the type code *psj*. (See section 7.25.) In this case, depending on the value of IOPT, the parameters P1 . . . P6 from the parameter set associated with *psj* are used as follows:

When IOPT = 1

P1 = BQD	(Tesla/m)	(normal)
P2 = AQD	(Tesla/m)	(skew)
P3 = BSEX	(Tesla/m^2)	(normal)
P4 = ASEX	(Tesla/m^2)	(skew)
P5 = BOCT	(Tesla/m^3)	(normal)
P6 = AOCT	(Tesla/m^3)	(skew)

When IOPT = 2

P1 = tay1	(Tesla/m)
P2 = AQD	(Tesla/m)
P3 = tay2	(Tesla/m^2)
P4 = ASEX	(Tesla/m^2)
P5 = tay3	(Tesla/m^3)
P6 = AOCT	(Tesla/m^3)

When IOPT = 3

P1 = tay1/brho	(m^{-2})
P2 = AQD/brho	(m^{-2})
P3 = tay2/brho	(m^{-3})
P4 = ASEX/brho	(m^{-3})
P5 = tay3/brho	(m^{-4})
P6 = AOCT/brho	(m^{-4})

When IOPT = 4

P1 = index = n	
P2 = AQD	(Tesla/m)
P3 = 0	
P4 = ASEX	(Tesla/m^2)
P5 = 0	
P6 = AOCT	(Tesla/m^3)

Example:

```
slcbend    cfbd
  10.0 , .5 , 1 , 1 , 1 , 3
```

This specifies a combined function bending magnet with user given name *slcbend* . It produces a bend angle of 10°, and has a dipole strength of .5 Tesla. Hard-edge leading and trailing fringe fields are "turned on", and higher multipole coefficient values are taken from parameter set 3 and interpreted according to option 1. Suppose the parameter values in parameter set 3 have been set by invoking (in #labor) the following command:

```
slcbdpar     ps3
 .1 , 0 , .2 , 0 , .3 , 0
```

Then the skew multipoles all have zero strength, and the normal multipoles have quadrupole, sextupole, and octupole strengths of .1 T/m, .2 T/m^2, and .3 T/m^3, respectively.

Description:

A combined function dipole is a dipole with added quadrupole, sextupole, and octupole components. Thus, as illustrated in Exhibit 6.8.1, pure dipoles, pure quadrupoles, pure sextupoles, and pure octupoles are special cases of the combined function dipole. For the sign conventions for normal and skew multipole elements, see sections 6.9, 6.10, and 6.11. Note that the parameter set associated with a particular combined function bend must be invoked in #labor before the combined function bend itself is actually used.

The geometry of a combined function bending magnet is the same as that of a normal entry bend. We define a combined function dipole to be a dipole for which the design orbit within the dipole is a circular arc. We also assume that the pole faces are such that the design orbit is perpendicular to the pole-face planes both on entry and exit. See figure 6.8.1. Thus for our purposes, a combined function dipole is a generalization of a sector bend. The construction of such a bend generally requires the use of poles whose shapes are surfaces of revolution. See figure 6.8.2.

Note that our definition excludes cases of non-normal entry and/or exit, and cases where the poles of an ordinary dipole are simply inclined at an angle in order to introduce higher order multipole terms. In general, the treatment of such dipoles is considerably more difficult because the design orbit is no longer a circular arc. The reader who is contemplating the use of such dipoles is warned that they may not be properly treated (beyond first order) by existing accelerator design codes. She or he is advised to explore this point in some detail before trusting the results of such codes beyond first order.

Due to curvature effects, the magnetic field in a combined function dipole depends in a complicated way on the multipole coefficients. Suppose the position of a point near the design orbit is specified by deviation coordinates ξ, η defined by the relations

$$\rho = \rho_0 + \xi, \tag{6.8.1}$$

$$y = \eta. \tag{6.8.2}$$

Here ρ_0 is the radius of the design orbit. It is given by the relation

$$\rho_0 = \frac{\text{Brho}}{B_0}, \tag{6.8.3}$$

where B_0 is the dipole strength. Then it can be shown that the vector potential can be taken to have only a phi component with the expansion

$$\rho A_\phi = \sum_n U_n(\xi, \eta). \tag{6.8.4}$$

Here

$$U_n = P_n + S_n \tag{6.8.5}$$

Figure 6.8.1: Geometry of Combined Function Bending Magnet, top view.

with

$$U_0 = -\frac{B_0}{2}\rho_0^2, \tag{6.8.6}$$

$$U_1 = -B_0\rho_0\xi, \tag{6.8.7}$$

$$P_2 = -\frac{B_0}{2}\xi^2, \tag{6.8.8}$$

$$S_2 = -\frac{b_2\rho_0}{2}(\xi^2-\eta^2) + \frac{a_2\rho_0}{2}(2\xi\eta), \tag{6.8.9}$$

$$P_3 = -\frac{b_2}{8}\xi(\xi^2+\eta^2) + \frac{a_2}{8}\eta(\xi^2+\eta^2), \tag{6.8.10}$$

$$S_3 = -\frac{b_3\rho_0}{3}(\xi^3-3\xi\eta^2) + \frac{a_3\rho_0}{3}(3\xi^2\eta-\eta^3), \tag{6.8.11}$$

$$P_4 = \frac{b_2}{64\rho_0}(\xi^2+\eta^2)^2 - \left(\frac{b_3}{12}-\frac{b_2}{32\rho_0}\right)(\xi^4-\eta^4) + \left(\frac{a_3}{6}-\frac{a_2}{16\rho_0}\right)\xi\eta(\xi^2+\eta^2), \tag{6.8.12}$$

$$S_4 = -\frac{b_4\rho_0}{4}(\xi^4-6\xi^2\eta^2+\eta^4) + \frac{a_4\rho_0}{4}(4\xi^3\eta-4\xi\eta^3), \text{ etc.} \tag{6.8.13}$$

Correspondingly, the components of the magnetic field are given by the relations

$$B_\rho(\xi,\eta) \;=\; \frac{1}{\rho}\frac{\partial}{\partial y}(\rho A_\phi) = \frac{1}{(\rho_0+\xi)}\frac{\partial}{\partial\eta}(\rho A_\phi)$$

Figure 6.8.2: Combined Function Dipole whose poles are surfaces of revolution.

$$
\begin{aligned}
&= b_2\eta + a_2\xi + \eta^2(-a_3 + \frac{3a_2}{8\rho_0}) + \eta\xi(2b_3 - \frac{5b_2}{4\rho_0}) + \xi^2(a_3 - \frac{7a_2}{8\rho_0}) \\
&\quad +\eta^3(-b_4 - \frac{b_2}{16\rho_0^2} + \frac{b_3}{3\rho_0}) + \eta^2\xi(-3a_4 - \frac{9a_2}{16\rho_0^2} + \frac{3a_3}{2\rho_0}) \\
&\quad +\eta\xi^2(3b_4 + \frac{21b_2}{16\rho_0^2} - \frac{2b_3}{\rho_0}) + \xi^3(a_4 + \frac{13a_2}{16\rho_0^2} - \frac{5a_3}{6\rho_0}),
\end{aligned} \tag{6.8.14}
$$

$$
\begin{aligned}
B_y(\xi,\eta) &= -\frac{1}{\rho}\frac{\partial}{\partial\rho}(\rho A_\phi) = -\frac{1}{(\rho_0+\xi)}\frac{\partial}{\partial\xi}(\rho A_\phi) \\
&= B_0 - a_2\eta + b_2\xi + \eta^2(-b_3 + \frac{b_2}{8\rho_0}) + \eta\xi(-2a_3 + \frac{3a_2}{4\rho_0}) \\
&\quad +\xi^2(b_3 - \frac{5b_2}{8\rho_0}) + \eta^3(a_4 + \frac{a_2}{16\rho_0^2} - \frac{a_3}{6\rho_0}) + \eta^2\xi(-3b_4 - \frac{3b_2}{16\rho_0^2} + \frac{b_3}{\rho_0}) \\
&\quad +\eta\xi^2(-3a_4 - \frac{9a_2}{16\rho_0^2} + \frac{3a_3}{2\rho_0}) + \xi^3(b_4 + \frac{7b_2}{16\rho_0^2} - \frac{2b^3}{3\rho_0}),
\end{aligned} \tag{6.8.15}
$$

$$B_\phi = 0. \tag{6.8.16}$$

Here we have used the notation

$$
\begin{array}{lclcl}
b_2 & = & \text{BQD} & , & a_2 = \text{AQD} \\
b_3 & = & \text{BSEX} & , & a_3 = \text{ASEX} \\
b_4 & = & \text{BOCT} & , & a_4 = \text{AOCT.}
\end{array}
$$

For some applications it is convenient to specify the midplane field in terms of Taylor coefficients instead of the multipole coefficients b_2, b_3, and b_4. Suppose the midplane vertical field is expanded in the Taylor form

$$B_y(\xi,0) = B_0 + t_1\xi + t_2\xi^2 + t_3\xi^3 + \cdots. \tag{6.8.17}$$

Then, comparison with the results above gives the relations

$$t_1 = b_2, \tag{6.8.18}$$

$$t_2 = b_3 - \frac{5b_2}{8\rho_0}, \tag{6.8.19}$$

$$t_3 = b_4 + \frac{7b_2}{16\rho_0^2} - \frac{2b_3}{3\rho_0}, \tag{6.8.20}$$

and the inverse relations

$$b_2 = t_1, \tag{6.8.21}$$

$$b_3 = \frac{5t_1}{8\rho_0} + t_2, \tag{6.8.22}$$

$$b_4 = -\frac{t_1}{48\rho_0^2} + \frac{2t_2}{3\rho_0} + t_3. \tag{6.8.23}$$

Observe that if there is a nonzero quadrupole component (b_2 and/or $a_2 \neq 0$), then U_3 (and all the U_n with $n > 2$) cannot vanish. Also if a sextupole component is nonzero, then all the U_n with $n > 3$ cannot vanish. This can be shown to be a simple consequence of Maxwell's equations when employed with curvilinear coordinates. *The presence of multilpoles in curvilinear coordinates has a feed-up effect.*

The U_n terms produce aberrations of order $(n-1)$. It follows that if the combined function dipole we have been describing has nonzero gradient, then it must also produce second (and higher) order aberrations. These aberrations will in general produce chromatic effects if the dispersion is nonzero. Thus, for example, combined function dipoles generally affect the chromaticity of a ring in a complicated way. The reader should be warned that not all accelerator design codes take this effect into account. In particular, one cannot properly obtain the chromatic effects of a combined function dipole simply by cutting an ordinary dipole into many segments and then inserting thin-lens quadrupoles (and/or higher order multipoles) between the segments.

For other applications (perhaps largely historical) it is convenient to specify the midplane field in terms of an *index* n by making the Ansatz

$$B_y(\xi, 0) = B_0[\rho_0/(\rho_0 + \xi)]^n. \tag{6.8.24}$$

In this case the Taylor coefficients are given by the relations

$$t_1 = -nB_0/\rho_0, \tag{6.8.25}$$

$$t_2 = n(n+1)B_0/(2\rho_0^2), \tag{6.8.26}$$

$$t_3 = -n(n+1)(n+2)B_0/(6\rho_0^3). \tag{6.8.27}$$

The true fringe fields of a combined function dipole are also a very complicated affair. Correspondingly, their proper treatment requires the use of GENMAP (see section 1.4.1) or equivalent routines based upon field data obtained either from high accuracy 3-dimensional

magnet code calculations (such calculations are near the current state of the art) or very careful field measurements. In the absence of such heroic efforts, one may estimate the effect of fringe fields by using the current MaryLie hard-edge models for dipoles and quadrupoles. That is, an approximation to the full transfer map for the combined function dipole in the hard-edge limit can be obtained simply by pre- and post-multiplying the transfer maps of pure dipoles and pure quadrupoles having strengths corresponding to those associated with the combined function dipole parameters. This option, or the option of supplying one's own fringe-field transfer maps, is controlled by the parameters LFRN and TFRN.

Where the MaryLie hard-edge model is used for the dipole portion of the fringe field, the gap size and normalized field integral are given default values of 0 and .5, respectively. If other values are desired, they may be set using a command with the type code *cfrn*. See section 6.29.

For further detail, see references listed in Chapter 11.

```
#comment
 Exhibit 6.8.1.
 This is a MARYLIE run illustrating the sign conventions for
 combined function bends.  A combined function bend is a bend
 with added quadrupole, sextupole, and octupole components.
 Thus, pure dipoles, pure quadrupoles, pure sextupoles,
 and pure octupoles are special cases of the combined function
 bend.  This is verified below for various cases.  Note that to
 obtain a finite length element without dipole content using
 the cfbd element, it is necessary to make both the dipole field
 small and the bend angle small in such a way that the design orbit
 path length as given in figure 6.2a takes on the desired value.
 This limiting operation is not necessary, of course, since
 there are special type codes for this purpose.  However, it
 is done here to illustrate that the combined function bend
 does have the right limiting behavior.
 For simplicity of calculation the value of brho has been set
 equal to 10/pi.  Thus, the path length is given by the relation

  path length = 10/180 * theta (in degrees) /B.

 In particular, if

      theta = 9.e-9 and B = 1.e-9,
 then

       path length = .5 meters.

#beam
  3.183098861830000
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-08  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 cfbd36   cfbd
   36.0000000000000       1.20000000000000       1.00000000000000
   1.00000000000000       1.00000000000000       1.00000000000000
 wcfbd    cfbd
  9.000000000000000E-09  1.000000000000000E-09   1.00000000000000
   1.00000000000000       1.00000000000000       1.00000000000000
 bpar     ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 qpar     ps1
   5.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
```

```
 skqpar   ps1
  0.000000000000000E+00   5.00000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 spar     ps1
  0.000000000000000E+00  0.000000000000000E+00   20.0000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 skspar   ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   20.0000000000000      0.000000000000000E+00  0.000000000000000E+00
 opar     ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   20.0000000000000      0.000000000000000E+00
 skopar   ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00   20.0000000000000
 nbnd     nbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000       1.00000000000000       1.00000000000000
 hfqd     quad
  0.500000000000000       5.00000000000000       1.00000000000000
   1.00000000000000
 sex      sext
  0.500000000000000       20.0000000000000
 oct      octm
  0.500000000000000       20.0000000000000
 arot+q   arot
   45.0000000000000
 arot-q   arot
  -45.0000000000000
 arot+s   arot
   30.0000000000000
 arot-s   arot
  -30.0000000000000
 arot+o   arot
   22.5000000000000
 arot-o   arot
  -22.5000000000000
 clear    iden
 inv      inv
 end      end
#lines
 pcfbd
     1*bpar        1*cfbd36
 qcfbd
     1*qpar        1*wcfbd
 skqcfbd
     1*skqpar      1*wcfbd
 scfbd
     1*spar        1*wcfbd
 skscfbd
     1*skspar      1*wcfbd
 ocfbd
     1*opar        1*wcfbd
```

```
 skocfbd
     1*skopar      1*wcfbd
 skquad
     1*arot+q      1*hfqd        1*arot-q
 sksex
     1*arot+s      1*sex         1*arot-s
 skoct
     1*arot+o      1*oct         1*arot-o
 bendtest
     1*clear       1*nbnd        1*inv         1*pcfbd       1*mapout
 quadtest
     1*clear       1*hfqd        1*inv         1*qcfbd       1*mapout   &
     1*clear       1*skquad      1*inv         1*skqcfbd     1*mapout
 sextest
     1*clear       1*sex         1*inv         1*scfbd       1*mapout   &
     1*clear       1*sksex       1*inv         1*skscfbd     1*mapout
 octtest
     1*clear       1*oct         1*inv         1*ocfbd       1*mapout   &
     1*clear       1*skoct       1*inv         1*skocfbd     1*mapout
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*bendtest
    1*quadtest
    1*sextest
    1*octtest
    1*end
```

Result of bendtest

```
matrix for map is :

 1.00000E+00  3.88578E-16  0.00000E+00 -2.77556E-17  0.00000E+00 -1.52656E-16
-2.42861E-17  1.00000E+00  1.73472E-18 -2.89121E-18  0.00000E+00 -5.82867E-16
-6.15035E-18 -2.24547E-17  1.00000E+00  2.77556E-17  0.00000E+00 -1.93800E-17
 1.40342E-18 -2.70469E-18  0.00000E+00  1.00000E+00  0.00000E+00 -1.04395E-18
-2.91434E-16  2.35922E-16  0.00000E+00  0.00000E+00  1.00000E+00  8.32667E-17
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

Result of quadtest

```
matrix for map is :

 1.00000E+00 -1.24191E-12  3.33861E-17 -1.48041E-17  0.00000E+00 -4.51426E-11
 1.95068E-12  1.00000E+00 -2.32542E-17  3.33861E-17  0.00000E+00 -1.74622E-10
 1.12409E-17 -6.49358E-18  1.00000E+00 -1.24187E-12  0.00000E+00 -4.03897E-28
 1.58205E-17  7.99411E-18 -1.95086E-12  1.00000E+00  0.00000E+00  0.00000E+00
 1.74622E-10 -4.51426E-11  0.00000E+00  0.00000E+00  1.00000E+00 -5.10553E-13
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00 -1.24187E-12 -2.41506E-17  1.90820E-17  0.00000E+00 -4.66693E-11
-2.42861E-17  1.00000E+00 -1.95080E-12  1.04083E-17  0.00000E+00 -1.86837E-10
-3.46945E-17  2.42861E-17  1.00000E+00 -1.24188E-12  0.00000E+00 -1.52674E-12
-1.95087E-12  2.42861E-17  1.73472E-17  1.00000E+00  0.00000E+00 -1.22150E-11
 1.86837E-10 -4.66693E-11  1.22150E-11 -1.52674E-12  1.00000E+00 -5.10553E-13
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

Result of sextest

```
matrix for map is :

 1.00000E+00 -1.24179E-12  0.00000E+00  0.00000E+00  0.00000E+00 -4.66493E-11
-4.93480E-20  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.86597E-10
 0.00000E+00  0.00000E+00  1.00000E+00 -1.24179E-12  0.00000E+00  4.03897E-28
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00 -3.23117E-27
 1.86597E-10 -4.66493E-11  3.23117E-27 -1.61559E-27  1.00000E+00 -5.10553E-13
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00 -1.24179E-12  0.00000E+00  0.00000E+00  0.00000E+00 -4.66493E-11
-4.93480E-20  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.86597E-10
 0.00000E+00  0.00000E+00  1.00000E+00 -1.24179E-12  0.00000E+00  4.03897E-28
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00 -3.23117E-27
 1.86597E-10 -4.66493E-11  3.23117E-27 -1.61559E-27  1.00000E+00 -5.10553E-13
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

Result of octtest

```
matrix for map is :

 1.00000E+00 -1.24179E-12  0.00000E+00  0.00000E+00  0.00000E+00 -4.66493E-11
-4.93480E-20  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.86597E-10
 0.00000E+00  0.00000E+00  1.00000E+00 -1.24179E-12  0.00000E+00  4.03897E-28
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00 -3.23117E-27
 1.86597E-10 -4.66493E-11  3.23117E-27 -1.61559E-27  1.00000E+00 -5.10553E-13
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

```
matrix for map is :

 1.00000E+00 -1.24179E-12  0.00000E+00  0.00000E+00  0.00000E+00 -4.66493E-11
-4.93480E-20  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.86597E-10
 0.00000E+00  0.00000E+00  1.00000E+00 -1.24179E-12  0.00000E+00  4.03897E-28
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00 -3.23117E-27
 1.86597E-10 -4.66493E-11  3.23117E-27 -1.61559E-27  1.00000E+00 -5.10553E-13
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.9 Magnetic Quadrupole

Type Code: quad

Required Parameters:

1. Length of quadrupole (m).
2. Strength, Q (Tesla/m).
   When $Q > 0$, the quadrupole is horizontally focusing.
   When $Q < 0$, the quadrupole is horizontally defocusing.
3. LFRN
   = 0 for no fringe field on leading edge (entry) of quadrupole.
   = 1 for hard-edge fringe field on leading edge of quadrupole.
4. TFRN
   = 0 for no fringe field on trailing edge (exit) of quadrupole.
   = 1 for hard-edge fringe field on trailing edge of quadrupole.

Example:

```
hfq    quad
 .5  , 1.92, 1, 1
```

This specifies a quadrupole with the user given name *hfq*. It has a length of .5 meters, a strength of 1.92 Tesla/meter, is horizontally focusing, and has hard-edge leading and trailing fringe fields.

Vector Potential: $A_z = -\frac{Q}{2}(x^2 - y^2)$

Remark:

There is no type code for the quadrupole hard-edge fringe field transfer maps by themselves. However, they may be obtained from the quadrupole transfer map by setting the length equal to zero.

Example:

```
lqff    quad
 0 ,  1.92 ,  1 ,  0
```

This specifies a map with the user given name *lqff*. It is the map for the leading quadrupole fringe field of a quadrupole having a strength of 1.92 Tesla/meter.

Description:

From the relation $\mathbf{B} = \nabla \times \mathbf{A}$ one has the result

$$\begin{aligned} B_x &= \partial_y A_z - \partial_z A_y = Qy, \\ B_y &= \partial_z A_x - \partial_x A_z = Qx, \\ B_z &= \partial_x A_y - \partial_y A_x = 0. \end{aligned}$$

See figure 6.9.1 for a plot of the midplane vertical field B_y for the horizontally focusing case $Q > 0$.

Note that one has the relations

$$\|\mathbf{B}\|^2 = B_x^2 + B_y^2 = Q^2y^2 + Q^2x^2 = Q^2r^2.$$

See figure 6.9.2. It follows that if B is the pole tip magnetic field and R is the pole tip radius, then the quadrupole strength Q is given by the relation

$$Q = \frac{B}{R}.$$

The map for a quadrupole rotated clockwise by angle θ about the z-axis may be gotten by preceding and following the map for a regular quadrupole by maps corresponding to $arot(-\theta)$ and $arot(+\theta)$, respectively. See Exhibit 6.9.1 and figure 6.14.3.

Suppose the vector potential for a rotated quadrupole is written in the form

$$A_z = -\frac{b}{2}(x^2 - y^2) + \frac{a}{2}2xy.$$

Here b and a are by definition the normal and skew components of the quadrupole strength. Correspondingly, from $\mathbf{B} = \nabla \times \mathbf{A}$, the components of the magnetic field are given by the relations

$$\begin{aligned} B_x &= ax + by, \quad (*) \\ B_y &= bx - ay, \quad (**) \\ B_z &= 0. \end{aligned}$$

Note that the relations $(*)$ and $(**)$ are equivalent to the formal complex relation

$$B_y + iB_x = (b + ia)(x + iy).$$

Then, with these definitions, the quantities $a, b, Q,$ and θ are connected by the relations

$$\begin{aligned} a &= -Q\sin(2\theta), \\ b &= Q\cos(2\theta). \end{aligned}$$

Thus, what is usually called a "positive" skew quadrupole is gotten by rotating a normal quadrupole by $-45°$ around the z-axis. Correspondingly, the poles for a positive skew quadrupole are configured as shown in figure 6.9.3. For this quadrupole the magnetic field is given by the relations

$$B_x = ax, B_y = -ay, B_z = 0,$$

with $a = Q$.

Figure 6.9.1: Midplane magnetic field of a normal quadrupole that horizontally focuses positive particles (end view). Particles on the design orbit move in the $+z$ direction along the z axis.

Figure 6.9.2: Pole configuration for the quadrupole of figure 6.9.1, same end view.

Figure 6.9.3: Pole configuration for a positive skew quadrupole, end view. Note that this figure is gotten by rotating the poles in figure 6.9.2 by $-45°$ clockwise. That is, the rotation is $+45°$ counterclockwise.

```
#comment
 Exhibit 6.9.1.
 This is a MARYLIE run illustrating the sign conventions for a
 a quadrupole.  First, it is shown that for positive particles a quadrupole with positive
 strength is horizontally focusing and vertically defocusing.
 Next, the map for a quadrupole "rotquad" rotated clockwise by angle
 theta about the z axis is gotten by the rule

    rotquad = arot(-theta) * quad * arot(theta).

 Also, what is usually called a skew quad is the result of setting
 theta = -45 degrees.  Thus, the map "skewquad" for a skew quad is
 gotten by the rule

    skewquad = arot(45) * quad * arot(-45).

 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 hfquad   quad
   1.00000000000000       3.00000000000000       1.00000000000000
   1.00000000000000
 arot+45  arot
   45.0000000000000
 arot-45  arot
  -45.0000000000000
 clear    iden
 end      end
#lines
 skewquad
     1*arot+45     1*hfquad      1*arot-45
#lumps
#loops
#labor
    1*fileout
    1*hfquad
    1*matout
    1*raysin
```

```
    1*trace
    1*clear
    1*skewquad
    1*matout
    1*trace
    1*end

matrix for map is :

 7.08109E-01  9.00665E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-5.53571E-01  7.08109E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.32338E+00  1.10563E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  6.79548E-01  1.32338E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.11143E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

matrix for map is :

 1.01574E+00  1.00315E+00  3.07635E-01  1.02483E-01  0.00000E+00  0.00000E+00
 6.29888E-02  1.01574E+00  6.16559E-01  3.07635E-01  0.00000E+00  0.00000E+00
 3.07635E-01  1.02483E-01  1.01574E+00  1.00315E+00  0.00000E+00  0.00000E+00
 6.16559E-01  3.07635E-01  6.29888E-02  1.01574E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.11143E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray traces (contents of file 13):

```
 .01 .00 .00 .00 .00 .00
 .00 .00 .01 .00 .00 .00
```

Final conditions for ray traces (contents of file 14):

Results for normal quadrupole

```
 7.08109E-03 -5.53577E-03  0.00000E+00  0.00000E+00  6.61182E-06  0.00000E+00
 0.00000E+00  0.00000E+00  1.32339E-02  6.79527E-03  8.45425E-06  0.00000E+00
```

Note that the first particle is focused horizontally, and the second is defocused vertically.

Results for skew quadrupole

```
 1.01574E-02  6.29633E-04  3.07638E-03  6.16559E-03  7.53303E-06  0.00000E+00
 3.07638E-03  6.16559E-03  1.01574E-02  6.29633E-04  7.53303E-06  0.00000E+00
```

Note that the first particle is deflected in the $+y$ direction, and the second particle is deflected in the $+x$ direction.

6.10 Magnetic Sextupole

Type Code: sext

Required Parameters:

1. Length of sextupole (m).
2. Strength, S (Tesla/m^2). Consider a positive particle passing through a sextupole and having an energy slightly larger than the design energy. When $S > 0$ (and assuming the local eta dispersion function is positive), the sextupole provides horizontal focusing and vertical defocusing for such a particle. It thus raises the horizontal chromaticity and lowers the vertical chromaticity. When $S < 0$, the sextupole lowers the horizontal chromaticity and raises the vertical chromaticity.

Example:

```
hcs    sext
.5  ,  3.5
```

This specifies a sextupole with the user given name *hcs*. It has a length of .5 meters and a strength of 3.5 Tesla/(meter)^2. Under normal usage it raises the horizontal chromaticity and lowers the vertical chromaticity.

Vector Potential: $A_z = -\frac{S}{3}(x^3 - 3xy^2)$

Description:

From the relation $\mathbf{B} = \nabla \times \mathbf{A}$ one has the result

$$\begin{aligned} B_x &= \partial_y A_z - \partial_z A_y &&= 2Sxy, \\ B_y &= \partial_z A_x - \partial_x A_z &&= S(x^2 - y^2), \\ B_z &= \partial_x A_y - \partial_y A_x &&= 0. \end{aligned}$$

See figure 6.10.1 for a plot of the midplane vertical field B_y for the case $S > 0$.

Note that one has the relations

$$\begin{aligned} \|\mathbf{B}\|^2 &= B_x^2 + B_y^2 = 4S^2x^2y^2 + S^2(x^4 - 2x^2y^2 + y^4) \\ &= S^2(x^4 + 2x^2y^2 + y^4) = S^2r^4. \end{aligned}$$

See figure 6.10.2. It follows that if B is the pole tip magnetic field and R is the pole tip radius, then the sextupole strength S is given by the relation

$$S = \frac{B}{R^2}.$$

The map for a sextupole rotated clockwise by angle θ about the z-axis may be gotten by preceding and following the map for a regular sextupole by maps corresponding to $arot(-\theta)$ and $arot(+\theta)$, respectively. See Exhibit 6.10.1 and figure 6.14.3.

Suppose the vector potential for a rotated sextupole is written in the form

$$A_z = -\frac{b}{3}(x^3 - 3xy^2) + \frac{a}{3}(3x^2y - y^3).$$

Here b and a are by definition the normal and skew components of the sextupole strength. Correspondingly, from $\mathbf{B} = \nabla \times \mathbf{A}$, the components of the magnetic field are given by the relations

$$\begin{aligned} B_x &= 2bxy + a(x^2 - y^2), \quad (*) \\ B_y &= b(x^2 - y^2) - 2axy, \quad (**) \\ B_z &= 0. \end{aligned}$$

Note that the relations (*) and (**) are equivalent to the formal complex relation

$$B_y + iB_x = (b + ia)(x + iy)^2.$$

Then, with these definitions, the quantities a, b, S, and θ are connected by the relations

$$\begin{aligned} a &= -S\sin(3\theta), \\ b &= S\cos(3\theta). \end{aligned}$$

Thus, what is usually called a "positive" skew sextupole is gotten by rotating a normal sextupole by $-30°$ around the z-axis. Correspondingly, the poles for a positive skew sextupole are configured as shown in figure 6.10.3. For this sextupole the magnetic field is given by the relations

$$B_x = a(x^2 - y^2), B_y = -2axy, B_z = 0,$$

with $a = S$.

Figure 6.10.1: End view showing midplane magnetic field of a normal sextupole that provides horizontal focusing for a positive particle if $x > 0$ and horizontal defocusing if $x < 0$. Note that such a sextupole always provides (for a positive particle) a net deflection in the $-x$ direction unless x is initially zero. See Exhibit 6.10.1.

Figure 6.10.2: Pole configuration for the sextupole of figure 6.10.1, same end view.

Figure 6.10.3: Pole configuration for a positive skew sextupole, end view. Note that this figure is gotten by rotating the poles in figure 6.10.2 by $-30°$ clockwise. That is, the rotation is $+30°$ counterclockwise.

```
#comment
 Exhibit 6.10.1.
 This is a MARYLIE run illustrating the sign conventions for a
 a sextupole.  First, it is shown that a sextupole always deflects a
 positive particle
 in the -x direction unless x is initially zero.
 Next, the map for a sextupole "rotsext" rotated clockwise by angle
 theta about the z axis is gotten by the rule

    rotsext = arot(-theta) * sext * arot(theta).

 Also, what is usually called a positive skew sextupole is the result
 of setting theta = -30 degrees.  Thus, the map "skewsext" for a skew
 sextupole is gotten by the rule

    skewsext = arot(30) * sext * arot(-30).

 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  czer
  1.000000000000000E-14  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 sext     sext
  0.500000000000000       3.00000000000000
 arot+30  arot
   30.0000000000000
 arot-30  arot
  -30.0000000000000
 clear    iden
 end      end
#lines
 skewsext
     1*arot+30     1*sext        1*arot-30
#lumps
#loops
#labor
    1*fileout
    1*setzero
```

```
    1*sext
    1*mapout
    1*raysin
    1*trace
    1*clear
    1*skewsext
    1*mapout
    1*trace
    1*end

matrix for map is :

 1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.05572E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=-0.10243738181844D+00
 f( 29)=f( 21 00 00 )= 0.76828036363829D-01
 f( 34)=f( 12 00 00 )=-0.25609345454610D-01
 f( 39)=f( 10 20 00 )= 0.30731214545532D+00
 f( 40)=f( 10 11 00 )=-0.15365607272766D+00
 f( 43)=f( 10 02 00 )= 0.25609345454610D-01
 f( 49)=f( 03 00 00 )= 0.32011681818262D-02
 f( 53)=f( 02 00 01 )=-0.29697889251752D+00
 f( 54)=f( 01 20 00 )=-0.76828036363829D-01
 f( 55)=f( 01 11 00 )= 0.51218690909220D-01
 f( 58)=f( 01 02 00 )=-0.96035045454787D-02
 f( 76)=f( 00 02 01 )=-0.29697889251752D+00
 f( 83)=f( 00 00 03 )=-0.12210091207749D+00
 f( 84)=f( 40 00 00 )= 0.39350314476813D-02
 f( 85)=f( 31 00 00 )=-0.39350314476813D-02
 f( 90)=f( 22 00 00 )= 0.14756367928805D-02
 f( 95)=f( 20 20 00 )= 0.78700628953625D-02
 f( 96)=f( 20 11 00 )=-0.39350314476813D-02
 f( 99)=f( 20 02 00 )= 0.98375786192031D-04
 f(105)=f( 13 00 00 )=-0.24593946548008D-03
 f(109)=f( 12 00 01 )=-0.15210870102417D-01
 f(110)=f( 11 20 00 )=-0.39350314476813D-02
 f(111)=f( 11 11 00 )= 0.27545220133769D-02
 f(114)=f( 11 02 00 )=-0.24593946548008D-03
 f(132)=f( 10 02 01 )= 0.15210870102417D-01
 f(140)=f( 04 00 00 )=-0.62482432895323D-01
 f(144)=f( 03 00 01 )= 0.38027175256043D-02
 f(145)=f( 02 20 00 )= 0.98375786192031D-04
 f(146)=f( 02 11 00 )=-0.24593946548008D-03
 f(149)=f( 02 02 00 )=-0.12496486579065D+00
 f(154)=f( 02 00 02 )=-0.40417877560560D+00
 f(161)=f( 01 11 01 )= 0.30421740204834D-01
```

```
 f(167)=f( 01 02 01 )=-0.11408152576813D-01
 f(175)=f( 00 40 00 )= 0.39350314476813D-02
 f(176)=f( 00 31 00 )=-0.39350314476813D-02
 f(179)=f( 00 22 00 )= 0.14756367928805D-02
 f(185)=f( 00 13 00 )=-0.24593946548008D-03
 f(195)=f( 00 04 00 )=-0.62482432895323D-01
 f(200)=f( 00 02 02 )=-0.40417877560560D+00
 f(209)=f( 00 00 04 )=-0.15561050561982D+00

matrix for map is :

 1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.05572E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 30)=f( 20 10 00 )= 0.30731214545532D+00
 f( 31)=f( 20 01 00 )=-0.76828036363829D-01
 f( 35)=f( 11 10 00 )=-0.15365607272766D+00
 f( 36)=f( 11 01 00 )= 0.51218690909220D-01
 f( 50)=f( 02 10 00 )= 0.25609345454610D-01
 f( 51)=f( 02 01 00 )=-0.96035045454787D-02
 f( 53)=f( 02 00 01 )=-0.29697889251752D+00
 f( 64)=f( 00 30 00 )=-0.10243738181844D+00
 f( 65)=f( 00 21 00 )= 0.76828036363829D-01
 f( 68)=f( 00 12 00 )=-0.25609345454610D-01
 f( 74)=f( 00 03 00 )= 0.32011681818262D-02
 f( 76)=f( 00 02 01 )=-0.29697889251752D+00
 f( 83)=f( 00 00 03 )=-0.12210091207749D+00
 f( 84)=f( 40 00 00 )= 0.39350314476813D-02
 f( 85)=f( 31 00 00 )=-0.39350314476813D-02
 f( 90)=f( 22 00 00 )= 0.14756367928805D-02
 f( 95)=f( 20 20 00 )= 0.78700628953625D-02
 f( 96)=f( 20 11 00 )=-0.39350314476813D-02
 f( 99)=f( 20 02 00 )= 0.98375786192031D-04
 f(105)=f( 13 00 00 )=-0.24593946548008D-03
 f(110)=f( 11 20 00 )=-0.39350314476813D-02
 f(111)=f( 11 11 00 )= 0.27545220133769D-02
 f(114)=f( 11 02 00 )=-0.24593946548008D-03
 f(116)=f( 11 01 01 )= 0.30421740204834D-01
 f(140)=f( 04 00 00 )=-0.62482432895323D-01
 f(145)=f( 02 20 00 )= 0.98375786192031D-04
 f(146)=f( 02 11 00 )=-0.24593946548008D-03
 f(148)=f( 02 10 01 )= 0.15210870102417D-01
 f(149)=f( 02 02 00 )=-0.12496486579065D+00
 f(151)=f( 02 01 01 )=-0.11408152576813D-01
 f(154)=f( 02 00 02 )=-0.40417877560560D+00
 f(175)=f( 00 40 00 )= 0.39350314476813D-02
 f(176)=f( 00 31 00 )=-0.39350314476813D-02
```

```
 f(179)=f( 00 22 00 )= 0.14756367928805D-02
 f(185)=f( 00 13 00 )=-0.24593946548008D-03
 f(187)=f( 00 12 01 )=-0.15210870102417D-01
 f(195)=f( 00 04 00 )=-0.62482432895323D-01
 f(197)=f( 00 03 01 )= 0.38027175256043D-02
 f(200)=f( 00 02 02 )=-0.40417877560560D+00
 f(209)=f( 00 00 04 )=-0.15561050561982D+00
```

Initial conditions for ray traces (contents of file 13):

```
-.01 .00 .00 .00 .00 .00
+.01 .00 .00 .00 .00 .00
```

Final conditions for ray traces (contents of file 14):

Results for normal sextupole

```
-1.00077E-02 -3.07470E-05  0.00000E+00  0.00000E+00  9.34897E-11  0.00000E+00
 9.99232E-03 -3.07155E-05  0.00000E+00  0.00000E+00  9.34897E-11  0.00000E+00
```

Note that both rays are deflected toward $-x$.

Results for skew sextupole

```
-1.00000E-02 -1.57401E-08  7.68280E-06  3.07312E-05  9.34897E-11  0.00000E+00
 1.00000E-02  1.57401E-08  7.68280E-06  3.07312E-05  9.34897E-11  0.00000E+00
```

Note that both rays are deflected up (toward $+y$).

6.11 Magnetic Octupole

Type Code: octm

Required Parameters:

1. Length of octupole (m).
2. Strength, O (Tesla/m^3)

Example:

```
oct2    octm
 1.  ,  1.5
```

This specifies a magnetic octupole with the user given name *oct2*. It has a length of 1 meter, and a strength of 1.5 Tesla/(meter)^3.

Vector Potential: $A_z = \dfrac{-O}{4}(x^4 - 6x^2y^2 + y^4)$

Description:

From the relation $\mathbf{B} = \nabla \times \mathbf{A}$ one has the result

$$\begin{aligned} B_x &= \partial_y A_z - \partial_z A_y &&= -O(y^3 - 3x^2y), \\ B_y &= \partial_z A_x - \partial_x A_z &&= O(x^3 - 3xy^2), \\ B_z &= \partial_x A_y - \partial_y A_x &&= 0. \end{aligned}$$

See figure 6.11.1 for a plot of the midplane vertical field B_y for the case $O > 0$.

Note that one has the relations

$$\begin{aligned} \|\mathbf{B}\|^2 &= B_x^2 + B_y^2 = O^2(y^3 - 3x^2y)^2 + O^2(x^3 - 3xy^2)^2 \\ &= O^2(x^6 + 3x^4y^2 + 3x^2y^4 + y^6) = O^2r^6 \end{aligned}$$

See figure 6.11.2. It follows that if B is the pole tip magnetic field and R is the pole tip radius, then the octupole strength O is given by the relation

$$O = \frac{B}{R^3}.$$

The map for an octupole rotated clockwise by angle θ about the z-axis may be gotten by preceding and following the map for a regular octupole by maps corresponding to $arot(-\theta)$ and $arot(\theta)$, respectively. See Exhibit 6.11.1 and figure 6.14.3.

Suppose the vector potential for a rotated octupole is written in the form

$$A_z = -\frac{b}{4}(x^4 - 6x^2y^2 + y^4) + \frac{a}{4}(4x^3y - 4xy^3).$$

Here b and a are by definition the normal and skew components of the octupole strength. Correspondingly, from $\mathbf{B} = \nabla \times \mathbf{A}$, the components of the magnetic field are given by the relations

$$\begin{aligned} B_x &= -b(y^3 - 3x^2y) + a(x^3 - 3xy^2), \quad (*) \\ B_y &= b(x^3 - 3xy^2) - a(3x^2y - y^3), \quad (**) \\ B_z &= 0. \end{aligned}$$

Note that the relations (*) and (**) are equivalent to the formal complex relation

$$B_y + iB_x = (b + ia)(x + iy)^3.$$

Then, with these definitions, the quantities a, b, O, and θ are connected by the relations

$$\begin{aligned} a &= -O\sin(4\theta), \\ b &= O\cos(4\theta). \end{aligned}$$

Thus, what is usually called a "positive" skew octupole is gotten by rotating a normal octupole by $-22.5°$ around the z-axis. Correspondingly, the poles for a positive skew octupole are configured as shown in figure 6.11.3. For this octupole the magnetic field is given by the relations

$$B_x = a(x^3 - 3xy^2), B_y = -a(3x^2y - y^3), B_z = 0,$$

with $a = O$.

Figure 6.11.1: End view showing midplane magnetic field of a normal octupole. It horizontally focuses (in a nonlinear way) positive particles. See Exhibit 6.11.1, and compare with figure 6.9.1.

Figure 6.11.2: Pole configuration for the octupole of figure 6.11.1, same end view.

Figure 6.11.3: Pole configuration for a positive skew octupole, end view. Note that this figure is gotten by rotating the poles in figure 6.11.2 by $-22.5°$ clockwise. That is, the rotation is $+22.5°$ counterclockwise.

```
#comment
 Exhibit 6.11.1.
 This is a MARYLIE run illustrating the sign conventions for an
 octupole.  First, it is shown that an octupole with positive
 strength is horizontally focusing and vertically defocusing for positive
 particles.
 Next, the map for an octupole "rotoct" rotated clockwise by angle
 theta about the z axis is gotten by the rule

    rotoct = arot(-theta) * oct * arot(theta).

 Also, what is usually called a positive skew octupole is the result
 of setting theta = -22.5 degrees.  Thus, the map "skewoct" for a
 skew octupole is gotten by the rule

    skewoct = arot(+22.5) * oct * arot(-22.5).

 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-14  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 oct      octm
  0.500000000000000       8.00000000000000
 arot+    arot
   22.5000000000000
 arot-    arot
  -22.5000000000000
 clear    iden
 end      end
#lines
 skewoct
     1*arot+       1*oct         1*arot-
#lumps
#loops
#labor
    1*fileout
    1*setzero
```

```
    1*oct
    1*mapout
    1*raysin
    1*trace
    1*clear
    1*skewoct
    1*mapout
    1*trace
    1*end

matrix for map is :

 1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.05572E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 53)=f( 02 00 01 )=-0.29697889251752D+00
 f( 76)=f( 00 02 01 )=-0.29697889251752D+00
 f( 83)=f( 00 00 03 )=-0.12210091207749D+00
 f( 84)=f( 40 00 00 )=-0.20487476363688D+00
 f( 85)=f( 31 00 00 )= 0.20487476363688D+00
 f( 90)=f( 22 00 00 )=-0.10243738181844D+00
 f( 95)=f( 20 20 00 )= 0.12292485818213D+01
 f( 96)=f( 20 11 00 )=-0.61462429091063D+00
 f( 99)=f( 20 02 00 )= 0.10243738181844D+00
 f(105)=f( 13 00 00 )= 0.25609345454610D-01
 f(110)=f( 11 20 00 )=-0.61462429091063D+00
 f(111)=f( 11 11 00 )= 0.40974952727376D+00
 f(114)=f( 11 02 00 )=-0.76828036363829D-01
 f(140)=f( 04 00 00 )=-0.65060934545461D-01
 f(145)=f( 02 20 00 )= 0.10243738181844D+00
 f(146)=f( 02 11 00 )=-0.76828036363829D-01
 f(149)=f( 02 02 00 )=-0.10963439272723D+00
 f(154)=f( 02 00 02 )=-0.40417877560560D+00
 f(175)=f( 00 40 00 )=-0.20487476363688D+00
 f(176)=f( 00 31 00 )= 0.20487476363688D+00
 f(179)=f( 00 22 00 )=-0.10243738181844D+00
 f(185)=f( 00 13 00 )= 0.25609345454610D-01
 f(195)=f( 00 04 00 )=-0.65060934545461D-01
 f(200)=f( 00 02 02 )=-0.40417877560560D+00
 f(209)=f( 00 00 04 )=-0.15561050561982D+00

matrix for map is :

 1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  5.00000E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
```

```
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.05572E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 53)=f( 02 00 01 )=-0.29697889251752D+00
 f( 76)=f( 00 02 01 )=-0.29697889251752D+00
 f( 83)=f( 00 00 03 )=-0.12210091207749D+00
 f( 86)=f( 30 10 00 )= 0.81949905454751D+00
 f( 87)=f( 30 01 00 )=-0.20487476363688D+00
 f( 91)=f( 21 10 00 )=-0.61462429091063D+00
 f( 92)=f( 21 01 00 )= 0.20487476363688D+00
 f(106)=f( 12 10 00 )= 0.20487476363688D+00
 f(107)=f( 12 01 00 )=-0.76828036363829D-01
 f(120)=f( 10 30 00 )=-0.81949905454751D+00
 f(121)=f( 10 21 00 )= 0.61462429091063D+00
 f(124)=f( 10 12 00 )=-0.20487476363688D+00
 f(130)=f( 10 03 00 )= 0.25609345454610D-01
 f(140)=f( 04 00 00 )=-0.62500000000000D-01
 f(141)=f( 03 10 00 )=-0.25609345454610D-01
 f(142)=f( 03 01 00 )= 0.10243738181844D-01
 f(149)=f( 02 02 00 )=-0.12500000000000D+00
 f(154)=f( 02 00 02 )=-0.40417877560560D+00
 f(155)=f( 01 30 00 )= 0.20487476363688D+00
 f(156)=f( 01 21 00 )=-0.20487476363688D+00
 f(159)=f( 01 12 00 )= 0.76828036363829D-01
 f(165)=f( 01 03 00 )=-0.10243738181844D-01
 f(195)=f( 00 04 00 )=-0.62500000000000D-01
 f(200)=f( 00 02 02 )=-0.40417877560560D+00
 f(209)=f( 00 00 04 )=-0.15561050561982D+00
```

Initial conditions for ray traces (contents of file 13):

```
+.01 .00  .00 .00 .00 .00
-.01 .00  .00 .00 .00 .00
 .00 .00 +.01 .00 .00 .00
 .00 .00 -.01 .00 .00 .00
```

Final conditions for ray traces: Results for normal octupole

```
 9.99980E-03 -8.19499E-07  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-9.99980E-03  8.19499E-07  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  9.99980E-03 -8.19499E-07  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -9.99980E-03  8.19499E-07  0.00000E+00  0.00000E+00
```

Note that there is (nonlinear) focusing both horizontally and vertically. (It can be shown, however, that skew rays are defocused.)

Results for skew octupole

```
 1.00000E-02  1.17094E-23  2.04875E-07  8.19499E-07  0.00000E+00  0.00000E+00
-1.00000E-02 -1.17094E-23 -2.04875E-07 -8.19499E-07  0.00000E+00  0.00000E+00
-2.04875E-07 -8.19499E-07  1.00000E-02  2.77556E-23  0.00000E+00  0.00000E+00
 2.04875E-07  8.19499E-07 -1.00000E-02 -2.77556E-23  0.00000E+00  0.00000E+00
```

Note that horizontal rays (the first two) are deflected vertically, and vertical rays (the last two) are deflected horizontally.

6.12 Electrostatic Octupole

Type Code: octe

Required Parameters:

1. Length of octupole (m).
2. Strength, O (volts/m^4)

Example:

```
oct3    octe
 1.  ,  1.5D04
```

This specifies an electric octupole with the user given name *oct3*. It has a length of 1 meter, and a strength of 15 kV/(meter)^4.

Scalar Potential: $\psi = \frac{O}{4}(x^4 - 6x^2y^2 + y^4)$

Description:

Available but not yet documented.

6.13 Short Radio Frequency Cavity

Type Code: srfc

Required Parameters:

1. Maximum potential drop across the cavity, $V(= \epsilon_0 L)$, in volts.
2. Frequency of the cavity in *Hertz*.

Example:

```
cvty    srfc
80.D+03  ,  1.D+09
```

This specifies a short radio frequency cavity with the user given name *cvty*. It has a maximum potential drop of 80 kV, and operates at a frequency of 1000 Megahertz.

Vector Potential: $A_z = -\dfrac{\epsilon_0}{\omega}\cos\omega t$

Description:

The short radio frequency cavity is modeled in the impulsive approximation; the cavity is assumed to have an infinitely short "active" section preceded and followed by finite drift sections. The active section is assumed to have a longitudinal electric field given by $\boldsymbol{\epsilon} = (\epsilon_0 \sin\omega t)\mathbf{e}_z$. In the impulsive approximation, the length of the cavity, L, is allowed to tend to zero in such a way that the product $\epsilon_0 L$ is finite. A particle passing through the cavity receives an accelerating or decelerating "kick" in energy, the magnitude of the kick being dependent on the arrival time of the particle.

Note:

If a lattice is operated with a beam energy below the lattice transition energy, bunching for positive particles occurs when $V > 0$. If a lattice is operated with a beam energy above the lattice transition energy, bunching for positive particles occurs when $V < 0$. See sections 2.6 and 8.1.

6.14 Axial Rotation

Type Code: arot

Required Parameters:

1. Angle in degrees.

Example:

```
twist    arot
45
```

This specifies a map with the user given name *twist*. It applies a 45° axial rotation to phase-space points.

Description:

The type code *arot* produces an active rotation about the z-axis of phase space points. When phase space is viewed as in figure 6.14.1, the rotation is clockwise if the angle is positive. See Exhibit 6.14.1. By contrast, if phase space is plotted as in figure 6.14.2, the rotation appears to be counterclockwise. As illustrated in sections 6.9, 6.10, and 6.11, in order to get the map for a straight line element (an element whose design orbit is a straight line) rotated clockwise by the angle θ, the element map should be preceded by $arot(-\theta)$ and followed by $arot(+\theta)$. See figure 6.14.3. The case of curved elements is similar for angles of 90° and 180°. See sections 6.2 and 6.3. The case of curved elements for arbitrary angles requires more thought.

Figure 6.14.1: Geometry for Axial Rotation with reference to standard beamline element coordinate system. The type code *arot* with a positive angle produces an active clockwise rotation of phase-space points about the z axis.

Figure 6.14.2: The same phase-space points referred to phase-space axes having the usual orientation. Note that the rotation now appears to be counterclockwise.

Figure 6.14.3: The map for a straight element rotated clockwise by angle θ about the z axis may be gotten by preceding the map for that element by $arot(-\theta)$ and following it by $arot(+\theta)$.

```
#comment
 Exhibit 6.14.1.
 This is a MARYLIE run illustrating the sign conventions for
 axial rotations.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rot+10   arot
   10.0000000000000
 end      end
#labor
    1*fileout
    1*rot+10
    1*matout
    1*raysin
    1*trace
    1*end

matrix for map is :

 9.84808E-01  0.00000E+00 -1.73648E-01  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  9.84808E-01  0.00000E+00 -1.73648E-01  0.00000E+00  0.00000E+00
 1.73648E-01  0.00000E+00  9.84808E-01  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.73648E-01  0.00000E+00  9.84808E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

Initial conditions for ray trace (contents of file 13):

```
 .02 .01 .00 .00 .00 .00
 .00 .00 .02 .01 .00 .00
```

Final conditions for ray trace (contents of file 14):

```
 1.96962E-02  9.84808E-03  3.47296E-03  1.73648E-03  0.00000E+00  0.00000E+00
-3.47296E-03 -1.73648E-03  1.96962E-02  9.84808E-03  0.00000E+00  0.00000E+00
```

6.15 Twiss Matrix

Type Code: twsm

Required Parameters:

1. IPLANE
   = 1 for X, P_x plane.
   = 2 for Y, P_y plane.
   = 3 for τ, P_τ plane.
2. PHAD (phase advance) in degrees.
3. ALPHA
4. BETA

Example:

```
twsx    twsm
 1 , 60 , 0 , 1
```

This specifies a map with the user given name *twsx*. It produces a 60° clockwise rotation in the X, P_x phase-space plane, and leaves the vertical and temporal phase-space coordinates unchanged.

Description:

This type code is useful for producing idealized betatron motion. For example, when IPLANE = 1, it produces a linear map with the transfer matrix

$$R = \begin{pmatrix} \cos w + \alpha \sin w & \beta \sin w & 0 & 0 & 0 & 0 \\ -\gamma \sin w & \cos w - \alpha \sin w & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.15.1}$$

where w = PHAD and the quantity γ is computed from the relation

$$\gamma = \frac{1 + \alpha^2}{\beta}. \tag{6.15.2}$$

Simultaneous transformations in several planes can be accomplished by defining a *line* whose contents consists of multiple twiss elements with different values of IPLANE. Exhibit 6.15.1 illustrates that the map produced by *twsm* has the advertised tunes and twiss parameters. Section 10.7 illustrates the use of *twsm* to produce phase-space data lying on the topological product of two ellipses in the X, P_x and Y, P_y planes (a 2-torus). See also section 7.37.

```
#comment
 Exhibit 6.15.1.
 This is a MARYLIE run illustrating that the type code twsm
 does indeed produce a map with the expected tunes and
 twiss parameters.
#beam
 0.0000000000000000E+00
 0.0000000000000000E+00
 0.0000000000000000E+00
 0.0000000000000000E+00
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 twsx     twsm
   1.00000000000000       45.0000000000000       1.00000000000000
   2.00000000000000
 twsy     twsm
   2.00000000000000       60.0000000000000       3.00000000000000
   4.00000000000000
 twst     twsm
   3.00000000000000       3.60000000000000       5.00000000000000
   6.00000000000000
 tadm     tadm
   12.0000000000000      0.000000000000000E+00   3.00000000000000
  0.000000000000000E+00
 end      end
#lines
 twsa
     1*twsx        1*twsy        1*twst
#lumps
#loops
#labor
    1*fileout
    1*twsa
    1*mapout
    1*tadm
    1*end

matrix for map is :

 1.41421E+00  1.41421E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-7.07107E-01 -1.38778E-17  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.09808E+00  3.46410E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -2.16506E+00 -2.09808E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.31198E+00  3.76743E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -2.72092E-01  6.84074E-01

nonzero elements in generating polynomial are :
```

```
twiss analysis of dynamic map

horizontal tune =  0.1250000000000000
vertical tune =  0.1666666666666667
temporal tune =  1.0000000000000002E-02

normalized anharmonicities
 hhn=  0.0000000000000000E+00
 vvn=  0.0000000000000000E+00
 ttn=  0.0000000000000000E+00
 hvn=  0.0000000000000000E+00
 htn=  0.0000000000000000E+00
 vtn=  0.0000000000000000E+00

horizontal twiss parameters
diagonal terms (alpha,beta,gamma)
  1.000000000000000       2.000000000000000       1.000000000000000
full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.10000000000000D+01
 f(  8)=f( 11 00 00 )= 0.20000000000000D+01
 f( 13)=f( 02 00 00 )= 0.20000000000000D+01

vertical twiss parameters
diagonal terms (alpha,beta,gamma)
  3.000000000000000       4.000000000000000       2.500000000000000
full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f( 18)=f( 00 20 00 )= 0.25000000000000D+01
 f( 19)=f( 00 11 00 )= 0.60000000000000D+01
 f( 22)=f( 00 02 00 )= 0.40000000000000D+01

temporal twiss parameters
diagonal terms (alpha,beta,gamma)
  4.999999999999999       5.999999999999999       4.333333333333332
full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f( 25)=f( 00 00 20 )= 0.43333333333333D+01
 f( 26)=f( 00 00 11 )= 0.10000000000000D+02
 f( 27)=f( 00 00 02 )= 0.60000000000000D+01

horizontal envelopes (exh,exv,ext;epxh,epxv,epxt)
  1.414213562373095      0.0000000000000000E+00  0.0000000000000000E+00
  1.000000000000000      0.0000000000000000E+00  0.0000000000000000E+00

vertical envelopes (eyh,eyv,eyt;epyh,epyv,epyt)
 0.0000000000000000E+00   2.000000000000000      0.0000000000000000E+00
```

```
 0.0000000000000000E+00   1.581138830084190      0.0000000000000000E+00

temporal envelopes (eth,etv,ett;epth,eptv,eptt)
 0.0000000000000000E+00  0.0000000000000000E+00   2.449489742783178
 0.0000000000000000E+00  0.0000000000000000E+00   2.081665999466133
```

6.16 Thin Low Order Multipole

Type Code: thlm

Required Parameters:

1. BQD (Tesla) (normal)
2. AQD (Tesla) (skew)
3. BSEX (Tesla/m) (normal)
4. ASEX (Tesla/m) (skew)
5. BOCT (Tesla/m^2) (normal)
6. AOCT (Tesla/m^2) (skew)

Example:

```
octkick    thlm
 0 , 0 , 0 , 0 , 0 , 2.
```

This specifies a map with the user given name *octkick*. It describes a skew magnetic octupole, with a path integrated strength of 2 Tesla/m^2, in the kick approximation.

Description:

The type code *thlm* may be used to produce the results of thick multipoles in the limiting case that they become very strong and very short in such a way that the product length times strength approaches a definite limit. See Exhibit 6.16.1. Note that the thin low order quadrupole element has no fringe fields.

```
#comment
 Exhibit 6.16.1.
 This is a MARYLIE run illustrating the sign conventions for a
 thin low order multipole.  In this run thin low order multipoles
 are compared with their thick counterparts in limiting cases.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-08  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 thqd     thlm
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 skthqd   thlm
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 thsex    thlm
  0.000000000000000E+00  0.000000000000000E+00   1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 skthsex  thlm
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
 thoct    thlm
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 skthoct  thlm
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00   1.00000000000000
 limqd    quad
  5.000000000000000E-09   200000000.000000      0.000000000000000E+00
  0.000000000000000E+00
 limsex   sext
  5.000000000000000E-10   2000000000.00000
 limoct   octm
  5.000000000000000E-10   2000000000.00000
 arot+q   arot
   45.0000000000000
 arot-q   arot
  -45.0000000000000
 arot+s   arot
   30.0000000000000
 arot-s   arot
  -30.0000000000000
 arot+o   arot
```

```
   22.5000000000000
 arot-o   arot
  -22.5000000000000
 clear    iden
 inv      inv
 end      end
#lines
 sklimqd
     1*arot+q      1*limqd       1*arot-q
 sklimsex
     1*arot+s      1*limsex      1*arot-s
 sklimoct
     1*arot+o      1*limoct      1*arot-o
 qdtest
     1*clear       1*limqd       1*inv         1*thqd        1*mapout   &
     1*clear       1*sklimqd     1*inv         1*skthqd      1*mapout
 sextest
     1*clear       1*limsex      1*inv         1*thsex       1*mapout   &
     1*clear       1*sklimsex    1*inv         1*skthsex     1*mapout
 octtest
     1*clear       1*limoct      1*inv         1*thoct       1*mapout   &
     1*clear       1*sklimoct    1*inv         1*skthoct     1*mapout
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*qdtest
    1*sextest
    1*octtest
    1*end
```

Results of qdtest:

```
matrix for map is :

 1.00000E+00 -5.00000E-09  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 6.99561E-11  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00 -5.00000E-09  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  6.99547E-11  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -2.05572E-09
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00 -5.00000E-09  5.12187E-10 -8.53663E-19  0.00000E+00  0.00000E+00
 6.99554E-11  1.00000E+00 -7.32053E-16 -5.12187E-10  0.00000E+00  0.00000E+00
 5.12187E-10 -8.53663E-19  1.00000E+00 -5.00000E-09  0.00000E+00  0.00000E+00
-7.32053E-16 -5.12187E-10  6.99554E-11  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -2.05572E-09
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :
```

Results of sextest:

```
matrix for map is :

 1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -2.05572E-10
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -2.05572E-10
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

Results of octtest:

```
matrix for map is :

 1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -2.05572E-10
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00 -5.00000E-10  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -2.05572E-10
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.17 "Compressed" Low Order Multipole

Type Code: cplm

Required Parameters:

1. Length of element (m).
2. BSEX (path integrated strength,Tesla/m) (normal)
3. ASEX (path integrated strength,Tesla/m) (skew)
4. BOCT (path integrated strength,Tesla/m^2) (normal)
5. AOCT (path integrated strength,Tesla/m^2) (skew)

Example:

```
compsex    cplm
2.0 , 3.5 , 0 , 0 , 0
```

This specifies a map with the user given name *compsex*. It describes a "compressed" normal sextupole with an original length of 2 meters, and a path integrated strength of 3.5 Tesla/m.

Description:

An Exhibit 6.17.1 illustrates, compressed multipoles are the result of removing two half drifts from thick multipoles. Compressed multipoles are sometimes useful for adding multipole content to dipoles.

```
#comment
 Exhibit 6.17.1.
 This is a MARYLIE run illustrating the sign conventions for
 compressed low order multipoles.  Consider a thick multipole
 of length L.  Then a compressed multipole is gotten by removing
 two half drifts from the thick multipole.  This is done by
 sandwiching the thick multipole between two half drifts with
 negative lengths.  Thus, a compressed multipole is defined by
 the rule

 (compressed multipole) = drift(-L/2) * (thick multipole) * drift(-L/2).

 Conversely, one has the result

 (thick multipole) = drift(+L/2) * (compressed multipole) * drift(+L/2).

 This is verified below for various cases.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-10  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 nhdr     drft
  -1.00000000000000
 cpsex    cplm
   2.00000000000000       3.50000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00
 skcpsex  cplm
   2.00000000000000      0.000000000000000E+00   3.50000000000000
  0.000000000000000E+00  0.000000000000000E+00
 cpoct    cplm
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   3.50000000000000      0.000000000000000E+00
 skcpoct  cplm
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   3.50000000000000
 sex      sext
   2.00000000000000       1.75000000000000
 oct      octm
   2.00000000000000       1.75000000000000
 arot+s   arot
   30.0000000000000
 arot-s   arot
  -30.0000000000000
```

```
 arot+o   arot
   22.5000000000000
 arot-o   arot
  -22.5000000000000
 clear    iden
 inv      inv
 end      end
#lines
 sdsex
     1*nhdr        1*sex         1*nhdr
 sksdsex
     1*arot+s      1*sdsex       1*arot-s
 sdoct
     1*nhdr        1*oct         1*nhdr
 sksdoct
     1*arot+o      1*sdoct       1*arot-o
 sextest
     1*clear       1*sdsex       1*inv         1*cpsex       1*mapout   &
     1*clear       1*sksdsex     1*inv         1*skcpsex     1*mapout
 octtest
     1*clear       1*sdoct       1*inv         1*cpoct       1*mapout   &
     1*clear       1*sksdoct     1*inv         1*skcpoct     1*mapout
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*sextest
    1*octtest
    1*end
```

Result of sextest

```
matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :
```

Result of octtest

```
matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.18 J Map

Type Code: jmap

Required Parameters: None

Example:

```
jay    jmap
```

This specifies a command with the user given name *jay*. It produces a transfer map whose matrix part is J, and whose polynomials f_3 and f_4 are zero.

Description:

It is useful to have the Poisson matrix J available for checking and exploiting the symplectic condition. See section 3.5. When the ordering

$$z = (X, P_x, Y, P_y, \tau, P_\tau) \tag{6.18.1}$$

is employed for the phase-space coordinates, J takes the form

$$J = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{pmatrix}. \tag{6.18.2}$$

6.19 Random Elements

Type Code: r**** where **** is the type code for the corresponding standard element.

Required Parameters:

1. ISOURCE
   = $-J$ (with J an integer from 1 to 9) to take parameters from a parameter set associated with the type code *psj*.
   = NFILE (with NFILE≥1) to read parameters from the file unit NFILE.
2. IECHO
   = 0 not to echo back parameter set values.
   = 1 to write parameter set values at the terminal.
   = 2 to write parameter set values on file 12.
   = 3 to write parameter set values at the terminal and on file 12.

Example:

```
badsex    rsext
 17          1
```

The above specifies a random sextupole with user given name *badsex*. The parameters for this element are to be read from file 17.

Description:

An idealized lattice generally has a relatively small number of elements. However, in a real lattice, every element may be different due to errors. Consequently, the description of a large real lattice with errors requires a large number of parameters.

The elements of a large lattice may be specified individually in #menu by employing individual user specified names and parameter values. However, it is also possible to read element parameters from auxiliary external files. This may be done for most beam line elements by using the corresponding *random type code mnemonic*. Random type code mnemonics are obtained by prefixing the corresponding standard type code mnemonic by the letter *r*. Listed below are the standard and random mnemonics for elements having this feature:

Standard Mnemonic	Random Mnemonic
arot	rarot
cfbd	rcfbd
cfqd	rcfqd
cfrn	rcfrn
cplm	rcplm
dism	rdism
dp	none
drft	rdrft
frng	rfrng
gbdy	rgbdy
gbnd	rgbnd
jmap	none
mark	none
nbnd	rnbnd
octe	rocte
octm	roctm
pbnd	rpbnd
prot	rprot
quad	rquad
recm	rrecm
sext	rsext
sol	rsol
spce	rspce
srfc	rsrfc
thlm	rthlm
twsm	rtwsm
usr1. . . usr20	rusr1. . . rusr20

A random element is described by giving a user specified name and random type code mnemonic on one line followed by the number of a parameter data file on a second line. See the example below:

```
badsex    rsext
17        1
```

This describes a random sextupole with user specified name *badsex*. Each time MaryLie makes use of this element, it will read file 17 to obtain the corresponding set of required parameter values. The format of the parameter data file should be the same as the parameter format for a standard element. In this case (see section 6.10), file 17 should contain a list of the form shown below:

$$\begin{array}{ccc} .50 & , & 3.5 \\ .49 & , & 3.7 \\ .51 & , & 3.8 \\ \vdots & & \vdots \end{array}$$

The first number in each line is a sextupole length, and the second is a sextupole strength. Note that file 17 is read (without rewinding) each time the element *badsex* is used. Thus, the user must be careful to be sure that successive lines in file 17 occur in the order in which they are needed.

If desired, use may be made of several parameter data files simultaneously. Thus, for example, random quadrupole strengths might be stored in file 19. See section 4.2. The menu could then contain an entry of the form shown below:

```
badquad     rquad
19          1
```

This defines a random quadrupole with the user specified name *badquad*. In this case (see section 6.10), file 19 should contain a list of the form

$$\begin{array}{cccccccc} .50 & , & 1.92 & , & 1 & , & 1 \\ .51 & , & 1.93 & , & 1 & , & 1 \\ .51 & , & 1.91 & , & 1 & , & 1 \\ \vdots & & \vdots & & \vdots & & \vdots \end{array}$$

Here the first number in each line is a quadrupole length, the second is a quadrupole strength, and the remaining 1,1 indicate that each quadrupole has leading and trailing hard-edge fringe fields.

MaryLie may come to the end of a random parameter file, such as file 17, during the course of a run. This could occur, for example, in a tracking run with a *loop*. When this happens, MaryLie rewinds the random parameter file and resumes reading from the beginning.

Upon occasion it may be useful to draw the parameters for an element from a parameter set. See section 7.25. For example, one may wish to tie the parameter values of various elements together. This can be done by taking their parameter values from a common parameter set. Note that, if desired, these common parameter values can be obtained from a random list on some file. See section 7.26. The example below shows the use of a parameter set:

```
  badpar      ps3
.50   3.5   0 0 0 0

  badsex      rsext
   -3           1
```

This example describes a random sextupole with user specified name *badsex*. Each time MaryLie makes use of this element, it will look at parameter set 3 to obtain the needed parameter values, which in this case are .50 for the length and 3.5 for the strength. Note that the command *badpar* must be invoked in #labor before *badsex* is used in order for these parameter values to be initialized. Note also that, to fulfill format requirements, the parameter set must have six parameter values even though, in this case, only the first two are used.

6.20 User Specified Subroutine that acts on phase space data

Type Code: usr1, usr2, $\cdots$ usr10

Required Parameters:

1. P1
2. P2
3. P3
4. P4
5. P5
6. P6

Example:

```
foil    usr1
.1 , .2 , .8 , 0 , 0 , 0
```

This specifies a user written subroutine with the user given name *foil*. It is to do its calculations using the parameter values .1, .2, .8, 0, 0, 0.

Description:

To be documented by user. Shown below is the place where the user can put his/her own code in user1. The include file rays.inc specifies the array where phase space data is stored. The parameters P1 through P6 are available for the user to employ, and can be varied by the *vary* command in connection with fitting and optimization routines. See section 9.6.

```
       subroutine user1(p)
c
c This subroutine .....
c
      include 'impli.inc'
      include 'files.inc'
      include 'rays.inc'
      include 'parset.inc'
c
      dimension p(6)
c
c User supplied code goes here.
c
      return
      end
```

6.21 User Specified Subroutine that produces or acts on a map

Type Code: usr11, usr12, $\cdots$ usr20

Required Parameters

1. P1
2. P2
3. P3
4. P4
5. P5
6. P6

Example:

```
mymap    usr11
 1 , 1 , 0 , 0 , 0 , 0
```

This specifies a user written subroutine with the user given name *mymap*. It is to do its calculations using the parameters values 1, 1, 0, 0, 0, 0.

Description:

To be documented by user. Shown below is the place where the user can put his/her own code in user11. The calling arrays fa and fm contain the current transfer map. The parameters P1 through P6 are available for the user to employ, and can be varied by the *vary* command in connection with fitting and optimization routines. See section 9.6.

```
      subroutine user11(p,fa,fm)
c
c  This subroutine ....
c
      include 'impli.inc'
      include 'usrdat.inc'
      include 'mldex.inc'
c
      dimension p(6)
      dimension fa(*), fm(6,6)
c
c User supplied code goes here.
c
      return
      end
```

6.22 Dispersion Matrix

Type Code: dism

Required Parameters:

1. R_{16}
2. R_{26}
3. R_{36}
4. R_{46}
5. R_{56}

Example:

```
dismat   dism
 .1 , .2 , 0 , 0 , 0
```

The above specifies a map with user given name *dismat*. The diagonal entries of the matrix part R are 1. The off diagonal entries R_{16} and R_{26} have the values .1 and .2, respectively. The map has no nonlinear part.

Description:

This type code produces a linear map with matrix part

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & R_{16} \\ 0 & 1 & 0 & 0 & 0 & R_{26} \\ 0 & 0 & 1 & 0 & 0 & R_{36} \\ 0 & 0 & 0 & 1 & 0 & R_{46} \\ -R_{26} & R_{16} & -R_{46} & R_{36} & 1 & R_{56} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6.22.1}$$

In Lie algebraic language, it produces the map

$$M = e^{:f_2:}$$

with

$$f_2 = (R_{26})XP_\tau - (R_{16})P_xP_\tau + (R_{46})YP_\tau - (R_{36})P_yP_\tau - (1/2)(R_{56})P_\tau^2.$$

This map has only dispersion and time of flight entries. Note that the entries in the fifth row of R are determined by the entries in the sixth column as a result of the symplectic condition. That is, if a map has dispersion, the time of flight must be trajectory dependent, and vice versa. See also section 7.38.

6.23 Solenoid

Type Code: sol

Required Parameters:

1. zi (initial longitudinal coordinate, in meters).
2. zf (final longitudinal coordinate, in meters).
3. NS (number of integration steps).
4. IPROFILE
   = 0 to not write out field profile.
   = IFILE to write profile on file IFILE.
   Note: If IFILE< 0, the effect is the same as when IFILE> 0 except that only the profile is computed and the numerical integration required to compute the map specified by *sol* is bypassed. Instead, the identity map is returned. This feature makes it possible to compute profiles, if that is all that is desired, without spending time on unnecessary numerical integration.
5. IPSET (an integer from 1 to 9).
6. MULTIPOLES = 0 (not currently used, but must be present).

This element also requires additional parameters whose values are specified by the parameter set IPSET. (See section 7.25.) The contents of parameter set IPSET are as follows:

1. P1 = DI (length of initial drift space in front of solenoid, starting from $z = 0$, in meters).
2. P2 = Length of body of solenoid, in meters.
3. P3 = Characteristic length *cl* associated with leading and trailing fringe-field fall off, in meters.
4. P4 = Strength B of solenoid, if it were infinitely long (Tesla).
5. IECHO
   =0 to not echo back parameter values.
   =1 to write parameter values at the terminal.
   =2 to write parameter values on file 12.
   =3 to write parameter values at the terminal and on file 12.
6. JOB
   = 1 to compute full transfer map for air-core model.
   = -1 to compute the same transfer map with the paraxial rotation part removed.
   = 2 to compute full transfer map for iron-dominated model.
   =-2 to compute the same transfer map with the paraxial rotation part removed.

Example:

```
sol      sol
 0 , 2 , 200 , 10 , 1 , 0
```

The above specifies a solenoid with user given name *sol*. Numerical integration begins at $z = 0$ meters, and ends at 2 meters. Two hundred integration steps are used. Field profiles are written on file 10. Additional parameters are taken from parameter set 1. Suppose the parameter values in parameter set 1 have been set with the following command:

```
  solpar   ps1
.5 , 1 , .1 , 3 , 3 , 2
```

Then the solenoid is preceded by an initial drift (extended leading fringe field region) of .5 meters. The solenoid itself has a length of 1 meter. The characteristic length associated with the leading and trailing fringe-field fall off is .1 meter. The strength of the solenoid is 3 Tesla. Parameter values are to be echoed back at the terminal and on file 12. Finally, the full transfer map is to be computed using the iron-dominated model.

Since the region of integration extends to $z = 2$ meters, the solenoid in this case is also effectively followed by an extended trailing fringe-field region of .5 meters ($\mathrm{zf} - \mathrm{P1} - \mathrm{P2} = 2 - .5 - 1 = .5$).

Description:

The type code *sol* causes the production of the transfer map for a solenoid. This is done by numerical integration using a special purpose built-in GENMAP routine. (See section 1.4.1). The accuracy of the numerical integration procedure depends on the number NS of integration steps employed and the spatial rate of variation of the magnetic field. The more rapidly the field varies, the more steps are required. Therefore, the user should try several values for NS to be sure that sufficient accuracy has been achieved. Increasing NS improves the accuracy, but also increases the computation time.

The specification of the magnetic field of a solenoid, including fringe fields, requires fairly complicated mathematical expressions for all three components of $\mathbf{B}(x, y, z)$. However, it can be shown that if $B_z(0, 0, z)$ (the on-axis longitudinal component of $\mathbf{B}$) is known, then Maxwell's equations plus the requirement of axial symmetry are sufficient to determine all components of $\mathbf{B}$ everywhere. In particular, $\mathbf{B}$ can be obtained from a vector potential $\mathbf{A}$ given (through fourth order) by the expressions

$$\begin{aligned} A_x &= -yU, \\ A_y &= xU, \\ A_z &= 0, \end{aligned}$$

where

$$U = \frac{1}{2}B_z(0, 0, z) - \frac{1}{16}(x^2 + y^2)B_z''(0, 0, z).$$

Here B_z'' is the second derivative of B_z with respect to z. The x, y, z coordinates are shown in figure 6.23.1.

Figure 6.23.1: Coordinate system for the solenoid.

For simplicity, MaryLie currently provides two soft-edge "bump" function models for the on-axis longitudinal field of the form

$$B_z(0,0,z) = B \text{ bump}(z, cl, L).$$

As described below, one corresponds to a simple air-core solenoid with a single winding, and the other models an iron-dominated solenoid. Both have physically reasonable properties that should be adequate for at least preliminary calculations. However, if the user wishes to make detailed calculations for a particular solenoid, he or she may wish to modify various MaryLie subroutines to incorporate an improved model.

The soft-edge bump function has the properties

$$\begin{gathered}\text{bump}(z, cl, L) \simeq 1 \text{ for } z \in [0, L], \\ \text{bump}(z, cl, L) \simeq 0 \text{ elsewhere}, \\ \text{bump}(L/2 + w, cl, L) = \text{bump}(L/2 - w, cl, L), \\ \textstyle\int_{-\infty}^{\infty} \text{bump}(z, cl, L) dz = L.\end{gathered}$$

Here L is the length of the body of the solenoid. In particular, from the results above one has the relation

$$\int_{-\infty}^{\infty} B_z(0,0,z) dz = BL.$$

Specifically, the soft-edge bump function is defined in terms of an approximating "signum" function $\text{sgn}(z, cl)$ by the relation

$$\text{bump}(z, cl, L) = [\text{sgn}(z, cl) - \text{sgn}(z - L, cl)]/2.$$

The approximating signum function is in turn defined by two possible relations. In the air-core model the relation is

$$\text{sgn}(z, cl) = z/[z^2 + (cl)^2]^{1/2}.$$

This model describes the case of a single-layer solenoid of length L, radius cl, and interior field B (in the infinite-length limit). In this case the on-axis field falls off as $L(cl)^2/|z|^3$ for large distances, and the characteristic fall-off length is governed by cl. In the iron-dominated model the relation is

$$\text{sgn}(z, cl) = \tanh(z/cl).$$

In this case the on-axis field falls of exponentially as $\exp(-2|z/cl|)$ for large $|z|$, and the characteristic fall-off length is again governed by cl. This model does not correspond to any easily-described iron and current distribution. However, it does have the property of exponential fall off, which is characteristic of all iron-dominated solenoids.

In both cases, the approximating signum function becomes the true signum function in the limit that the characteristic length cl goes to zero,

$$\lim_{cl \to 0} \text{sgn}(z, cl) = \text{sgn}(z).$$

Recall that the true signum function has the definition

$$\begin{aligned} \mathrm{sgn}(z) &= 1 \text{ if } z > 0, \\ \mathrm{sgn}(z) &= 0 \text{ if } z = 0, \\ \mathrm{sgn}(z) &= -1 \text{ if } z < 0. \end{aligned}$$

In this same limit the soft-edge bump function becomes the hard-edge bump function,

$$\lim_{cl \to 0} \mathrm{bump}(z, cl, L) = \mathrm{bump}(z, L).$$

The hard-edge bump function has the properties,

$$\begin{aligned} \mathrm{bump}(z, L) &= 1 \text{ for } z \in (0, L), \\ \mathrm{bump}(0, L) &= \mathrm{bump}(L, L) = 1/2, \\ \mathrm{bump}(z, L) &= 0 \text{ elsewhere.} \end{aligned}$$

It follows that the characteristic length cl (=P3) controls the rate of fall off of the fringe fields. The fringe-field region is large if cl is large, and vanishes as cl goes to zero.

The quantities $B_z(0, 0, z)$ and $B_z''(0, 0, z)$ are written on the file IPROFILE. This file is formatted into six columns as follows:

- Column 1 contains z.
- Column 2 contains $B_z(0, 0, z)$.
- Column 3 contains $B_z''(0, 0, z)$.
- Column 4 contains the number 0.
- Column 5 contains the number 0.
- Column 6 contains the number 0.

As a result of the formatting the contents of IPROFILE can be plotted by any plot routine capable of plotting MaryLie phase-space data. See section 4.5.2.

Exhibit 6.23.1 shows the use of a solenoid along with two drifts to form a simple imaging system as depicted schematically in figure 6.23.2. It also gives an example of the production of field profile plots. In particular, figures 6.23.3 and 6.23.4 show the field profiles $B_z(0, 0, z)$ and $B_z''(0, 0, z)$ for the solenoid of Exhibit 6.23.1.

Since both drifts have the same length, the imaging system should have unit magnification, and the image should be inverted. In addition, the image should be slightly rotated due to the spiral nature of charged particle trajectories in a longitudinal magnetic field. Indeed, it can be shown that the rotation angle θ_r is given by the relation

$$\theta_r = 1/2\mathrm{brho} \int_{-\infty}^{\infty} B_z(0, 0, z) dz = BL/(2\mathrm{brho}).$$

For positive particles the rotation is counter-clockwise in the context of figure 6.14.1. Correspondingly, it is clockwise in the context of figure 6.14.3. That is, in the terminology of

section 6.14, the rotation is equivalent to *arot*$(-\theta_r)$. Caution: In the formula above, θ_r is given in radians. By contrast, the type code *arot* requires an argument given in degrees.

Sometimes it is useful to artificially remove this rotation from the transfer map. (This rotation couples the horizontal and vertical phase spaces, which may unduely complicate fitting or some other calculation.) In that case, one should use negative integer values for the parameter JOB.

For further detail, see references listed in Chapter 11.

```
#comment
 Exhibit 6.23.1.
 This is a MARYLIE run illustrating use of the solenoid type code.
 A solenoid is combined with two drifts to produce an imaging system.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace    rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 dr       drft
   22.8417247128300
 sol      sol
  0.000000000000000E+00   2.00000000000000       200.000000000000
   10.0000000000000       1.00000000000000      0.000000000000000E+00
 solpar   ps1
  0.500000000000000       1.00000000000000      0.100000000000000
   3.00000000000000       3.00000000000000      2.000000000000000E+00
 end      end
#lines
 lens
     1*solpar      1*sol
 image
     1*dr          1*lens        1*dr
#lumps
#loops
#labor
    1*fileout
    1*image
    1*raysin
    1*trace
    1*end

 zi=  0.0000000000000000E+00 zf=   2.000000000000000
 ns=         200
 di=  0.5000000000000000     length=   1.000000000000000
 cl=  0.1000000000000000     B=   3.000000000000000
 job=          2
```

Figure 6.23.2: Schematic of the imaging system of Exhibit 6.23.1.

Figure 6.23.3: Profile of $B_z(0,0,z)$, (Tesla), for the solenoid of Exhibit 6.23.1. Parameter values are DI=.5, $L = 1$, $cl = .1$, and $B = 3$.

Figure 6.23.4: Profile of $B_z''(0,0,z)$, (Tesla/m^2), for the solenoid of Exhibit 6.23.1. Parameter values are DI=.5, $L = 1$, $cl = .1$, and $B = 3$.

How well does this system work in practice? Suppose the word MARYLIE, as shown in figure 2.3.4, is used as an object. Figure 6.23.5 shows the image produced. Evidently the image is inverted and rotated as expected. It can also be shown that there is unit magnification near the origin. However, it is evident that there are very severe aberrations away from the origin! Indeed, these aberrations are so severe that third-order results alone may be insufficient to describe the outer parts of the image.

It can be shown that the third and higher order aberrations of a solenoid become infinite in the hard-edge limit (cl goes to zero). Thus, the hard-edge limit is unphysical for a solenoid. Correspondingly, third-order solenoid aberrations can be reduced by making the fringe-field region large. It also helps to make the solenoid weak since aberrations are proportional to B. If this is done, the solenoid must also be made long (to compensate for the small B) in order to maintain the desired paraxial properties.

Figure 6.23.5: Image of the word MARYLIE produced by the simple imaging system of Exhibit 6.23.1.

6.24 Combined Function Magnetic Quadrupole

Type Code: cfqd

Required Parameters:

1. Length of quadrupole (m).
2. IPSET
   = 0 to set all multipole coefficients to zero.
   = J (with J an integer from 1 to 9) to take multipole coefficient values from the parameter set associated with the type code *psj*. (See section 7.25.) In this case the parameters P1...P6 from the parameter set associated with *psj* are used as follows:
   P1 = BQD (Tesla/m) (normal)
   P2 = AQD (Tesla/m) (skew)
   P3 = BSEX (Tesla/m^2) (normal)
   P4 = ASEX (Tesla/m^2) (skew)
   P5 = BOCT (Tesla/m^3) (normal)
   P6 = AOCT (Tesla/m^3) (skew)
3. LFRN
   = 0 for no fringe field on leading edge (entry) of quadrupole.
   = 1 for hard-edge fringe field on leading edge of quadrupole.
4. TFRN
   = 0 for no fringe field on trailing edge (exit) of quadrupole.
   = 1 for hard-edge fringe field on trailing edge of quadrupole.

Example:

```
allcfqd   cfqd
.5 , 1 , 1 , 1
```

The above specifies a combined function magnetic quadrupole with user given name *allcfqd*. It has a length of .5 meters and has hard-edge leading and trailing fringe fields. Multipole coefficient values are taken from parameter set 1. Suppose the parameter values in parameter set 1 have been set by previously invoking in #labor the following command:

```
   allpar      ps1
2.5 , .2 , .3 , .4 , .5 , .6
```

Then the normal quad strength is 2.5 Tesla/m, and, in the absence of other multipoles, the quadrupole would be horizontally focusing. However, there are also other multipoles: the skew quad strength is .2 Tesla/m, the normal sextupole strength is .3 Tesla/m^2, the skew

sextupole strength is .4 Tesla/m^2, the normal octupole strength is .5 Tesla/m^3, and the skew octupole strength is .6 Tesla/m^3.

Description:

A combined function quadrupole is a quadrupole with added sextupole and octupole components. Thus, pure quadrupoles, pure sextupoles, and pure octupoles are special cases of the combined function quadrupole. As illustrated in Exhibit 6.24.1, the combined function quadrupole is, in turn, a special limiting case of the combined function bend.

```
#comment
 Exhibit 6.24.1.
 This is a MARYLIE run illustrating the sign conventions for
 combined function magnetic quadrupoles.  A combined function
 magnetic quadrupole is a quadrupole with added sextupole and
 octupole components.  Thus, pure quadrupoles, pure sextupoles,
 and pure octupoles are special cases of the combined function
 quadrupole.  In this run a comparison is made with the combined
 function bend in the limit of no dipole field.  Note that to
 obtain a finite length element without dipole content using
 the cfbd element, it is necessary to make both the dipole field
 small and the bend angle small in such a way that the design orbit
 path length as given in figure 6.2a takes on the desired value.
 For simplicity of calculation the value of brho has been set
 equal to 10/pi.  Thus, the path length is given by the relation

  path length = 10/180 * theta (in degrees) /B.

 In particular, if

      theta = 9.e-9 and B = 1.e-9,
 then

       path length = .5 meters.

#beam
  3.183098861830000
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-08  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 wcfbd    cfbd
  9.000000000000000E-09  1.000000000000000E-09   1.00000000000000
   1.00000000000000       1.00000000000000       1.00000000000000
 fcfbdpar ps1
   5.00000000000000       2.50000000000000       20.0000000000000
   30.0000000000000       40.0000000000000       50.0000000000000
 dcfbdpar ps1
  -5.00000000000000       2.50000000000000       20.0000000000000
   30.0000000000000       40.0000000000000       50.0000000000000
 cfqd    cfqd
  0.500000000000000       1.00000000000000       1.00000000000000
   1.00000000000000
 clear    iden
 inv      inv
 end      end
```

```
#lines
 tfcfbd
     1*fcfbdpar    1*wcfbd
 tdcfbd
     1*dcfbdpar    1*wcfbd
 tfcfqd
     1*fcfbdpar     1*cfqd
 tdcfqd
     1*dcfbdpar     1*cfqd
 test
     1*clear       1*tfcfbd      1*inv         1*tfcfqd      1*mapout   &
     1*clear       1*tdcfbd      1*inv         1*tdcfqd      1*mapout
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*test
    1*end
```

Results of test

```
matrix for map is :

  1.00000E+00  1.24184E-12  1.27068E-16 -3.98986E-17  0.00000E+00  4.51475E-11
 -1.95069E-12  1.00000E+00  9.76028E-13 -9.71445E-17  0.00000E+00  1.74682E-10
 -9.36751E-17  1.64799E-17  1.00000E+00  1.24192E-12  0.00000E+00  7.63386E-13
  9.74838E-13  6.24500E-17  1.95086E-12  1.00000E+00  0.00000E+00  6.10779E-12
 -1.74682E-10  4.51475E-11 -6.10779E-12  7.63386E-13  1.00000E+00  5.10553E-13
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

 nonzero elements in generating polynomial are :


 matrix for map is :

  1.00000E+00  1.24191E-12 -9.58435E-17  1.73472E-17  0.00000E+00  4.82011E-11
  1.95084E-12  1.00000E+00  9.74811E-13  6.76542E-17  0.00000E+00  1.99113E-10
  1.17961E-16 -3.90313E-17  1.00000E+00  1.24183E-12  0.00000E+00  7.63386E-13
  9.76004E-13 -8.50015E-17 -1.95068E-12  1.00000E+00  0.00000E+00  6.10779E-12
 -1.99113E-10  4.82011E-11 -6.10779E-12  7.63386E-13  1.00000E+00  5.10553E-13
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :
```

6.25 Marker

Type Code: mark

Required Parameters: None

Example:

```
spot1      mark
```

This specifies a marker with user given name *spot1*.

Description:

These elements have no direct effect in a MaryLie run. They simply afford the user the opportunity to give names to various locations in a lattice or beam line.

6.26 Data Point

Type Code: dp

Required Parameters: None

Example:

```
point1      dp
```

Description:

These elements are recognized by the *geom* type code and are used for producing graphics. See sections 8.38, 10.9, and 10.11 through 10.14.

6.27 Rare Earth Cobalt (REC) Quadrupole Multiplet

Type Code: recm

Required Parameters:

1. zi (initial longitudinal coordinate in meters).
2. zf (final longitudinal coordinate in meters).
3. NS (number of integration steps).
4. IPROFILE
   = 0 to not write out field profile.
   = IFILE to write profile on file IFILE.
   Note: If IFILE< 0, the effect is the same as when IFILE> 0 except that only the profile is computed and the numerical integration required to compute the map specified by *recm* is bypassed. Instead, the identity map is returned. This feature makes it possible to compute profiles, if that is all that is desired, without spending time on unnecessary numerical integration.
5. MULTIPOLES
   = 0 to set all higher multipole coefficients to zero.
   = index (an integer from 1 to 9) of parameter set containing multipole strength information for the region of integration zi,zf.
6. NPAT (index of parameter set containing quadrupole pattern information, an integer from 1 to 9).

This element also requires additional parameters whose values are specified by the parameter sets with indices MULTIPOLES, NPAT, and IPS, IPS+1,. . . , (IPS+NTSEC−1). See section 7.25. The contents of the parameter set with index MULTIPOLES are as follows:

1. BSEX (Tesla/m^2). (normal)
2. ASEX (Tesla/m^2). (skew)
3. BOCT (Tesla/m^3). (normal)
4. AOCT (Tesla/m^3). (skew)
5. 0 (not used, but must be present).
6. 0 (not used, but must be present).

The contents of the parameter set with index NPAT control and describe the pattern of quadrupoles as follows:

1. DI (length of initial drift space in front of pattern, starting from $z = 0$, in meters).

2. NTSEC (number of types of quadrupole sections to be described by additional parameter sets). Note: This number must currently be limited to be less than or equal to 9.

3. IPS (index of first parameter set describing the first section of quadrupoles). Subsequent parameter sets used to describe additional sections have indices IPS+1...(IPS+NTSEC−1).

4. MAXSEC (maximum number of sections to be used in quadrupole pattern or repeated quadrupole pattern).

5. NCYC (number of cycles through pattern, except that the sequence of sections terminates when the number of sections employed reaches MAXSEC).

6. ISEND

   =0 to not write out pattern.

   =1 to write pattern at the terminal.

   =2 to write pattern on file 12.

   =3 to write pattern at the terminal and on file 12.

Finally, the contents of the NTSEC parameter sets with indices IPS ...(IPS+NTSEC−1) define the type and number of quadrupoles in each section as follows:

1. Length L of quadrupole (m).

2. Strength Q of quadrupole, if it were infinitely long (Tesla/m).

3. Inner radius (m).

4. Outer radius (m).

5. Length of trailing drift space after quadrupole (m). Note that the length assigned to the trailing drift after the *last* quadrupole in the whole pattern or repeated pattern is ignored. The integration always stops at zf.

6. NQUAD [number of (identical) quadrupoles that appear in the section].

Example:

See Exhibits 6.27.1 through 6.27.3 and figures 6.27.1 through 6.27.3.

Description:

The type code *recm* has great flexibility at the cost of some complexity. It causes the production of the transfer map for a part or all of a collection of REC quadrupoles that

will be referred to as "whole". This is done by numerical integration over the interval from zi to zf in NS steps using a special purpose built-in GENMAP routine. (See section 1.4.1). Superimposed on the interval zi,zf is a (piece-wise constant) multipole (sextupole plus octupole) magnetic field.

The accuracy of the numerical integration procedure depends on the number NS of integration steps employed and the spatial rate of variation of the magnetic field. The more rapidly the field varies, the more steps are required. Therefore, the user should try several values for NS to be sure that sufficient accuracy has been achieved. Increasing NS improves the accuracy, but also increases the computation time.

The contents of "whole" is equivalent to the following sequence of MaryLie lines:

```
#lines
whole
 drin   rpattern
        (drin  is an initial drift with length DI; rpattern  refers to
        "repeated pattern" as defined in the next line)
rpattern
 NCYC*pattern
pattern
 sect1  sect2 ··· sectNTSEC
sect1
 NQUAD1*pair1
sect2
 NQUAD2*pair2
         ⋮
sectNTSEC
 NQUADNTSEC*pairNTSEC
pair1
 quad1  tdr1
pair2
 quad2  tdr2
         ⋮
pairNTSEC
 quadNTSEC    tdrNTSEC
        (here quadj and tdrj refer to the quadrupole and trailing drift
        pair in the jth section)
```

However, it is to be understood that all of *rpattern* is not used if NCYC*NTSEC exceeds MAXSEC. In this case, only the first MAXSEC sections in *rpattern* are used. Also, it should be understood that the magnetic field at any integration point is the superimposed multipole field plus the sum of all REC quadrupole fields in the pattern, repeated pattern, or MAXSEC sections (whichever is applicable). Thus, the type code *recm* includes the effect of overlapping fringe fields. Finally, the ability to superimpose multipole fields makes it

possible, if desired, to add (piece-wise constant) sextupole and octupole content to each REC quadrupole.

It can be shown that the vector potential for a quadrupole configuration is (through fourth order) of the form

$$\begin{aligned} A_x &= \frac{g'(z)}{4}(x^3 - xy^2), \\ A_y &= -\frac{g'(z)}{4}(y^3 - x^2y), \\ A_z &= -\frac{g(z)}{2}(x^2 - y^2) + \frac{g''(z)}{12}(x^4 - y^4). \end{aligned}$$

Here $g(z)$ is the on-axis field gradient, and the quantities $g'(z)$ and $g''(z)$ are derivatives of g with respect to z.

For a single REC quadrupole of length L and strength Q with entrance face at $z = 0$ (and exit face at $z = L$), it can be shown that $g(z)$ has the following properties:

$$\begin{aligned} g(z) &\simeq Q \text{ for } z \in [0, L], \\ g(z) &\simeq 0 \text{ elsewhere}, \\ g(\frac{L}{2} + w) &= g(\frac{L}{2} - w), \\ \int_{-\infty}^{\infty} g(z)dz &= QL. \end{aligned}$$

For a description of REC quadrupoles and the computation of $g(z)$, see the reference to the work of Halbach in Chapter 11.

The quantities g, g', g'' as a function of z are written on file IPROFILE. This file is formatted into six columns as follows:

1. Column 1 contains z.
2. Column 2 contains $g(z)$.
3. Column 3 contains $g'(z)$.
4. Column 4 contains $g''(z)$.
5. Column 5 contains the number 0.
6. Column 6 contains the number 0.

As a result of this formatting the contents of IPROFILE can be plotted by any plot routine capable of plotting MaryLie phase-space data. See section 4.5.2.

Figures 6.27.1 through 6.27.3 show the field profiles g, g', and g'' for the soft-edge REC quadrupole singlet of Exhibit 6.27.1. Figure 6.27.4 shows the gradient g for the REC quadrupole singlet of Exhibit 6.27.1 when the inner radius is very small. In this case the profile is essentially that of a hard-edge quadrupole.

Figure 6.27.1: Profile of $g(z)$, (Tesla/m), for the soft-edge REC quadrupole singlet of Exhibit 6.27.1.

Figure 6.27.2: Profile of $g'(z)$, (Tesla/m^2), for the soft-edge REC quadrupole single of Exhibit 6.27.1.

Figure 6.27.3: Profile of $g''(z)$, (Tesla/m^3), for the soft-edge REC quadrupole singlet of Exhibit 6.27.1.

Figure 6.27.4: Profile of $g(z)$, (Tesla/m), for the very small bore REC quadrupole singlet of Exhibit 6.27.1. In this case, the profile is essentially that of a hard-edge quadrupole.

Figures 6.27.5 and 6.27.6 and Exhibits 6.27.2 and 6.27.3 illustrate the use of the type code *recm* to produce REC quadrupole doublets and triplets. Should the user wish to treat more complicated REC quadrupole arrays, she or he is encouraged to check that the proper gradient profile is indeed being produced.

Figure 6.27.5: Profile of $g(z)$, (Tesla/m), for the soft-edge REC quadrupole doublet of Exhibit 6.27.2.

Figure 6.27.6: Profile of $g(z)$, (Tesla/m), for the soft-edge REC quadrupole triplet of Exhibit 6.27.3.

```
#comment
 Exhibit 6.27.1.
 This is a MARYLIE input file to test a REC quadrupole singlet.
 First the map is computed for a typical REC quadrupole that has
 soft edges.  Next the inner radius is made very small so that
 the edges in effect become hard, and comparison is made with the
 the map for the standard MARYLIE hard-edge quadrupole to show
 that in the hard-edge limit the two maps agree.  Comparison is
 also made with a standard MARYLIE quadrupole with hard-edge
 fringe fields turned off to show that these effects are important.
 Note that the limiting process is quite delicate.  The integration
 region has to be broken up into separate pieces, and a large
 number of steps is required in the fringe field regions where
 the field is changing very rapidly.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-14  0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 srecm    recm
  0.000000000000000E+00   1.00000000000000       1000.00000000000
   20.0000000000000       1.00000000000000       2.00000000000000
 ldrecm   recm
  0.000000000000000E+00  0.240000000000000       1000.00000000000
   22.0000000000000       1.00000000000000       2.00000000000000
 lffrecm  recm
  0.240000000000000      0.260000000000000       10000.0000000000
   22.0000000000000       1.00000000000000       2.00000000000000
 bdrecm   recm
  0.260000000000000      0.740000000000000       1000.00000000000
   22.0000000000000       1.00000000000000       2.00000000000000
 tffrecm  recm
  0.740000000000000      0.760000000000000       10000.0000000000
   22.0000000000000       1.00000000000000       2.00000000000000
 tdrecm   recm
  0.760000000000000       1.00000000000000       1000.00000000000
   22.0000000000000       1.00000000000000       2.00000000000000
 mpoles   ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 npatps   ps2
  0.250000000000000       1.00000000000000       3.00000000000000
   1.00000000000000       1.00000000000000       3.00000000000000
 npatpsq  ps2
```

```
   0.250000000000000       1.00000000000000       3.00000000000000
    1.00000000000000       1.00000000000000      0.000000000000000E+00
  stype1   ps3
   0.500000000000000       3.00000000000000      0.100000000000000
   0.700000000000000      0.250000000000000       1.00000000000000
  htype1   ps3
   0.500000000000000       3.00000000000000      1.000000000000000E-04
   0.700000000000000      0.250000000000000       1.00000000000000
  dr       drft
   0.250000000000000
  hfqwff   quad
   0.500000000000000       3.00000000000000       1.00000000000000
    1.00000000000000
  hfqwoff  quad
   0.500000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00
  clear    iden
  inv      inv
  stm      stm
    1.00000000000000
  gtm      gtm
    1.00000000000000       1.00000000000000
  end      end
 #lines
  asrecm
      1*mpoles      1*npatps      1*stype1      1*srecm
  ahrecm
      1*mpoles      1*npatpsq     1*htype1      1*ldrecm      1*lffrecm  &
      1*bdrecm      1*tffrecm     1*tdrecm
  stest
      1*clear       1*asrecm      1*mapout
  htest
      1*clear       1*ahrecm      1*mapout      1*inv         1*stm      &
      1*clear       1*dr          1*hfqwoff     1*dr          1*gtm      &
      1*mapout      1*clear       1*dr          1*hfqwff      1*dr       &
     1*gtm         1*mapout
#lumps
#loops
#labor
    1*fileout
    1*setzero
    1*stest
    1*htest
    1*end

Integrating in   1000 steps from   0.00000=zi to   1.00000=zf
multipole strengths from pset 1:
 Sextupole:  0.0000E+00  0.0000E+00
 Octupole:   0.0000E+00  0.0000E+00
Pattern from pset 2:
  1 cycle(s) of  1 type(s) of section(s) with a maximum of  1 section(s)
 di=   0.25000
 types of section(s) are:
```

```
pset  length    strength   radii: inner      outer      tdrift    number
  3    0.500000    3.000000    0.100000    0.700000    0.250000     1
```

Map for REC quadrupole singlet

```
matrix for map is :

 8.51904E-01  9.34965E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-2.93337E-01  8.51904E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.15449E+00  1.06709E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.11911E-01  1.15449E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.11143E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-0.20650905671254D-01
 f( 38)=f( 11 00 01 )=-0.15679802699651D+00
 f( 53)=f( 02 00 01 )=-0.52119059187004D+00
 f( 67)=f( 00 20 01 )=-0.22724186694284D-01
 f( 70)=f( 00 11 01 )= 0.20430893727797D+00
 f( 76)=f( 00 02 01 )=-0.67847222509590D+00
 f( 83)=f( 00 00 03 )=-0.24420182415497D+00
 f( 84)=f( 40 00 00 )=-0.28227598748729D-01
 f( 85)=f( 31 00 00 )= 0.61418807395008D-01
 f( 90)=f( 22 00 00 )=-0.45953894062225D-01
 f( 95)=f( 20 20 00 )=-0.68166016385667D-01
 f( 96)=f( 20 11 00 )= 0.90851764997720D-01
 f( 99)=f( 20 02 00 )=-0.10292677698251D+00
 f(104)=f( 20 00 02 )=-0.27515802694512D-01
 f(105)=f( 13 00 00 )=-0.62031480712773D-01
 f(110)=f( 11 20 00 )= 0.88370184117920D-01
 f(111)=f( 11 11 00 )=-0.87903464652352D-01
 f(114)=f( 11 02 00 )= 0.23844472185938D-01
 f(119)=f( 11 00 02 )=-0.20029442871966D+00
 f(140)=f( 04 00 00 )=-0.92578217993770D-01
 f(145)=f( 02 20 00 )= 0.39532700351210D-01
 f(146)=f( 02 11 00 )= 0.33117887348238D-01
 f(149)=f( 02 02 00 )=-0.25827294412386D+00
 f(154)=f( 02 00 02 )=-0.66939257293787D+00
 f(175)=f( 00 40 00 )= 0.20193765407441D-01
 f(176)=f( 00 31 00 )=-0.23952603185415D-01
 f(179)=f( 00 22 00 )=-0.57353972049412D-01
 f(184)=f( 00 20 02 )=-0.31569580597948D-01
 f(185)=f( 00 13 00 )= 0.13997371868078D+00
 f(190)=f( 00 11 02 )= 0.29321220080138D+00
 f(195)=f( 00 04 00 )=-0.17434530307670D+00
 f(200)=f( 00 02 02 )=-0.97727123848868D+00
 f(209)=f( 00 00 04 )=-0.31122101123964D+00
```

Map for small diameter REC quadrupole singlet

```
matrix for map is :
```

```
 8.49278E-01  9.30654E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-2.99497E-01  8.49278E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.15663E+00  1.07151E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.15241E-01  1.15663E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.11143E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-0.22383724600551D-01
 f( 38)=f( 11 00 01 )=-0.15769160712296D+00
 f( 53)=f( 02 00 01 )=-0.51671870932605D+00
 f( 67)=f( 00 20 01 )=-0.24395388813471D-01
 f( 70)=f( 00 11 01 )= 0.20913639918336D+00
 f( 76)=f( 00 02 01 )=-0.68440637499314D+00
 f( 83)=f( 00 00 03 )=-0.24420182415497D+00
 f( 84)=f( 40 00 00 )=-0.12462596970828D-01
 f( 85)=f( 31 00 00 )= 0.40726292683912D-01
 f( 90)=f( 22 00 00 )=-0.16932946630259D-02
 f( 95)=f( 20 20 00 )=-0.94658638766057D-01
 f( 96)=f( 20 11 00 )= 0.11998933245921D+00
 f( 99)=f( 20 02 00 )=-0.11848758501364D+00
 f(104)=f( 20 00 02 )=-0.29888361271843D-01
 f(105)=f( 13 00 00 )=-0.98757989972425D-01
 f(110)=f( 11 20 00 )= 0.11635896310053D+00
 f(111)=f( 11 11 00 )=-0.11808182970568D+00
 f(114)=f( 11 02 00 )= 0.37792421430458D-01
 f(119)=f( 11 00 02 )=-0.20037039416016D+00
 f(140)=f( 04 00 00 )=-0.80929178377441D-01
 f(145)=f( 02 20 00 )= 0.35825922423949D-01
 f(146)=f( 02 11 00 )= 0.37934328439097D-01
 f(149)=f( 02 02 00 )=-0.26098469654250D+00
 f(154)=f( 02 00 02 )=-0.66106215344546D+00
 f(175)=f( 00 40 00 )=-0.20591479038237D-01
 f(176)=f( 00 31 00 )= 0.73514878474503D-01
 f(179)=f( 00 22 00 )=-0.19473344292996D+00
 f(184)=f( 00 20 02 )=-0.33821484838676D-01
 f(185)=f( 00 13 00 )= 0.22700117736392D+00
 f(190)=f( 00 11 02 )= 0.30097611645157D+00
 f(195)=f( 00 04 00 )=-0.19593148870190D+00
 f(200)=f( 00 02 02 )=-0.98932915776751D+00
 f(209)=f( 00 00 04 )=-0.31122101123964D+00
```

Comparison of maps for small diameter REC quadrupole singlet and standard MARYLIE quadrupole with hard edge fringe fields turned off.

```
matrix for map is :

 1.00000E+00 -1.56686E-06  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-5.66626E-06  9.99997E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  9.99999E-01  3.45027E-06  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.13207E-06  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -4.64212E-15
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-0.81161528916714D-06
 f( 38)=f( 11 00 01 )=-0.54767003357242D-05
 f( 53)=f( 02 00 01 )= 0.13594565761027D-05
 f( 67)=f( 00 20 01 )=-0.13029011691045D-05
 f( 70)=f( 00 11 01 )=-0.13392474697907D-05
 f( 76)=f( 00 02 01 )=-0.47180614553693D-05
 f( 84)=f( 40 00 00 )= 0.12107560465887D-01
 f( 85)=f( 31 00 00 )= 0.45110908045393D-01
 f( 90)=f( 22 00 00 )=-0.19684440564328D-01
 f( 95)=f( 20 20 00 )= 0.93878623381152D-01
 f( 96)=f( 20 11 00 )= 0.11391493676901D+00
 f( 99)=f( 20 02 00 )= 0.10632687359025D+00
 f(104)=f( 20 00 02 )=-0.97208244522503D-06
 f(105)=f( 13 00 00 )=-0.46920988100673D-01
 f(110)=f( 11 20 00 )= 0.12115811283676D+00
 f(111)=f( 11 11 00 )= 0.15686090127052D+00
 f(114)=f( 11 02 00 )= 0.12136644020664D+00
 f(119)=f( 11 00 02 )=-0.90532925139108D-05
 f(140)=f( 04 00 00 )=-0.15316136318697D-01
 f(145)=f( 02 20 00 )=-0.43576956173094D-01
 f(146)=f( 02 11 00 )=-0.30135101039669D-01
 f(149)=f( 02 02 00 )= 0.14351678141668D-01
 f(154)=f( 02 00 02 )= 0.17960371384799D-05
 f(175)=f( 00 40 00 )= 0.20162922300193D-01
 f(176)=f( 00 31 00 )= 0.66857170105718D-01
 f(179)=f( 00 22 00 )= 0.15491118105062D+00
 f(184)=f( 00 20 02 )=-0.19330543978900D-05
 f(185)=f( 00 13 00 )= 0.11528384959358D+00
 f(190)=f( 00 11 02 )=-0.43018721767318D-05
 f(195)=f( 00 04 00 )= 0.27546960679326D-01
 f(200)=f( 00 02 02 )=-0.10364516324444D-04
```

Comparison of maps for small diameter REC quadrupole singlet and standard MARYLIE quadrupole with hard edge fringe fields turned on.

```
matrix for map is :

 1.00000E+00 -1.56686E-06  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-5.66626E-06  9.99997E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  9.99999E-01  3.45027E-06  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.13207E-06  1.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 -4.64212E-15
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-0.81161528916714D-06
 f( 38)=f( 11 00 01 )=-0.54767003357242D-05
 f( 53)=f( 02 00 01 )= 0.13594565761027D-05
 f( 67)=f( 00 20 01 )=-0.13029011691045D-05
 f( 70)=f( 00 11 01 )=-0.13392474697907D-05
```

```
f( 76)=f( 00 02 01 )=-0.47180614553693D-05
f( 84)=f( 40 00 00 )=-0.10342718650296D-06
f( 85)=f( 31 00 00 )=-0.80460828215103D-05
f( 90)=f( 22 00 00 )= 0.15468738391533D-04
f( 95)=f( 20 20 00 )=-0.11639997890997D-03
f( 96)=f( 20 11 00 )=-0.11561285091838D-03
f( 99)=f( 20 02 00 )=-0.40475869682373D-04
f(104)=f( 20 00 02 )=-0.97208244522503D-06
f(105)=f( 13 00 00 )= 0.18909350831993D-04
f(110)=f( 11 20 00 )=-0.11946587430542D-03
f(111)=f( 11 11 00 )=-0.14100799571575D-03
f(114)=f( 11 02 00 )=-0.53944481173714D-04
f(119)=f( 11 00 02 )=-0.90532925139108D-05
f(140)=f( 04 00 00 )= 0.70335401435168D-05
f(145)=f( 02 20 00 )=-0.34181738981153D-04
f(146)=f( 02 11 00 )=-0.45094719394781D-04
f(149)=f( 02 02 00 )=-0.18308771305195D-04
f(154)=f( 02 00 02 )= 0.17960371384799D-05
f(175)=f( 00 40 00 )=-0.53603010102471D-04
f(176)=f( 00 31 00 )=-0.12223894393250D-03
f(179)=f( 00 22 00 )=-0.15378212460296D-03
f(184)=f( 00 20 02 )=-0.19330543978900D-05
f(185)=f( 00 13 00 )=-0.89409485270531D-04
f(190)=f( 00 11 02 )=-0.43018721767318D-05
f(195)=f( 00 04 00 )=-0.20896250376087D-04
f(200)=f( 00 02 02 )=-0.10364516324444D-04
```

```
#comment
 Exhibit 6.27.2.
 This is a MARYLIE input file illustrating the use of the type
 code recm to treat a REC quadrupole doublet.
 Each quadrupole is .5 meter long, and the spacing between
 quadrupoles is 1 meter.  The doublet is preceded and followed
 by half meter drifts.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 drecm    recm
  0.000000000000000E+00   3.00000000000000       1000.00000000000
   20.0000000000000       1.00000000000000       2.00000000000000
 mpoles   ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 npatps   ps2
  0.500000000000000       2.00000000000000       3.00000000000000
   2.00000000000000       1.00000000000000       3.00000000000000
 sect1    ps3
  0.500000000000000       3.00000000000000      0.100000000000000
  0.150000000000000       1.00000000000000       1.00000000000000
 sect2    ps4
  0.500000000000000      -3.00000000000000      0.100000000000000
  0.150000000000000      0.250000000000000       1.00000000000000
 end      end
#lines
 adrecm
     1*mpoles      1*npatps      1*sect1       1*sect2       1*drecm
 dtest
     1*adrecm      1*mapout
#lumps
#loops
#labor
    1*fileout
    1*dtest
    1*end

Integrating in   1000 steps from   0.00000=zi to   3.00000=zf
multipole strengths from pset 1:
 Sextupole:  0.0000E+00  0.0000E+00
 Octupole:   0.0000E+00  0.0000E+00
Pattern from pset 2:
  1 cycle(s) of  2 type(s) of section(s) with a maximum of  2 section(s)
```

```
 di=   0.50000
 types of section(s) are:
pset  length    strength   radii: inner      outer      tdrift    number
  3    0.500000    3.000000    0.100000    0.150000    1.000000     1
  4    0.500000   -3.000000    0.100000    0.150000    0.250000     1

matrix for map is :

 4.58640E-01  2.94577E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-1.24624E-01  1.37992E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.37992E+00  2.94577E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.24624E-01  4.58640E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.23343E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-0.79994216183894D-01
 f( 38)=f( 11 00 01 )= 0.59857331879881D+00
 f( 53)=f( 02 00 01 )=-0.18601761368241D+01
 f( 67)=f( 00 20 01 )=-0.79930560334125D-01
 f( 70)=f( 00 11 01 )=-0.58272401793467D+00
 f( 76)=f( 00 02 01 )=-0.18030981792421D+01
 f( 83)=f( 00 00 03 )=-0.73260547246492D+00
 f( 84)=f( 40 00 00 )=-0.73585950580279D-01
 f( 85)=f( 31 00 00 )= 0.46633279368660D+00
 f( 90)=f( 22 00 00 )=-0.11869949501820D+01
 f( 95)=f( 20 20 00 )=-0.14477042501046D+00
 f( 96)=f( 20 11 00 )= 0.43784264012853D+00
 f( 99)=f( 20 02 00 )=-0.84417905028554D+00
 f(104)=f( 20 00 02 )=-0.11596050528219D+00
 f(105)=f( 13 00 00 )= 0.13524335132499D+01
 f(110)=f( 11 20 00 )= 0.37222414340213D+00
 f(111)=f( 11 11 00 )=-0.12062073106956D+01
 f(114)=f( 11 02 00 )= 0.30829344895626D+01
 f(119)=f( 11 00 02 )= 0.87992026062122D+00
 f(140)=f( 04 00 00 )=-0.76593562539574D+00
 f(145)=f( 02 20 00 )=-0.34610542871067D+00
 f(146)=f( 02 11 00 )= 0.78377894590269D+00
 f(149)=f( 02 02 00 )=-0.31196451448320D+01
 f(154)=f( 02 00 02 )=-0.26446144220690D+01
 f(175)=f( 00 40 00 )=-0.14126020847969D-01
 f(176)=f( 00 31 00 )= 0.53187680246686D-01
 f(179)=f( 00 22 00 )=-0.15719171473213D+00
 f(184)=f( 00 20 02 )=-0.11576045266246D+00
 f(185)=f( 00 13 00 )= 0.21097924546462D-01
 f(190)=f( 00 11 02 )=-0.83011475568755D+00
 f(195)=f( 00 04 00 )=-0.10141281400139D+01
 f(200)=f( 00 02 02 )=-0.24652506856675D+01
 f(209)=f( 00 00 04 )=-0.93366303371893D+00
```

```
#comment
 Exhibit 6.27.3.
 This is a MARYLIE input file illustrating the use of the type
 code recm to treat a REC quadrupole triplet.  The leading and
 trailing quadrupoles are .25 meters long, and the center quadrupole
 is .5 meters long.  The spacing between quadrupoles is .5 meters.
 The triplet is preceded and followed by .5 meter drifts.
 The beam parameters are those for 800 MeV protons.
#beam
  4.881030646470486
 0.8526309382400936
  1.000000000000000
  1.000000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 trecm    recm
  0.000000000000000E+00   3.00000000000000       1000.00000000000
   20.0000000000000       1.00000000000000       2.00000000000000
 mpoles   ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 npatps   ps2
  0.500000000000000       3.00000000000000       3.00000000000000
   3.00000000000000       1.00000000000000       3.00000000000000
 sect1    ps3
  0.250000000000000       3.00000000000000      0.100000000000000
  0.150000000000000      0.500000000000000       1.00000000000000
 sect2    ps4
  0.500000000000000      -3.00000000000000      0.100000000000000
  0.150000000000000      0.500000000000000       1.00000000000000
 sect3    ps5
  0.250000000000000       3.00000000000000      0.100000000000000
  0.150000000000000      0.250000000000000       1.00000000000000
 end      end
#lines
 atrecm
     1*mpoles      1*npatps      1*sect1       1*sect2       1*sect3    &
     1*trecm
 ttest
     1*atrecm      1*mapout
#lumps
#loops
#labor
    1*fileout
    1*ttest
    1*end

Integrating in   1000 steps from   0.00000=zi to   3.00000=zf
multipole strengths from pset 1:
```

```
 Sextupole:  0.0000E+00  0.0000E+00
 Octupole:   0.0000E+00  0.0000E+00
Pattern from pset 2:
  1 cycle(s) of  3 type(s) of section(s) with a maximum of  3 section(s)
 di=   0.50000
 types of section(s) are:
pset  length    strength   radii: inner      outer      tdrift    number
  3    0.250000    3.000000    0.100000    0.150000    0.500000     1
  4    0.500000   -3.000000    0.100000    0.150000    0.500000     1
  5    0.250000    3.000000    0.100000    0.150000    0.250000     1

matrix for map is :

 9.56327E-01  3.19133E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-2.67721E-02  9.56327E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  9.52850E-01  2.72873E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.37433E-02  9.52850E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.23343E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-0.14558201321718D-01
 f( 38)=f( 11 00 01 )=-0.91396235620611D-02
 f( 53)=f( 02 00 01 )=-0.20618633840331D+01
 f( 67)=f( 00 20 01 )=-0.22955112353890D-01
 f( 70)=f( 00 11 01 )= 0.11809023728187D-01
 f( 76)=f( 00 02 01 )=-0.15228529986194D+01
 f( 83)=f( 00 00 03 )=-0.73260547246492D+00
 f( 84)=f( 40 00 00 )=-0.19290283831543D-01
 f( 85)=f( 31 00 00 )= 0.12397886307188D+00
 f( 90)=f( 22 00 00 )=-0.36977582024921D+00
 f( 95)=f( 20 20 00 )=-0.14594876580289D+00
 f( 96)=f( 20 11 00 )= 0.40366673398412D+00
 f( 99)=f( 20 02 00 )=-0.34148543473674D+00
 f(104)=f( 20 00 02 )=-0.18161374632310D-01
 f(105)=f( 13 00 00 )= 0.52683288181157D+00
 f(110)=f( 11 20 00 )= 0.47040181547617D+00
 f(111)=f( 11 11 00 )=-0.14583888579032D+01
 f(114)=f( 11 02 00 )= 0.13009303607433D+01
 f(119)=f( 11 00 02 )=-0.22662545686902D-01
 f(140)=f( 04 00 00 )=-0.73990461221895D+00
 f(145)=f( 02 20 00 )=-0.49557533408271D+00
 f(146)=f( 02 11 00 )= 0.16029433983071D+01
 f(149)=f( 02 02 00 )=-0.22250060801651D+01
 f(154)=f( 02 00 02 )=-0.29744260093858D+01
 f(175)=f( 00 40 00 )=-0.31853351723490D-01
 f(176)=f( 00 31 00 )= 0.17659934541379D+00
 f(179)=f( 00 22 00 )=-0.41008410541512D+00
 f(184)=f( 00 20 02 )=-0.34723075255933D-01
 f(185)=f( 00 13 00 )= 0.43673939150943D+00
 f(190)=f( 00 11 02 )= 0.31055692977809D-01
 f(195)=f( 00 04 00 )=-0.50310033952143D+00
```

```
f(200)=f( 00 02 02 )=-0.19310058689881D+01
f(209)=f( 00 00 04 )=-0.93366303371893D+00
```

6.28 Space

Type Code: spce

Required Parameters:

1. Length (m).

Example:

```
fakequad      spce
    .5
```

The above specifies a space for accounting purposes with user give name *fakequad*, and length .5 meters.

Description:

The type code *spce* may be used, for example, to refer to the path length of the design orbit in a quad or any other straight element whose transfer map has been produced by GENMAP (see section 1.4.1.) and has been read into a MaryLie run from an external file. This element type code has no direct effect in MaryLie runs that only compute and analyze maps, and perform ray traces. However, the contents of such elements are picked up by commands with the type codes *wcl* and *geom*. See sections 7.33 and 8.38. Thus, the lengths of spacer elements are available to the geometry command and auxiliary programs. See also section 6.31.

6.29 Change or Write Out Values of Parameters for Combined Function Dipole

Type Code: cfrn

Required Parameters:

1. JOB
   = 0 to change values of parameters.
   = 1 to write current values at the terminal.
   = 2 to write current values on file 12.
   = 3 to write current values at the terminal and on file 12.
2. Gap size (pole to pole distance) in meters.
3. Normalized field integral (dimensionless) for leading (entry) edge fringe field.
4. Normalized field integral (dimensionless) for trailing (exit) edge fringe field.

Example:

```
setfrn    cfrn
 0   .02   .5   .5
```

The above sets the gap of a combined function bend to .02 meters, and the leading and trailing normalized field integrals to .5 and .5, respectively.

Description:

The fringe-field parameters for the combined function bend have default values of 0, .5, .5 for the gap and leading and trailing normalized field integrals, respectively. See section 6.8. This command can be used to give them other values. It must be invoked in the #labor component before the element with type code *cfbd* is invoked.

6.30 Random Map

Type Code: rmap

Required Parameters:

1. IPSET (An integer from 1 to 9).

   This element also requires additional parameters whose values are specified by the parameter set IPSET. The contents of parameter set IPSET are used as follows:

   1. P1 = ISEED
   2. P2 = JOB
      = 0 for single use of random number generator.
      = 1 for multiple use of random number generator.
   3. P3 = IFLAG
      = 0 on first use.
   4. P4 = 0 (not used, but required by format specifications).
   5. P5 = 0 (not used, but required by format specifications).
   6. P6 = 0 (not used, but required by format specifications).
2. SF1
3. SF2C
4. SF2A
5. SF3
6. SF4

Example:

```
randmap    rmap
 1 , .5 , .6 , .7 , .8 , .9
```

The above specifies a map with user given name *randmap*. It makes use of parameter set 1. The contents of parameter set 1 might be as listed blow.

```
randps1    ps1
 137 , 0 , 0 , 0, 0 , 0
```

The command *randps1* must be invoked in #labor before *randmap* is invoked.

Description:

When a command with type code *rmap* is executed, a random map is generated and concatenated with the current transfer map. In addition, the following maps are put in the following buffers:

1. Buffer 1 contains a random f_1, f_3, and f_4 in its array part, and the random symplectic matrix associated with $e^{:f_2^c:}\ e^{:f_2^a:}$ in its matrix part.

2. Buffer 2 contains a random f_2^c in its array part, and the matrix associated with $e^{:f_2^c:}$ in its matrix part.

3. Buffer 3 contains a random f_2^a in its array part, and the matrix associated with $e^{:f_2^a:}$ in its matrix part.

4. Buffer 4 is left unchanged.

5. The previous contents of Buffer 5 are destroyed.

The type code *rmap* uses a random number generator in the generation of polynomials. Two different generators are available: ran1 and ran2. They are described in the Numerical Recipes book by Press et al. See section 11.12. When ISEED $\geq +1$, ran1 is used. When ISEED ≤ -1, ran2 is used. (Setting ISEED $= 0$ is not allowed, and produces an error message.) Both are better than most computer system-supplied random number generators. The generator ran1 is believed to be satisfactory for the production of up to 10^8 pseudo random numbers. The generator ran2 is believed to be among the best available for the production of as many as 10^{16} pseudo random numbers. However it is somewhat slower than ran1. If the user is concerned that his/her calculation may be sensitive to the choice of random number generator, the results of both should be compared for several different seeds.

The outputs of either ran1 or ran2 are shifted and scaled to produce random numbers in the interval (-1,1). These numbers are again scaled by the scale factors SF1, SF2C, SF2A, SF3, and SF4 to produce entries in $f_1 f_2^c, f_2^a, f_3$, and f_4, respectively.

If only one random map is to be generated, with the choice of the random numbers to be controlled by ISEED, then JOB = 0 should be used. If several maps are to be generated, say in some procedure (see section 9), then JOB = 1 should be used, and the parameter set IPSET should be evoked in #labor before entering the procedure. In this case, MaryLie will reset IFLAG to 1 after generating the first map. The value of IFLAG in the parameter set IPSET will not be changed wherever it appears in #menu. However, it is changed internally. (This can be verified, if desired, by subsequent use of a command with type code *wps*.)

Theory:

Any symplectic map through third order can be uniquely written in the form

$$\mathcal{M} = e^{:f_1:} e^{:f_2^c:} e^{:f_2^a:} e^{:f_3:} e^{:f_4:}.$$

The quadratic polynomial $e^{:f_2^c:}$ is of the form

$$f_2^c = -(1/2) \sum_{de} S^c_{de} z_d z_e$$

where S^c is a matrix that *commutes* with J. The quadratic polynomial f_2^a is of the form

$$f_2^a = -(1/2) \sum_{de} S^a_{de} z_d z_e$$

where S^a is a matrix that *anticommutes* with J. See section 8.22. It can be shown that maps of the form $\exp(: f_2^c :)$ have the topology of the 3-dimensional unitary group $U(3)$, which is *compact*. Therefore, use of a large value of SF2C simply results in "circumnavigating" to maps that could also have been reached with smaller values of SF2C. By contrast, maps of the form $\exp(: f_1 :)$, $\exp(: f_2^a :)$, $\exp(: f_3 :)$, and $\exp(: f_4 :)$ have Euclidean topology. Thus, use of larger values of SF1, SF2A, SF3, and SF4 results in maps that are farther from the identity map.

6.31 Arc

Type Code: arc

Required Parameters:

1. Dbefore (m).
2. Dafter (m).
3. Length (m).
4. Angle (degrees).

Example:

```
fakebend       arc
 .1   .2   .3   10
```

The above specifies an arc for accounting purposes with user given name *fakebend*. It has a "leading" drift of .1 m, a "trailing" drift of .2 m, an effective length of .3 meters, and a bend angle of 10 degrees.

Description:

The type code *arc* may be used, for example, to specify the net geometrical effect of a curved element whose transfer map has been produced by GENMAP (see section 1.4.1) and has been read into a MaryLie run from an external file. This element type code has no direct effect in MaryLie runs that only compute and analyze maps, and perform ray traces. However, the contents of such elements are picked up by commands with the type codes *wcl* and *geom*. See sections 7.33 and 8.38. Thus, the information specified by arc elements is available to the geometry command and auxiliary programs. See also section 6.28.

The meaning of the parameters is illustrated in figure 6.31.1. They are defined as follows:

Dbefore = distance from A to B.

Dafter = distance from B to C.

Length = distance from A to C along the design orbit.

Angle = angle required to rotate direction of AB into the direction of BC.

Figure 6.31.1: Geometrical significance of the parameters for the *arc* type code.

Chapter 7

Catalog of Simple Commands

MaryLie 3.0 is capable of executing a wide variety of commands. For simplicity of presentation, these commands are divided into the two categories of *simple* and *advanced*. Simple commands are described in this section, and advanced commands are described in section 8. The simple commands and their type code mnemonics, as currently available in MaryLie 3.0, are listed below. Also listed are the subsections that describe them in detail.

Type Code	Command	Subsection
end	Halt execution. Must occur somewhere in the #labor listing.	7.1
rt	Perform a ray trace.	7.2
circ	Set parameters and circulate.	7.3
pmif	Print contents of Master Input File (file 11).	7.4
tmi	Input matrix elements and polynomial coefficients from an external file.	7.5
tmo	Output matrix elements and polynomial coefficients to an external file.	7.6
ptm	Print transfer map.	7.7
iden	Replace existing transfer map by the identity map.	7.8
inv	Replace existing transfer map by its inverse.	7.9
tran	Replace existing transfer map by its "transpose".	7.10
rev	Replace existing transfer map by its reversed map.	7.11
revf	Replace existing transfer map by reverse factorized form.	7.12
mask	Mask off selected portions of existing transfer map.	7.13

Type Code	Command	Subsection
symp	Symplectify matrix portion of transfer map.	7.14
sqr	Square the existing transfer map.	7.15
stm	Store the existing transfer map.	7.16
gtm	Get a transfer map from storage.	7.17
rapt	Aperture the beam with a rectangular aperture.	7.18
eapt	Aperture the beam with an elliptic aperture.	7.19
wnd	Window a beam.	7.20
whst	Write history of beam loss.	7.21
ftm	Filter the existing transfer map.	7.22
cf	Close files.	7.23
of	Open files.	7.24
ps1 ⋮ ps9	Parameter set specification.	7.25
rps1 ⋮ rps9	Random parameter set specification.	7.26
num	Number lines in a file.	7.27
wps	Write out parameters in a parameter set.	7.28
time	Write out execution time.	7.29
cdf	Change drop file.	7.30
bell	Ring bell at terminal.	7.31
wmrt	Write out value of merit function.	7.32
wcl	Write contents of a loop.	7.33
paws	Pause.	7.34
inf	Change or write out values of infinities.	7.35
zer	Change or write out values of zeroes.	7.36
tpol	Twiss polynomial.	7.37
dpol	Dispersion polynomial.	7.38
cbm	Change or write out beam parameters.	7.39
dims	Dimensions.	7.40

Type Code	Command	Subsection
wuca	Write out contents of ucalc array.	7.41
wnda	Window a beam in all planes.	7.42
pli	Path length information.	7.43
shoa	Show contents of arrays.	7.44

Note that the type codes are given in lower case. If entries are made in upper case, they are automatically converted to lower case by PREP and MaryLie.

The purpose of this section is to outline the use of these simple commands by MaryLie, and to describe the parameters required to specify these commands in the #menu component of the Master Input File.

7.1 Halt Execution

Type Code: end

Required Parameters: None

Example:

```
stop    end
```

This specifies a command with the user given name *stop*. It terminates a MaryLie run. A command with the type code *end* must always occur somewhere in the #labor listing. See section 5.11.

7.2 Perform a Ray Trace

Type Code: rt

Required Parameters:

1. ICFILE (file number from which initial conditions are to be read)
   = 0 to use data currently stored in the initial conditions buffer.
   = NFILE $\geq$1 to read initial conditions from file number NFILE.
   = $-$J (with J an integer from 1 to 9) to read an initial condition from the parameter set array associated with the type code *psj*. See section 7.25.
2. NFCFILE (file number on which final conditions are to be written)
   = 0 to not write final conditions on a file.
   = MFILE $\geq$1 to write final conditions on file number MFILE in standard format (see section 4.5.2).
   = $-$MFILE (with MFILE $\geq$1) to write final conditions on file number MFILE in full precision (see section 4.5.2).
3. NORDER
   = -1 to simply read in initial conditions without performing a ray trace.
   = 0 to write contents of initial conditions buffer on file MFILE. (Equivalent to tracing rays using the identity map.)
   = 1 to trace rays using only linear matrix portion R of maps.
   = 2 to trace rays using only R and f_3 portions of maps through 2nd order.
   = 3 to trace rays using R, f_3, and f_4 portions of maps through 3rd order.
   = 4 to trace rays using R, f_3, and f_4 portions of maps through 4th order.
   $\geq$ 5 for a symplectic ray trace using a mixed variable generating function.
4. NTRACE
   = 0 to perform a single ray trace of all initial rays stored in the initial conditions buffer. Leaves initial conditions unchanged.
   $\geq$ 1 to perform NTRACE ray traces of all initial rays *with tracking*. When tracking occurs, the stored initial conditions are replaced each time by the results of the previous ray trace.
5. NWRITE
   Write the final conditions on file unit MFILE after every NWRITEth ray-trace.
6. IBRIEF (to suppress or redirect output):
   = 0 to be "very brief" (useful when tracking). In this mode, nothing concerning a ray trace will be printed at the terminal. The final conditions of a ray trace will be on file unit MFILE.
   = 1 to be "brief". The initial and final conditions of a ray trace will be printed at the terminal.

= 2 to be "not brief". Initial conditions, intermediate calculations, and final conditions will be printed at the terminal.

Note: If IBRIEF = −1 or −2, information is printed on an external file (file unit 12), not at the terminal. Apart from the choice of output file, the resulting output is the same as when IBRIEF = 1 and IBRIEF = 2, respectively.

Example:

```
rays    rt
 13, 14, 5, 100, 10, 0
```

This is a command with the user specified name *rays*. It calls for reading initial conditions from file 13 and the performance of a symplectic ray trace. Each initial ray in the initial conditions buffer will be tracked through the current total transfer map 100 times; the results will be written on file 14 every 10th trace. No other information concerning the tracing of rays is printed at the terminal or written on file 12.

NOTE WELL:

The initial conditions buffer cannot be empty if its contents are to be written out or if an actual ray trace is to occur. If either action is attempted with ICFILE = 0 and the initial conditions buffer *empty*, MaryLie will terminate. Before terminating, it writes a diagnostic message at the terminal and on file 12.

Description:

If ICFILE = NFILE ≥1, MaryLie reads from file NFILE and stores in memory (the initial conditions buffer called *zblock*) the initial conditions of all rays to be traced (up to a maximum of 10,000 rays). If ICFILE = 0, it is presumed that the initial conditions buffer is already nonempty as a result of some previous command. Initial conditions are conveniently read from file 13. However, other files may also be used if desired. See section 4.4.2.

When initial conditions are to be read from file NFILE, it is rewound prior to reading. Thus, the same file or several files can be used repeatedly to change the contents of the initial conditions buffer during a MaryLie run.

MaryLie traces all of the rays in the initial conditions buffer using the current transfer map. Initial conditions, intermediate calculations, and final conditions are printed if desired. Final conditions alone are always written on file unit MFILE for future use by plotting and other routines. The file unit MFILE is often conveniently taken to be file unit 14. However, other file units may also be used if desired. See section 4.5.2.

When NFCFILE = MFILE, final conditions are written on file MFILE in a standard 6 column format. See section 4.5.2. This format is useful for plotting and other post-processing routines.

However, use of this format gives only 6 digit precision. Upon occasion it may be desirable to have full precision. (For example, one may wish to write out the initial conditions buffer in full precision to be used in a subsequent MaryLie run to continue a tracking study.

(See sections 4.4.2, 4.5.2.) This may be accomplished by setting NFCFILE = −MFILE with MFILE ≥1. When this is done, final conditions are written on file unit MFILE in full precision.

When NORDER = −1, initial conditions are read in and stored in the initial conditions buffer, but no ray trace is performed. In this case, only the parameters ICFILE and NORDER are used. However, the other parameters must be present to satisfy format requirements. Moreover, one must have ICFILE ≥1. If this requirement is violated, the MaryLie run is terminated, and error messages are printed at the terminal (file 6) and on file 12.

When NORDER = 0, again no ray trace is actually performed. Instead, the contents of the initial conditions buffer are simply written on file unit MFILE. In this case only the parameters ICFILE, NFCFILE, and NORDER are used. However, the other parameters must again be present to satisfy format requirements.

When NORDER = 1, *only* the transfer matrix is used to perform a ray trace. In this case the result is correct only through first (linear) order. It is exactly symplectic because the transfer matrix is symplectic. (If the transfer matrix is not symplectic, it may be symplectified. See section 7.14.)

When NORDER = 2, the transfer matrix and $e^{:f_3:}$ are used to perform the ray trace with $e^{:f_3:}$ expanded and truncated as

$$e^{:f_3:} \simeq 1 + :f_3: \,. \tag{7.2.1}$$

In this case the result is correct through second order and symplectic through second order.

When NORDER = 3, the transfer matrix and $e^{:f_3:}e^{:f_4:}$ are used to perform the ray trace with $e^{:f_3:}e^{:f_4:}$ expanded and truncated as

$$e^{:f_3:}e^{:f_4:} \simeq 1 + :f_3: + \frac{:f_3:^2}{2} + :f_4: \,. \tag{7.2.2}$$

In this case, the result is correct through third order and symplectic through third order.

When NORDER = 4, the truncated expansion (3.11.6) is used. As explained in section 3.11.3, this procedure gives a result that is correct through third order and symplectic through fourth order.

Finally, when NORDER≥ 5, ray traces are performed in such a way that results are correct through third order, but are symplectic to all orders (to machine precision). This is achieved by two means. First, prior to the initiation of ray trace calculations, the matrix portion of the transfer map is "symplectified" using "symplectic polar decomposition" symplectification (see section 7.14) to remove any nonsymplectic matrix part that may have been produced by round off error in the process of building up the map by concatenation.

Second, each time a ray is traced, the *nonlinear* portion of the map is treated using a transformation function of mixed variables. As described in section 3.11.3, a transformation function $F(q, P)$ is computed. This function is of the form

$$F(q, P) = \sum_{j=1}^{3} q_j P_j + G(q, P) \tag{7.2.3}$$

where G is a polynomial in q, P computed from f_3 and f_4. The transformation function (7.2.3) leads to the implicit relations

$$Q_i = \partial F/\partial P_i = q_i + \partial G/\partial P_i = q_i + A_i(q, P), \tag{7.2.4}$$

$$p_i = \partial F/\partial q_i = P_i + \partial G/\partial q_i = P_i + B_i(q, P). \tag{7.2.5}$$

Here A_i and B_i (with $i = 1, 2, 3$) are composed of polynomials of degrees 2 and 3 in the components of q, P. The three implicit nonlinear relations (7.2.5) are rewritten in the form

$$P_i = p_i - B_i(q, P), \tag{7.2.6}$$

and solved numerically by Newton's method (using $P_i = p_i$ as an initial guess) to find the explicit results $P_i(q, p)$. These results are then substituted numerically into (7.2.4) to give $Q_i(q, p)$.

Sometimes the Newton procedure fails to converge properly. When this happens, the symplectic ray trace procedure goes into a *diagnostic* mode. Diagnostic messages appear at the terminal (file 6), and are also written on file 12. If the convergence problem becomes severe, that particular ray is marked as "lost", and is no longer tracked. Convergence failure generally indicates that nonlinear effects are very important, and that escape to distant regions of phase space is imminent.

When a ray trace occurs, the final spatial and temporal phase space coordinates are also checked to see if they exceed certain *infinities*. See section 7.35. Should this occur, that particular ray is also marked as lost. Lost rays are not tracked or traced. That is, their phase space coordinates are left unchanged. A history of what rays are lost, and when and why, is kept in MaryLie arrays whose contents can be examined by use of a command with type code *whst*. See section 7.21.

7.3 Circulate Around a Loop

Type Code: circ

Required Parameters:

1. ICFILE (file number from which initial conditions are to be read)
   = 0 to use data currently stored in the initial conditions buffer.
   = NFILE ≥ 1 to read initial conditions from file number NFILE.
   = $-$J (with J an integer from 1 to 9) to read an initial condition from the parameter set array associated with the type code *psj*. See section 7.25.
2. NFCFILE (file number on which final conditions are to be written)
   = MFILE ≥ 1 to write final conditions on file number MFILE in standard format (see section 4.5.2).
   = $-$MFILE (with MFILE $\geq$ 1) to write final conditions on file number MFILE in full precision (see section 4.5.2).
3. NORDER
   = 1 to trace rays using only linear matrix portion R of maps.
   = 2 to trace rays using only R and f_3 portions of maps.
   = 3 to trace rays using R, f_3, and f_4 portions of maps.
   = 4 to trace rays using R, f_3, and f_4 portions of maps through 4th order.
   ≥ 5 for symplectic rays traces.
4. NTIMES
   Specifies the number of times MaryLie is to circulate about a loop.
5. NWRITE
   Write the final conditions on file MFILE after every NWRITEth circulation.
6. ISEND
   = 0 to suppress writing of items in loop.
   = 1 to write items in loop at the terminal.
   = 2 to write items in loop on file 12.
   = 3 to write items in loop at the terminal and on file 12.

Example:

```
track     circ
13 , 14 , 5 , 50000 , 10 , 0
```

This is a command with the user given name *track*. It calls for circulation around the (last) loop proceeding it in the #labor component of the Master Input File. Initial conditions are to be read from file 13, and ray traces are to be carried out using the symplectic ray trace

procedure. Tracking will occur for 50,000 circulations, and results will be written on file 14 every 10th circulation. The items in the loop will not be written out.

NOTE WELL:

The initial conditions buffer cannot be empty if a circulation is to occur. If a circulation is attempted with ICFILE = 0 and the initial conditions buffer empty, MaryLie will terminate. Before terminating, it will write a diagnostic message at the terminal and on file 12.

Description:

In some cases, rather than use the net concatenated map for a collection of elements, it is more accurate (but slower) to track rays element by element, or perhaps through a small collection of elements and/or lumps. That is one of the purposes of loops. See section 5.10. To use a loop for tracking, the name and contents of the loop must be defined in the #loops component of the Master Input File. The name of the loop is invoked in the #labor component of the Master Input File followed (not necessarily immediately) by a command having the type code *circ*. Loops along with a command having the type code *circ* are useful for long-term tracking studies of circular machines. See sections 2.7 and 10.8. They can also be used to carry out single-pass element-by-element tracking through a beam line. See section 10.3. In this case, a command with the type code *circ* is used with NTIMES = 1.

Whenever an *element* is encountered in a loop in the course of executing a command having type code *circ* with NORDER ≥ 5, the required map $\mathcal{M}$, transformation function F, and polynomials A_i, B_i are computed and used to perform a symplectic ray trace. See sections 3.11.3 and 7.2. At the end of the ray trace through *this element*, the map, transformation function, and polynomials are all discarded. Thus, if they are needed again, they must all be recomputed.

However, in the case of a *lump*, the map, transformation function, and polynomials are computed (if not already computed previously) and stored for possible future use. Thus, if many passes are to be made through a loop (as in long-term tracking studies for which NTIMES is large), it is desirable to have as many maps in the loop as possible stored in the form of lumps. See section 5.9.

7.4 Print the Contents of the Master Input File

Type Code: pmif

Required Parameters:

1. ITYPE
   = 0 to print the contents just as they are without suppression of interspersed comments and "commented out" entries. See section 5.5.4.
   = 1 to print the contents as interpreted by MaryLie.
   = 2 to print only the #labor component as interpreted by MaryLie.
2. IFILE (external output file).
3. ISEND
   = 1 to write at the terminal.
   = 2 to write on file IFILE.
   = 3 to write at the terminal and on file IFILE.

Example:

```
fileout    pmif
   1 , 12 , 3
```

This is a command with the user given name *fileout*. It calls for the printing of the Master Input File (file 11), as interpreted by MaryLie, at the terminal and on file 12.

NOTE WELL:

Even if not used, an external file number must be present for format reasons. The external file is conveniently taken to be file 12. See section 4.2. However, other files may also be used if desired.

Comment:

A command having the type code *pmif* can be used at the end of a MaryLie run involving fitting or optimization to produce a revised Master Input File for subsequent use. See sections 9.7 and 9.8.

7.5 Input of a Transfer Map

Type Code: tmi

Required Parameters:

1. IOPT
   = 1 to concatenate existing map with the input map.
   = 2 to replace existing map with the input map.
2. IFILE (file number from which map is to be read).
3. NOPT (Rewind Option)
   = 0 to read (from file IFILE) in its current position.
   = 1 to first rewind file unit IFILE, then read.
4. NSKIP
   Number of data sets (maps) to be skipped in file IFILE (starting from the current position) before a read actually occurs.

NOTE: If NOPT = 1 and NSKIP = −1, file IFILE is simply rewound, but not read. Thus, when employed with these parameter values, the type code *tmi* can be used simply to rewind files.

Example 1:

```
mapin1    tmi
1, 15 , 0 , 0
```

The above describes a command with the user specified name *mapin1*. It would cause MaryLie to read into program memory (and then use) the nonzero matrix elements and polynomial coefficients of a map written on the current position of file unit 15.

Example 2:

```
mapin2    tmi
1, 15 , 0 , 2
```

The above describes a command with the user specified name *mapin2*. It would cause MaryLie to skip over the next 2 transfer maps on file 15, and then read in (and use) the subsequent transfer map.

Example 3:

```
mapin3    tmi
1, 15 , 1 , 2
```

The above describes a command with the user specified name *mapin3*. It would cause MaryLie to first rewind file 15, and then read in (and use) the 3rd transfer map stored on file 15.

Example 4:

```
rewind15    tmi
1, 15 , 1 , -1
```

The above describes a command with the user specified name *rewind15*. It would cause MaryLie to simply rewind file 15, and would not read in any map. Indeed, file 15 need not even contain a map.

Description:

The type code *tmi* is used primarily to read the nonzero matrix elements R_{ij} and nonzero coefficients of the polynomials f_3 and f_4 for a map and path length $f(0)$ from a file. It can also be used to read in a complete set of polynomials f_1, f_2, f_3, f_4. File 15 is conveniently used for this purpose. However, other files may also be used if desired. See section 4.4.3 for the format of a transfer map input file.

This type code could be used to read in a transfer map produced by the current or some other MaryLie run. See, for example, the use of a normal form map in section 2.6, and section 7.6. It could also be used to read in transfer maps calculated by hand or obtained from another source such as another MaryLie run or some external GENMAP routine. (See section 1.4.1). In this way, MaryLie can be used to manipulate and/or trace rays through *any* system (say, an ordinary light optical system) for which R, f_3, and f_4 are explicitly known.

If NSKIP = 0, MaryLie will read whatever transfer map occurs next on file IFILE. In the case that file IFILE contains several transfer maps, it is possible to read any one of them by setting NOPT = 1 and then setting NSKIP to skip over the appropriate number of maps before actually reading. Finally, as illustrated in Example 4 above, the type code *tmi* may be used simply to rewind a file (including files containing information other than transfer maps).

7.6 Output of a Transfer Map

Type Code: tmo

Required Parameters:

1. IFILE (file number on which map is to be written).

Example:

```
ritemap    tmo
 16
```

The above specifies a command with the user given name *ritemap*. It would cause MaryLie to write the current transfer map on file 16.

Description:

This type code is used to write on a file the nonzero matrix elements and the nonzero coefficients of the polynomials f_3 and f_4 (and also f_1 and f_2, if present) plus path length $f(0)$ for the current transfer map. File 16 is conveniently used for this purpose. However, other files may also be used for this purpose if desired. See section 4.5.3 for the format of a transfer map output file.

This type code could be used at the end of or during a MaryLie run, for example, to store one or several transfer maps on a file for future use. It can also be used to store maps on a file for subsequent use during the current MaryLie run.

Whenever a map is to be written on a file as a result of a command with type code *tmo*, the file is first brought to its end, and then the map is written. Therefore, if *tmo* is used on a file several times, several maps will be written, one after the other, without overwriting. This is true even if the file has been repositioned (by the use of *tmi*) once or several times during the course of a MaryLie run.

Note that the file handling procedure associated with *tmo* also implies that if a file written by *tmo* is to be read within the same MaryLie run using *tmi*, it must first be rewound using the option NOPT = 1. See section 7.5.

Since MaryLie, when executing a command with type code *tmo*, first reads the file IFILE in order to bring it to its end, the file must either be pre-existing (but perhaps empty), or must be opened using a command having the type code *of*. See section 7.24.

7.7 Print Transfer Map

Type Code: ptm

Required Parameters:

1. NM (controls output of the transfer matrix R)
   = 0 to not print R.
   = 1 to write R at the terminal.
   = 2 to write R on file 12.
   = 3 to write R at the terminal and on file 12.
2. NF (controls output of the polynomials f_1 through f_4)
   = 0 to not print f_1 through f_4.
   = 1 to write at the terminal the nonzero coefficients in the polynomials f_1 and f_2 (if any), and f_3 and f_4.
   = 2 to write nonzero coefficients of f_1 and f_2 (if any), and f_3 and f_4 on file 12.
   = 3 to write coefficients at the terminal and on file 12.
3. NT (controls output of the second order transfer matrix T)
   = 0 to not print T.
   = 1 to write T at the terminal.
   = 2 to write T on file 12.
   = 3 to write T at the terminal and on file 12.
4. NU (controls output of the third order transfer matrix U)
   = 0 to not print U.
   = 1 to write U at the terminal.
   = 2 to write U on file 12.
   = 3 to write U at the terminal and on file 12.
5. IBASIS
   = 1 for cartesian basis.
   = 2 for static resonance basis.
   = 3 for dynamic resonance basis.

Example:

```
mapout    ptm
  1, 1, 0, 0 , 1
```

The above specifies a command with the user given name *mapout*. It calls for a printing at the terminal of the matrix and f_3 and f_4 for the current total transfer map. The map is presumed to be in the cartesian basis.

NOTE WELL:

The use of NT$\neq$ 0 and NU$\neq$ 0 is meaningful only if the map is in the cartesian basis. In this case one should have IBASIS = 1. If the map is in a static resonance or dynamic resonance basis, only the matrix R and the polynomials f are meaningful. In this case, the f's are given in one of these bases, and one must set IBASIS = 2 or 3, respectively. The matrix R is always given in terms of the cartesian basis.

Description:

This type code is used to print information about the current total transfer map. For a description of the labeling of f_3 and f_4, see section 3.10. For a description of R, T, and U, see section 3.6. The arrays for z, R, T, and U are formatted in the following fashion:

<u>Coordinate Array</u>

$$\begin{array}{rcccccc} z = & (X, & P_x, & Y, & P_y, & \tau, & P_\tau) \\ & 1 & 2 & 3 & 4 & 5 & 6 \end{array}$$

<u>R Matrix Array</u>

$$R = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{16} \\ R_{21} & R_{22} & & R_{26} \\ \vdots & & & \vdots \\ R_{61} & R_{62} & \cdots & R_{66} \end{pmatrix}$$

<u>T Matrix Array</u>

According to section 3.6, the contributions to the quantities z_a^{fin} arising from the T array are expressions of the form

$$\sum_{b,c} T_{abc} z_b^{\text{in}} z_c^{\text{in}} . \tag{7.7.1}$$

That is, for each value of the index a, the quantities (7.7.1) are second order polynomials. These second order polynomials may be decomposed into second order monomials, and these monomials may be labeled using the scheme of section 3.10. Thus, for example,

t1(7) = t1(20 00 00)
= coefficient of $(X^{\text{in}})^2$ in the expression for X^{fin},

t4(8) = t4(11 00 00)
= coefficient of $(X^{\text{in}} P_x^{\text{in}})$ in the expression for P_y^{fin}, etc.

U Matrix Array

According to section 3.6, the contribution to the quantities z_a^{fin} arising from the U array are expressions of the form

$$\sum_{b,c,d} U_{abcd} z_b^{\text{in}} z_c^{\text{in}} z_d^{\text{in}} . \tag{7.7.2}$$

That is, for each value of the index a, the quantities (7.7.2) are third order polynomials. These third order polynomials may be decomposed into third order monomials, and these monomials may be labeled using the scheme of section 3.10. Thus, for example,

$$\begin{aligned} \text{u1}(29) &= \text{u1}(21\ 00\ 00) \\ &= \text{coefficient of } (X^{\text{in}})^2 (P_x^{\text{in}}) \text{ in the expression for } X^{\text{fin}}, \end{aligned}$$

$$\begin{aligned} \text{u4}(82) &= \text{u4}(00\ 00\ 12) \\ &= \text{coefficient of } (\tau^{\text{in}})(P_\tau^{\text{in}})^2 \text{ in the expression for } P_y^{\text{fin}}, \text{ etc.} \end{aligned}$$

Cartesian and Resonance Bases

For a description of cartesian, static resonance, and dynamic resonance bases, see sections 14.1 through 14.3. See also section 8.6.

7.8 Replace with Identity Map

Type Code: iden

Required Parameters: None

Example:

```
resetmap    iden
```

This specifies a command with the user given name *resetmap*. It causes the current transfer map to be replaced by the identity map. It also sets the path length to zero.

Description:

From time to time in a MaryLie run, it may be necessary to reset the current transfer map to the identity map. This can be done with a command having the type code *iden*.

7.9 Invert a Map

Type Code: inv

Required Parameters: None

Example:

```
invert    inv
```

This specifies a command with the user given name *invert*. It causes the current transfer map to be replaced by its inverse. It also changes the sign of the path length.

Description:

From time to time in a MaryLie run, it may be necessary to invert the current transfer map. This can be done with a command having the type code *inv*. The calculation of the inverse of R, the matrix portion of the map, is made on the assumption that R is symplectic.

To check whether two maps are the same, it is convenient to invert one and then multiply by the other. The two maps are the same if the result of this operation is the identity map.

7.10 "Transpose" a Map

Type Code: tran

Required Parameters: None

Example:

```
trnspose    tran
```

This is a command with the user specified name *trnspose*. When invoked, it causes the matrix part of the current transfer map to be replaced by its transpose. The polynomials f_3 and f_4 are unaffected.

Description:

It is useful to be able to transpose a matrix in order to check and exploit the symplectic condition. See sections 3.5 and 6.18.

7.11 Reverse a Map

Type Code: rev

Required Parameters: None

Example:

```
reverse    rev
```

This specifies a command with the user given name *reverse*. It causes the current transfer map $\mathcal{M}$ to be replaced by the reversed map $\mathcal{M}^r$.

Description:

Suppose $\mathcal{M}$ is the transfer map relating the initial conditions at the entrance of some element to the final conditions at the exit of the element. That is, $\mathcal{M}$ describes *direct* passage from the entrance to the exit. Then, roughly speaking, the reversed map $\mathcal{M}^r$ describes the hypothetical result of *reverse* passage from the exit to the entrance.

For many elements the direct and reverse maps are the same. Such elements are said to be *reversible*. Now suppose $\mathcal{M}$ is the map for a collection of elements $A, B, C \cdots$, and that each of the elements separately is reversible. Then $\mathcal{M}^r$ is the map for the reverse order collection $\cdots C, B, A$.

7.12 Reverse Factorize a Map

Type Code: revf

Required Parameters:

1. IORD
   = 0 to go from the normal MaryLie order to the reversed order.
   = 1 to go from the reversed order to the normal MaryLie order.

Example:

```
revfact    revf
   0
```

This specifies a command with the user given name *revfact*. Because IORD = 0, it assumes that the current transfer map has the normal MaryLie order representation. It replaces the polynomials f_3 and f_4 of the normal MaryLie order representation of the map by the polynomials g_3 and g_4 of the reversed factorized representation. The matrix part R of the map is left unchanged.

Description:

As described in section 3.9, an arbitrary symplectic map $\mathcal{M}$ can be written in the form (3.9.1). It can also be written in reversed factorized form. That is, the map $\mathcal{M}$ can be written in the forms

$$\mathcal{M} = e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdots \text{ (normal MaryLie order)},$$

$$\mathcal{M} = \cdots e^{:g_4:} e^{:g_3:} e^{:g_2:} \text{ (reversed order)} .$$

The polynomials f_2 and g_2 are identical and both lead to the same matrix R as given by (3.9.3). However, f_3 and g_3, and f_4 and g_4, generally differ.

If IORD = 0, it is presumed that the polynomials associated with the current transfer map are f_3 and f_4. In that case, g_3 and g_4 are computed, and f_3 and f_4 are then replaced by g_3 and g_4, respectively.

If IORD = 1, it is presumed that the polynomials associated with the current transfer map are g_3 and g_4. In that case, f_3 and f_4 are computed, and g_3 and g_4 are then replaced by f_3 and f_4, respectively.

7.13 Mask a Map

Type Code: mask

Required Parameters:

1. IMAT
   = 0 to replace the matrix portion R of the map by the identity matrix.
   = 1 to leave the matrix portion R of the map unaffected.
2. IF0
   = 0 to replace the f_0 polynomial of the map by the zero polynomial.
   = 1 to leave the f_0 polynomial of the map unaffected.
3. IF1
   = 0 to replace the f_1 polynomial of the map by the zero polynomial.
   = 1 to leave the f_1 polynomial of the map unaffected.
4. IF2
   = 0 to replace the f_2 polynomial of the map by the zero polynomial.
   = 1 to leave the f_2 polynomial of the map unaffected.
5. IF3
   = 0 to replace the f_3 polynomial of the map by the zero polynomial.
   = 1 to leave the f_3 polynomial of the map unaffected.
6. IF4
   = 0 to replace the f_4 polynomial of the map by the zero polynomial.
   = 1 to leave the f_4 polynomial of the map unaffected.

Example:

```
linear    mask
 1 , 0 , 0 , 0 , 0 , 0
```

This specifies a command with the user given name *linear*. It removes all the polynomial parts of the current transfer map, and leaves the linear part R unchanged.

Description:

Upon occasion in a MaryLie run, it is useful to be able to examine the effect of various orders of nonlinearity separately. For example, the above command *linear* could be used for a tracking study that employed only the linear part of the transfer map. For a more refined way of removing terms from the current transfer map, see section 7.22.

7.14 Symplectify a Matrix

Type Code: symp

Required Parameters:

1. IOPT
   = 1 for a static map.
   = 2 for a dynamic (time dependent) map.
2. KIND
   = 1 for symplectic polar decomposition symplectification.
   = 2 for "modified Darboux" symplectification.

Example:

```
makesymp    symp
  1  2
```

This specifies a command with the user given name *makesymp*. It replaces the matrix part R of the current transfer map by a matrix that is symplectic to machine precision. Since IOPT = 1, the map is assumed to be static. Since KIND = 2, "modified Darboux" symplectification is used.

Description:

Due to the roundoff error incurred in concatenating many maps or to the truncation error incurred in the use of numerical integration routines (e.g. GENMAP, see section 1.4.1), the matrix part of the current transfer map may become slightly nonsymplectic. Commands having the type code *symp* can be used to replace such matrices with nearby matrices that are exactly symplectic (to machine precision). For further detail, see references listed in Chapter 11.

7.15 Square a Map

Type Code: sqr

Required Parameters: None

Example:

```
pow2    sqr
```

This specifies a command with the user given name *pow2*. It squares the existing transfer map $\mathcal{M}$. That is, $\mathcal{M}$ is replaced by $\mathcal{M}^2$.

Description:

Repeated invocations of the square command produce successively higher powers. Thus, if the current transfer map were $\mathcal{M}$, two invocations of the square command would replace $\mathcal{M}$ by $\mathcal{M}^4$ $\{= (\mathcal{M}^2)^2\}$, etc. This feature makes it possible to produce very high powers of $\mathcal{M}$ with relatively few steps.

7.16 Store Transfer Map

Type Code: stm

Required Parameters:

1. NMAP (an integer from 1 through 9).

Example:

```
stormap3   stm
   3
```

This specifies a command with the user given name *stormap3*. It stores the current transfer map in location 3. In so doing, it writes over (destroys) any other map that might happen to be in location 3.

Description:

There are three ways to store a transfer map for possible later use within a MaryLie run. First, it can be written to a file using a comand with type code *tmo*, and read in later using a command wth type code *tmi*. See sections 7.5 and 7.6. Second, the map can be made into a lump. See section 5.9. In this case, not only is the map stored, but also all associated functions required for a symplectic ray trace are stored. See section 7.3. Finally, the map can be stored (in any one of nine storage locations set aside for this purpose) by using a command having type code *stm*. In that case, the map can be retrieved by use of a command with type code *gtm*. See section 7.17.

7.17 Get Transfer Map

Type Code: gtm

Required Parameters:

1. IOPT
   = 1 to concatenate existing map with map stored in location NMAP.
   = 2 to replace the existing map with map stored in location NMAP.
2. NMAP (an integer from 1 through 9).

Example:

```
getmap2   gtm
1, 2
```

This specifies a command having the user given name *getmap2*. It gets a map from storage location 2 (a map presumably placed there by an earlier use of a command having type code *stm*) and concatenates it with the existing map.

Description:

Nine internal storage locations are set aside in MaryLie for the storage and retrieval of transfer maps. Maps are stored in these locations using commands having type code *stm*. See section 7.16.

7.18 Aperture Beam with Rectangular Aperture

Type Code: rapt

Required Parameters:

1. JOB
   = 0 to simply remove particles outside the aperture.
   = 1 to also write out the phase-space coordinates of the removed particles on the file IFILE.
2. IFILE
   = 0 to not write out phase-space coordinates of removed particles.
   = MFILE ≥ 1 to write phase-space coordinates of removed particles on file number MFILE in standard format (see section 4.5.2).
   = $-$MFILE (with MFILE ≥ 1) to write phase-space coordinates of removed particles on file number MFILE in full precision (see section 4.5.2).
3. XMIN
4. XMAX
5. YMIN
6. YMAX

Example:

```
  scrape     rapt
1  16  -.01  .01  -.005  .005
```

This specifies a command with the user given name *scrape*. It "removes" particles whose transverse coordinates X,Y lie outside the rectangle (XMIN, XMAX), (YMIN, YMAX). It also writes on file 16 (in standard format) the phase-space coordinates of the removed particles.

Description:

This command examines the particles in the initial conditions buffer and marks as *lost* all those particles whose (dimensionless) transverse coordinates X,Y lie *outside* the rectangle

$$X \in \ (\mathrm{XMIN}, \mathrm{XMAX}),$$

$$Y \in \ (\mathrm{YMIN}, \mathrm{YMAX}).$$

Such lost particles are then no longer traced in subsequent ray traces. See sections 7.2 and 7.3. If JOB=1, the phase-space coordinates of the removed particles are also written on file IFILE. This feature makes it possible to model a *septum* using the type code *rapt* since the phase-space coordinates of the removed particles can be used as initial conditions for ray tracing these particles through some extraction beam line later in a MaryLie run or in some subsequent MaryLie run.

7.19 Aperture Beam with Elliptic Aperture

Type Code: eapt

Required Parameters:

1. SMAS (semi-major axis squared)
2. RAXS (ratio of axes squared)

Example:

```
pipe    eapt
  4.e-4 , 1.
```

This specifies a command with the user given name *pipe*. It "removes" particles whose transverse coordinates X, Y lie outside an ellipse whose semi-major axis squared is SMAS and ratio of axes squared is RAXS.

Description:

This command examines the particles in the initial conditions buffer and marks as *lost* all those particles whose (dimensionless) transverse coordinates X, Y lie *outside* the ellipse

$$X^2 + RAXS * Y^2 < SMAS.$$

Such lost particles are then no longer traced in subsequent ray traces. See sections 7.2 and 7.3.

7.20 Window a Beam

Type Code: wnd

Required Parameters:

1. IPLANE
   = 1 for X, P_x plane.
   = 2 for Y, P_y plane.
   = 3 for τ, P_τ plane.
2. QMIN
3. QMAX
4. PMIN
5. PMAX

Example:

```
xwind    wnd
1, -.01, .01, -.02, .02
```

This specifies a command with the user given name *xwind*. It "removes" particles whose X, P_x phase-space coordinates lie outside the rectangular region (QMIN, QMAX), (PMIN, PMAX).

Description:

This command examines the particles in the initial conditions buffer to see if they lie outside various rectangular regions in phase space. Suppose, for example, that IPLANE = 1. Then this command marks as *lost* all those particles whose phase-space coordinates X, P_x lie *outside* the rectangular region

$$\mathrm{X} \in (\mathrm{QMIN}, \mathrm{QMAX}),$$
$$\mathrm{P}_x \in (\mathrm{PMIN}, \mathrm{PMAX}).$$

Such lost particles are then no longer traced in subsequent ray traces. See sections 7.2 and 7.3. See also section 7.42.

7.21 Write History

Type Code: whst

Required Parameters:

1. IFILE (number of file on which history is to be written).
2. JOB
   = 1 to write out contents of the array *istat*.
   = 2 to write out the contents of the array *ihist*.

Example:

```
lost    whst
 22       1
```

This is a command with the user given name *lost*. It writes out the contents of the array *istat* on file 22.

Description:

MaryLie uses three arrays to treat phase-space data. The first, *zblock(i,j)*, stores the *i*'th coordinate of the *j*'th particle. It is the "initial conditions buffer". Here i ranges from 1 to 6, and j ranges from 1 to *maxray*.

The second, *istat(j)*, describes the "status" of the *j*'th particle. When phase-space data is read into *zblock*, the corresponding entry in *istat* for each particle is set to zero. Then, when a particle is "lost" either due to some aperturing command or failure of a symplectic ray trace to converge, the corresponding entry in *istat* is set to the "turn number" for which the particle is lost. (Successive passages through a given element are counted to produce a "turn number".) Thus, at the end of a ray tracing or aperturing operation, *istat* describes the status of each particle with $istat(j) = 0$ if the *j*'th particle was not lost, and $istat(j) = iturn$ if this particle was lost on turn *iturn*.

The third, $ihst(m, n)$, records the "history" of lost particles. Here m ranges from 1 to *nlost* where *nlost* is the total number of particles lost, and n ranges from 1 to 2. Suppose the m'th particle to be lost is lost on turn *iturn*, and that this particle is the *j*'th particle in the initial distribution. Then $ihst(m, 1) = iturn$ and $ihst(m, 2) = j$.

See section 10.8 for an example of the use of *whst* to display the contents of both the arrays *istat* and *ihst*.

7.22 Filter Transfer Map

Type Code: ftm

Required Parameters:

1. IFILE [file number from which filter (in the form of a map) is to be read].
2. NOPT (Rewind Option)
   = 0 to read (from file IFILE) in its current position.
   = 1 to first rewind file unit IFILE, then read.
3. NSKIP
   Number of records (maps) to be skipped in file IFILE (starting from the current position) before a read actually occurs.
4. KIND
   = 0 for normal filter.
   = 1 for reversed filter.

Example 1:

```
keepterm    ftm
 19 , 0 , 0 , 0
```

This is a command with the user given name *keepterm*. It keeps in the current transfer map all the terms that are indicated in file IFILE. The rest are set to zero.

Example 2:

```
killterm    ftm
 19 , 0 , 0 , 1
```

This is a command with the user given name *killterm*. It removes from the current transfer map all the terms that are indicated in file IFILE. That is, they are set to zero. The rest are left unchanged.

Description:

During the course of a MaryLie run it may be useful to remove various terms from the current transfer map. This can be done either by keeping specified terms and setting the rest to zero, or by removing (setting to zero) specified terms and leaving the rest unchanged. The file IFILE specifies what terms are to be kept or removed. It is written in map format. See section 4.4.3. Items to kept (or removed) are indicated by setting the corresponding polynomial entries in IFILE to 1. For example, suppose IFILE contains the lines

```
6, 6, 0
28, 1
209, 0
```

Then, if KIND = 0, the term in the current transfer map with MaryLie index 28 will be kept, and all others will be set to zero. And, if KIND = 1, the term in the current transfer map with MaryLie index 28 will be set to zero. Note that in this application all matrix entries in IFILE are ignored. However, the 6,6 entry must be present to satisfy format requirements.

7.23 Close Files

Type Code: cf

Required Parameters:

1. IFILE1
2. IFILE2
3. IFILE3
4. IFILE4
5. IFILE5
6. IFILE6

Note: All six parameters must be present to satisfy format requirements. However, some of them may be zero. Parameters having the value zero are ignored.

Example:

```
endfiles     cf
 14 , 16 , 0 , 0 , 0 , 0
```

This is a command with the user given name *endfiles*. It closes files 14 and 16.

Description:

For some operating systems and some circumstances it is useful to close files while in a MaryLie run. This command invokes, among others, the following FORTRAN statements for files IFILE1 through IFILE6:

```
      subroutine cf(p)
c  subroutine to close files
c
         .
         .
c set up control indices
      do 10 j=1,6
      ip(j)=nint(p(j))
   10 continue
c
c close indicated files
      do 20 j=1,6
      n=ip(j)
      if( n.gt.0) close(unit=n, err=30)
      go to 20
   30 write(jof,*) 'error in closing file unit ',n
   20 continue
```

7.24 Open Files

Type Code: of

Required Parameters:

1. IFILE1
2. IFILE2
3. IFILE3
4. IFILE4
5. IFILE5
6. IFILE6

Note: All six parameters must be present to satisfy format requirements. However, some of them may be zero. Parameters having the value zero are ignored.

Example:

```
openfile    of
 14 , 16 , 0 , 0 , 0 , 0
```

This is a command with the user given name *openfile*. It opens files 14 and 16.

Description:

For some operating systems and some circumstances it is useful to be able to open files while in a MaryLie run. This command makes it possible to open files in the range 1 through 50. It invokes, among others, the following FORTRAN statements for files IFILE1 through IFILE6:

```
      subroutine of(p)
c  subroutine to open files
c
      character*6 unit
      dimension unit(50)
           .
           .
c set up file names
      data (unit(i),i=1,50)/
     #'unit01','unit02','unit03','unit04','unit05',
           .
           .
     #'unit46','unit47','unit48','unit49','unit50'/
c
c set up control indices
      do 10 j=1,6
      ip(j)=nint(p(j))
   10 continue
```

```
c
c open indicated files
      do 20 j=1,6
      n=ip(j)
     if( n.gt.0 .and. n.le.50 .and. n.ne.lf .and. n.ne.jof
    # .and. n.ne.jodf) then
      open(unit=n, file=unit(n), status='unknown', err=30)
      endif
      go to 20
   30 write(jof,*) 'error in opening file unit ',n
   20 continue
```

7.25 Parameter Set Specification

Type Code: ps1, ps2, ps3, ps4, ps5, ps6, ps7, ps8, ps9

Required Parameters:

1. P1
2. P2
3. P3
4. P4
5. P5
6. P6

Example:

```
data    ps1
 .1 , .3 , 4 , 0 , 0 , 0
```

This is a command with the user given name *data*. It gives the parameters P1 through P6 in parameter set 1 the values .1, .3, 4, 0, 0, 0, respectively

Description:

Some elements require more than six parameters, and take these additional parameter values from parameter sets. Parameter sets can also be used to specify initial conditions for ray traces. See sections 7.2 and 7.3. This feature is useful because the *vary* command (see section 9.6) can vary the parameter values in commands having the type codes *ps1* through *ps9* in connection with fitting, optimization, and scanning operations. Also, random elements can take their values from parameter sets. See section 6.19. Finally, constraint commands having type codes *con1* through *con5* have access to the contents of parameter sets.

Note:

Parameter set commands must be invoked explicitly or implicitly in #labor to have an effect. See, for example, section 6.8.

7.26 Random Parameter Set Specification

Type Code: rps1, rps2, rps3, rps4, rps5, rps6, rps7, rps8, rps9

Required Parameters:

1. ISOURCE
   = −J (with J an integer from 1 to 9) to take parameters from a parameter set associated with the type code *psj*.
   = NFILE (with NFILE $\geq$ 1) to read parameters from the file unit NFILE.
2. IECHO
   = 0 not to echo back parameter set values.
   = 1 to write parameter set values at the terminal.
   = 2 to write parameter set values on file 12.
   = 3 to write parameter set values at the terminal and on file 12.

Example:

```
rdata    rps1
  17      0
```

This is a command with the user given name *rdata*. It reads in from file 17 six parameter values for parameter set 1. They are not echoed back.

Description:

Upon occasion it is useful to set parameter values by reading an external file, or by referencing some other parameter set. This feature is analogous to that for random elements. See section 6.19.

7.27 Number Lines in a File

Type Code: num

Required Parameters:

1. IOPT
   $= -1$ to read into the initial conditions buffer.
   $= +1$ to write out of the initial conditions buffer with line numbers.
2. NFILE (file to be read from or written on).
3. NCOL (number of items per line, must not exceed 6, ignored when IOPT = 1).
4. IFIRST (number of first line in initial conditions buffer, ignored when IOPT = -1).
5. ISTEP (difference between successive line numbers in initial conditions buffer, ignored when IOPT = -1).

Example:

```
numfile     num
 1 , 19 , 0 , 1 , 5
```

This is a command with the user given name *numfile*. It writes out the lines IFIRST + n*ISTEP (with $n = 0, 1, 2, \cdots$) of the initial conditions buffer. Each line begins with a line number m (with $m = 1, 2, 3, \cdots$) followed by 6 entries.

Description:

Sometimes it is useful to have a file in which the lines are numbered. That is, the first entry in a line is a line number m (with $m = 1, 2, 3, \cdots$) followed by additional entries. (For example, when a single particle is tracked for several turns in a storage ring, it is sometimes useful to be able to make plots of some phase-space coordinate versus turn number.) This can be accomplished by the use of a command with type code *num*.

When IOPT = 1, lines from the initial conditions buffer are written on file NFILE as described above. The format is (1x, i5, 6(1x, 1pe11.4)).

When IOPT $= -1$, the contents of file NFILE are read into the initial conditions buffer under the assumption that each line has NCOL entries with NCOL ≤ 6. If NCOL < 6, the line is padded with additional entries of zero so as to have a total of 6 entries. See section 4.4.2. If the contents of the initial conditions buffer are then written on another file (using IOPT = 1), the result will be a line-numbered version of (perhaps selected portions of) the original file.

7.28 Write Out Parameter Values in a Parameter Set

Type Code: wps

Required Parameters:

1. IPSET (an integer from 1 to 9 specifying a particular parameter set).
2. ISEND
   = 0 to do nothing.
   = 1 to write parameter values at the terminal.
   = 2 to write parameter values on file 12.
   = 3 to write parameter set values at the terminal and on file 12.

Example:

```
showpar     wps
  3          1
```

This is a command with the user given name *showpar*. It displays the values of the parameters in parameter set 3 by writing them at the terminal.

Description:

Upon occasion, it is useful to check what parameter values are present in a particular parameter set. This can be done by using a command having the type code *wps*.

7.29 Write Out Execution Time

Type Code: time

Required Parameters:

1. IOPT
   = 0 to not reset time to zero after being called.
   = 1 to reset time.
2. IFILE (file to be written on).
3. ISEND
   = 0 to not write execution time.
   = 1 to write execution time at the terminal.
   = 2 to write execution time on file IFILE.
   = 3 to write execution time at the terminal and on IFILE.

Example:

```
cputime     time
 0     12     3
```

This is a command with the user given name *cputime*. It writes the execution time at the terminal and on file 12. The time is not reset.

Description:

Upon occasion it may be useful to have cpu timing information during the course of a MaryLie run. When a MaryLie run begins, an internal cpu time clock is set to zero. The status of this clock can then be examined using a command having type code *time*. This command can also be used to reset the clock to zero.

Note:

This feature of MaryLie is not FORTRAN 77 standard. Its implementation within MaryLie depends on the computer operating system being used.

7.30 Change Drop File

Type Code: cdf

Required Parameters:

1. IFILE

Example:

```
resetdf     cdf
  32
```

This is a command with the user given name *resetdf*. It sets the dropfile, nominally file 12, to be file 32.

Description:

Several MaryLie commands write extensive output to an external file rather than the terminal. By default, file 12 is used for this purpose. See section 4.2. However, some other file can be designated for this purpose by use of a command having type code *cdf*.

7.31 Ring Bell at Terminal

Type Code: bell

Required Parameters: None

Example:

```
wakeup     bell
```

This is a command with the user given name *wakeup*. It causes the bell to ring at the terminal.

Description:

A command having type code *bell* can be used to alert the MaryLie user that a particular calculation has been completed.

7.32 Write Out Value of Merit Function

Type code: wmrt

Required Parameters

1. IFN (an integer from 0 to 5 specifying a particular merit function).
2. ISEND
   = 1 to write value at the terminal.
   = 2 to write value on file 12.
   = 3 to write value at the terminal and on file 12.

Example:

```
wmerit2     wmrt
     2 ,  3
```

This is a command with the user given name *wmerit2*. It causes the value of the merit function *mrt2* to be written both at the terminal and on file 12.

Description:

MaryLie allows the user to construct merit functions and, if desired, to use these functions in connection with an optimizer. See sections 9.8 through 9.11 and 10.3.2. Sometimes it is useful to examine the value of some or several merit functions during the course of a MaryLie calculation. This can be done with the use of a *wmrt* command.

7.33 Write Contents of a Loop

Type Code: wcl

Required Parameters:

1. IOPT
   = 1 to simply write names of loop contents at the terminal and/or file 12. In this case IFILE is ignored.
   = 2 to write on file IFILE names of loop contents and append an "&" sign at the end of each written line save for the last.
   = 3 to write on file IFILE names of loop contents in such a way that each entry is preceded and followed by a "%" sign. An "&" sign is also appended at the end of each written line save for the last.
   = 4 to write on file IFILE full loop contents.
   = 5 to write on file IFILE full loop contents along with control indices.
2. IFILE (file to be written on, an integer greater than 0).
3. ISEND
   = 1 to write at the terminal.
   = 2 to write on file 12.
   = 3 to write at the terminal and on file 12.
   Nothing is written at the terminal and/or file 12 if IOPT $\neq$ 1.

Example:

```
wcl1     wmrt
    1, 21, 3
```

This is a command with the user given name *wcl1*. It causes the contents of the current loop to be printed at the terminal and on file 12.

Description:

It is often useful to list the contents of a loop. This can be done by invoking the name of a loop in the #labor component of the master input file and then invoking some command having type code *wcl*. See Exhibit 7.33 below. The IOPT = 1 option is useful for checking the contents of a loop. The IOPT = 2 option produces MaryLie readable data that can be used in preparing some other MaryLie master input file. The IOPT = 3 option also produces MaryLie readable data with "%" signs surrounding each entry. This data can be used to prepare a MaryLie master input file for a MaryLie run that performs elaborate operations of the user's construction every time a "%" sign is encountered. See, for example, section 10.11. The options IOPT = 4 and IOPT = 5 prepare "flat files" that the advanced MaryLie user might find helpful.

```
#comment
  Exhibit 7.33.
  This is an example,for the nsex line of the PSR, of the use of the
  command with type code wcl.  All options are illustrated.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hdq      quad
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hcs      sext
   0.500000000000000      0.000000000000000E+00
  vcs      sext
   0.500000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  wcl1     wcl
    1.00000000000000       21.0000000000000       3.00000000000000
  wcl2     wcl
    2.00000000000000       22.0000000000000       3.00000000000000
  wcl3     wcl
    3.00000000000000       23.0000000000000       3.00000000000000
  wcl4     wcl
    4.00000000000000       24.0000000000000       3.00000000000000
  wcl5     wcl
    5.00000000000000       25.0000000000000       3.00000000000000
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
```

```
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
#lumps
#loops
  lring
      1*ring
  lnsex
      1*nsex
#labor
     1*fileout
     1*lnsex
     1*wcl1
     1*wcl2
     1*wcl3
     1*wcl4
     1*wcl5
     1*fin

contents of loop lnsex   ;     7 items:
  drl      hdq      drs      bend     drs
  hfq      drl

end of MARYLIE run

Contents of file 22

   drl      hdq      drs      bend     drs      &
   hfq      drl


Contents of file 23

   % drl      % hdq      % drs      % bend     % drs      &
   % hfq      % drl      %


Contents of file 24

#comment
  contents of loop lnsex
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#biglist
  drl      drft
    2.28646000000000
  hdq      quad
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
```

```
   0.000000000000000E+00
  drs      drft
   0.450000000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  drs      drft
   0.450000000000000
  hfq      quad
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
  drl      drft
    2.28646000000000


Contents of file 25

#comment
  contents of loop lnsex
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#biglist
  drl      drft         1     1     1
    2.28646000000000
  hdq      quad         4     1     9
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
   0.000000000000000E+00
  drs      drft         1     1     1
   0.450000000000000
  bend     pbnd         4     1     3
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  drs      drft         1     1     1
   0.450000000000000
  hfq      quad         4     1     9
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
  drl      drft         1     1     1
    2.28646000000000
```

7.34 Pause

Type Code: paws

Required Parameters: None

Example:

```
pause     paws
```

This is a command with the user given name *pause*. It causes a MaryLie run to pause.

Description:

During a MaryLie run it may be useful to have the program pause. This can be accomplished by using a command having the type code *paws*. When "paused", the program can be made to resume by entering any character at the terminal.

7.35 Change or Write Out Values of Infinities

Type Code: inf

Required Parameters:

1. JOB
   = 0 to change values of infinities.
   = 1 to write current values at the terminal.
   = 2 to write current values on file 12.
   = 3 to write current values at the terminal and on file 12.
2. xinf
3. yinf
4. tinf
5. ginf

Example:

```
  chinf     inf
0, 100, 100, 100, 100
```

This is a command with the user given name *chinf*. It sets the values of xinf, yinf, tinf, and ginf to 100.

Description:

The quantities xinf, yinf, tinf, and ginf are set internally to the value 1000 at the beginning of a MaryLie run. The values of xinf, yinf, and tinf are used in the raytrace routines: particles whose X, Y, τ coordinates exceed (in absolute value) xinf, yinf, tinf are marked as lost and not traced. This is done to prevent arithmetic overflow and subsequent computer crash of a MaryLie run. The quantity ginf is a "general" infinity, and is not currently used.

7.36 Change or Write Out Values of Zeroes

Type Code: zer

Required Parameters:

1. JOB
   = 0 to change values of zeroes.
   = 1 to write current values at the terminal.
   = 2 to write current values on file 12.
   = 3 to write current values at the terminal and on file 12.
2. rzero
3. fzero
4. detzero

Example:

```
czer     zer
  0, 1.d-10, 1.d-10, 1.d-10
```

This is a command with the user given name *czer*. It sets the values of rzero, fzero, and detzero to 1.d-10.

Description:

The quantities rzero, fzero, and detzero are set internally to zero at the beginning of a MaryLie run. When a map is printed using a command with type code *ptm*, entries of the R matrix are printed as zeroes if their absolute values are less than rzero, and entries in the polynomials f_1 through f_4 are not printed if their absolute values are less than fzero.

Various "purifying" routines in MaryLie try to "remove" various "offensive" terms from the transfer map, and in an attempt to do so compute various determinants. Some of these determinants vanish if tunes are resonant or equal. (MaryLie normally warns if this occurs.) The purifying routines are constructed in such a way that "offensive" terms are "removed" only if their associated determinants, in absolute value, exceed detzero. See sections 8.1 through 8.5, 8.8 through 8.11, and 8.15, 8.16.

7.37 Twiss Polynomial

Type Code: tpol

Required Parameters:

1. ALPHAX
2. BETAX
3. ALPHAY
4. BETAY
5. ALPHAτ
6. BETAτ

Note: If the BETA for some plane is specified to be ≤ 0, all coefficients for that plane are set to zero.

Example:

```
twisspol  tpol
 -7, 20, -5, 30, 0, -1
```

This is a command with user given name *twisspol*. It is used to set up a quadratic polynomial in the f_2 array of the current transfer map. See the description below.

Description:

In order to study the propagation of twiss parameters, it is useful to be able to set up the quadratic polynomial

$$\begin{aligned} f_2 &= \gamma_x X^2 + 2\alpha_x X P_x + \beta_x P_x^2 \\ &+ \gamma_y Y^2 + 2\alpha_y Y P_y + \beta_y P_y^2 \\ &+ \gamma_\tau \tau^2 + 2\alpha_\tau \tau P_\tau + \beta_\tau P_\tau^2. \end{aligned} \tag{7.37.1}$$

Here $\gamma_x, \gamma_y, \gamma_\tau$ are defined by the relations

$$\begin{aligned} \gamma_x &= (1+\alpha_x^2)/\beta_x, \\ \gamma_y &= (1+\alpha_y^2)/\beta_y, \\ \gamma_\tau &= (1+\alpha_\tau^2)/\beta_\tau. \end{aligned} \tag{7.37.2}$$

When a command having type code *tpol* is executed, the f_2 portion of the current transfer map is replaced by the f_2 given by (7.37.1). This polynomial is then available to be acted on by a command having type code *amap* and as input for a command having type code *exp*. See sections 8.17 and 8.7. See also section 6.15.

7.38 Dispersion Polynomial

Type Code: dpol

Required Parameters:

1. R_{16}
2. R_{26}
3. R_{36}
4. R_{46}
5. R_{56}

Example:

```
dispol     dpol
.1 , .2 , .3 , .4 , .5
```

This is a command with the user given name *dispol*. It is used to set up a quadratic polynomial in the f_2 array of the current transfer map. See the description below.

Description:

In order to study the propagation of dispersion functions, it is useful to be able to set up the quadratic polynomial

$$f_2 = (R_{26})XP_\tau - (R_{16})P_xP_\tau + (R_{46})YP_\tau - (R_{36})P_yP_\tau - (1/2)(R_{56})P_\tau^2. \tag{7.38.1}$$

When a command having type code *dpol* is executed, the f_2 portion of the current transfer map is replaced by the f_2 given by (7.38.1). This polynomial is then available to be acted on by a command having type code *amap* and as input for a command having type code *exp*. See sections 8.17 and 8.7. See also section 6.22.

7.39 Change or Write Out Beam Parameters

Type Code: cbm

Required Parameters:

1. JOB
   = 0 to write out beam parameters and fill related arrays.
   = 1 to compute beam parameters (save for scale length) based on values of energy and particle KIND and fill related arrays.
   = 2 to compute beam parameters (save for scale length) based on values of momentum and particle KIND and fill related arrays.
   = 3 to compute beam parameters (save for scale length) based on values of BRHO and particle KIND and fill related arrays.
2. ISEND
   = 0 to not write out results.
   = 1 to write at the terminal.
   = 2 to write on file 12.
   = 3 to do both.
3. Energy or Momentum or BRHO. If JOB=1, this third parameter is interpreted as kinetic energy (in MeV). If JOB=2, this third parameter is interpreted as momentum (in MeV/c). If JOB=3, this third parameter is interpreted as BRHO (in Tesla-meters).
4. KIND
   KIND = 0 to use mass and charge values specified by parameters 5 and 6.
   KIND = 1 for electron/positron.
   KIND = 2 for proton/antiproton.
   KIND = 3 for H^- ion.
   KIND = 4 for deuteron.
   KIND = 5 for triton.
   KIND = 6 for alpha (singly ionized He 4).
   KIND = 7 for muon.
   KIND = 8 for pion.
5. Particle mass (in MeV/c^2).
6. |(Particle charge)/e|, the absolute value of the charge of the particle in units of e.

Example:

```
newbm     cbm
  1.0 , 1.0 , 800.0, 0 , 7.0, 8.0
```

This is a command with the user given name *newbm*. It is used to specify a beam having a kinetic energy of 800 MeV. The beam consists of fictitious particles having a mass of 7 MeV/c^2 and charge 8.

Description:

It is sometimes convenient to be able to change the beam parameters during the course of a MaryLie run. In particular, it may be desirable to fit on beam energy, beam momentum, or particle mass, etc. This can be done using a command with type code *cbm*. Exhibit 7.39 illustrates the use of this type code simply to specify beam parameters.

```
#comment
  Exhibit 7.39.
  This is an example of the use of the type code cbm.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
   1.00000000000000
#menu
  cbme     cbm
    1.00000000000000       1.00000000000000      800.000000000000
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  cbmp     cbm
    2.00000000000000       1.00000000000000       1463.00000000000
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  cbmr     cbm
    3.00000000000000       1.00000000000000       4.88103064647049
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  cbmmc    cbm
    1.00000000000000       1.00000000000000       800.000000000000
   0.000000000000000E+00   7.00000000000000       8.00000000000000
  qcbmmc   cbm
    1.00000000000000      0.000000000000000E+00   800.000000000000
   0.000000000000000E+00   7.00000000000000       8.00000000000000
  seebeam  cbm
   0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  qseebeam cbm
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*seebeam
     1*cbmmc
     1*seebeam
     1*fileout
     1*end
```

```
     1*cbme
     1*seebeam
     1*fileout
     1*cbmp
     1*seebeam
     1*fileout
     1*cbmr
     1*seebeam
     1*fileout
     1*end

brho, gamm1, and achg beam parameters are:
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

design energy and momentum:
KIND =            0
rest mass (MeV/c*c) =    7.00000000000000
kinetic energy (MeV) =    800.000000000000
momentum (MeV/c) =    806.969640073281
values for brho, beta, gamma, (gamma-1), |q/e|, and scale length are
  0.336470122304278
  0.999962379272962
   115.285714285714
   114.285714285714
   8.00000000000000
   1.00000000000000

brho, gamm1, and achg beam parameters are:
  0.336470122304278        114.285714285714        8.00000000000000

#comment
  Exhibit 7.39.
  This is an example of the use of the type code cbm.
#beam
  0.336470122304278
   114.285714285714
   8.00000000000000
   1.00000000000000
#menu
  cbme     cbm
    1.00000000000000       1.00000000000000      800.000000000000
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  cbmp     cbm
    2.00000000000000       1.00000000000000       1463.00000000000
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  cbmr     cbm
    3.00000000000000       1.00000000000000       4.88103064647049
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  cbmmc    cbm
    1.00000000000000       1.00000000000000       800.000000000000
   0.000000000000000E+00   7.00000000000000       8.00000000000000
  qcbmmc   cbm
    1.00000000000000      0.000000000000000E+00   800.000000000000
```

```
   0.000000000000000E+00   7.00000000000000       8.00000000000000
  seebeam  cbm
   0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  qseebeam cbm
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*seebeam
     1*cbmmc
     1*seebeam
     1*fileout
     1*end
     1*cbme
     1*seebeam
     1*fileout
     1*cbmp
     1*seebeam
     1*fileout
     1*cbmr
     1*seebeam
     1*fileout
     1*end
```

7.40 Dimensions

Type Code: dims

Required Parameters:

1. DIM1
2. DIM2
3. DIM3
4. DIM4
5. DIM5
6. DIM6

Example:

```
size     dims
  1 , 1 , 2 , 2 , 3 , 3
```

This is a command with the user given name *size*. It specifies six dimensions.

Description:

Commands with the type code *dims* have no effect on a MaryLie run. In this regard they are like markers. See section 6.25. However they are recognized by *geom* commands and can be used to provide information such as aperture size to plotting routines such as POSTER. See sections 1.4.2 and 8.38.

7.41 Write Out Contents of the UCALC Array

Type Code: wuca

Required Parameters:

1. KMIN (index of first entry in UCALC to be written out).
2. KMAX (index of last entry in UCALC to be written out).
3. ISEND
   = 0 to do nothing.
   = 1 to write UCALC values at the terminal.
   = 2 to write UCALC values on file 12.
   = 3 to write UCALC values at the terminal and on file 12.
4. IFILE
   = 0 to not write.
   = positive integer to write UCALC values to that file.

Example:

```
results     wuca
   2 , 5 , 3 , 0
```

This is a command with the user given name *results*. When invoked, it causes the values in locations 2 through 5 of the UCALC array to be written at the terminal and on file 12.

Description:

User written subroutines and merit functions have access to a UCALC array. See sections 6.20, 6.21, and 9.11. The contents of this array can be used in fitting. See section 9.7. Commands with type code *wuca* allow the examination and writing out of the contents of this array.

Exhibit 7.41 below shows a simple user-written program that inserts values into UCALC and a simple MaryLie program that prints them out. See also Exhibit 8.38.

```
Exhibit 7.41

Simple user14 program that puts values in ucalc

**********************************************************************
c
       subroutine user14(p,fa,fm)
c
       include 'impli.inc'
       include 'usrdat.inc'
c
       do i=1,5
```

```
      ucalc(i)=i
      enddo
c
      return
      end
c
**********************************************************************

Simple MaryLie run

#comment
  Exhibit 7.41.
  This is a MARYLIE run to test the wuca command
#beam
   1.00000000000000
   1.00000000000000
   1.00000000000000
   1.00000000000000
#menu
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  fill     usr14
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  look     wuca
    1.00000000000000       10.0000000000000       3.00000000000000
   0.000000000000000E+00
  wfile21  wuca
    1.00000000000000       10.0000000000000      0.000000000000000E+00
    21.0000000000000
  end      end
#labor
     1*fileout
     1*fill
     1*look
     1*end
  k and ucalc(k) for nonzero values of array
          1   1.00000000000000
          2   2.00000000000000
          3   3.00000000000000
          4   4.00000000000000
          5   5.00000000000000

end of MARYLIE run
```

7.42 Window a Beam in All Planes

Type Code: wnda

Required Parameters:

1. AX (largest allowed value of $|X|$).
2. APX (largest allowed value of $|P_x|$).
3. AY (largest allowed value of $|Y|$).
4. APY (largest allowed value of $|P_y|$).
5. AT (largest allowed value of $|\tau|$).
6. APT (largest allowed value of $|P_\tau|$).

Example:

```
allwnd     wnda
 .1, .2, .15, .25, .01, .02
```

This specifies a command with the user given name *allwnd*. It removes particles whose X coordinates lie outside $(-AX, AX)$, or whose P_x coordinates lie outside $(-APX, APX)$, etc.

Description:

This command is similar to that of section 7.20 except that all planes are treated at once at the expense of the window being centered on the phase-space origin. See section 7.20.

7.43 Path Length Information

Type Code: pli

Required Parameters:

1. SEND
   = 0 simply to compute real and scaled time of flight and place path length and time of flight information in the fitting/writing array.
   = 1 to do the same and also write results at the terminal.
   = 2 to do the same and also write results on file 12.
   = 3 to do the same and also write results at the terminal and on file 12.

Example:

```
sinfo     pli
   3
```

This specifies a command with the user name *sinfo*. It causes the real and scaled time of flight to be computed, the placement of path length and time of flight information in the fitting/writing array, and the writing of results at the terminal and file 12.

Description:

When MaryLie computes the map for an element, it also finds the path length of the design orbit through this element, and regards this information as part of the map. When maps are concatenated, their path lengths are added. See sections 7.9 and 7.8 for the effect of commands with the type codes *inv* and *iden* on path length information.

When a command with type code *pli* is invoked, real and scaled time of flight are computed from the path length. They are defined by the relations

$$\text{real time of flight} = (\text{path length})/v^0,$$

$$\text{scaled time of flight} = (\text{real time of flight})(c/\ell).$$

Here v^0, c, and ℓ are the design velocity, the velocity of light, and the scale length. See sections 4.1.1 and 4.1.2. Path length and time of flight information are also placed in the fitting/writing array. Finally, they may also be written at the terminal and/or file 12. See Exhibit 7.43 below.

```
#comment
  Exhibit 7.43.
  This is a MaryLie run that illustrates use of the pli command.  It
  computes the path length for the ring of Exhibit 2.5.1.  It also
```

```
  illustrates that inverting the map changes the sign of the  path
  length, and setting the map to the identity sets the path length
  to zero.  The beam parameters are those for 800 MeV protons.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hdq      quad
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hcs      sext
   0.500000000000000      0.000000000000000E+00
  vcs      sext
   0.500000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  iden     iden
  inv      inv
  sinfo    pli
    3.00000000000000
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
```

```
      2*half
#lumps
#loops
#labor
     1*fileout
     1*sinfo
     1*ring
     1*sinfo
     1*inv
     1*sinfo
     1*ring
     1*sinfo
     1*iden
     1*sinfo
     1*fin

   path length            real time of flight      scaled time of flight:
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

   path length            real time of flight      scaled time of flight:
   90.2239999999613       3.577642009201167E-007   107.255009178248

   path length            real time of flight      scaled time of flight:
  -90.2239999999613      -3.577642009201167E-007  -107.255009178248

   path length            real time of flight      scaled time of flight:
-2.975397705995420E-014 -1.179831067903725E-022 -3.537044558716225E-014

   path length            real time of flight      scaled time of flight:
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

end of MARYLIE run
```

7.44 Show Contents of Arrays

Type Code: shoa

Required Parameters:

1. JOB
   = 1 to show dispersion and phase-slip array.
   = 2 to show betatron amplitudes array.
   = 3 to show tunes array.
   = 4 to show twiss parameter array.
   = 5 to show envelope array.
   = 6 to show EX array.
   = 7 to show fitdat array.
2. ISEND
   = 0 to do nothing.
   = 1 to write array values at the terminal.
   = 2 to write array values on file 12.
   = 3 to write array values at the terminal and on file 12.
3. IFILE
   = 0 to not write.
   = positive integer to write array values to that file.

Example:

```
display     shoa
     1 , 1 , 0
```

This is a command with the user given name *display*. When invoked, it causes the values in the dispersion and phase-slip array to be written at the terminal.

Description:

The commands *cod*, *tasm*, and *tadm* fill various arrays for subsequent fitting, optimization, or writing. See sections 9.5, 9.7, 8.26, 8.1, 8.3, and 8.2. An *extra* array is used to store various quantities computed by invoking various other commands. These quantities and commands are listed below.

ex(1) = momentum compaction. See *cod*, section 8.1.

ex(2) = transition gamma. See *cod*, section 8.1.

ex(3) = polynomial scalar product. See *psp*, section 8.32.

ex(4) = symplectic violation. See *csym*, section 8.31.

ex(5) = matrix norm. See *mn*, section 8.33.

The fitdat array contains the quantities listed below:

```
 common/fitdat/
$tux, tuy, tut, cx, cy, qx, qy, hh, vv, tt,
$hv,  ht,  vt,  fx, xb, xa, xu, xd, fy, yb,
$ya,  yu,  yd,  pl, rt, st, ek, pd, br, be,
$ga,  gm
 dimension fitval(32)
 equivalence (fitval,tux)
```

Commands with type code *shoa* allow the examination and writing of the contents of these arrays.

Chapter 8

Catalog of Advanced Commands

Because MaryLie works with explicit representations of transfer maps, there are almost unlimited possibilities for commands to manipulate and analyze these maps. Listed below are the advanced map analysis and manipulation commands, with their type-code mnemonics, that are currently available in MaryLie 3.0. Also listed are the subsections that describe them in detail. It is expected that additional advanced commands will be added in the future.

Type Code	Command	Subsection
cod	Compute off-momentum closed orbit data.	8.1
tasm	Twiss analyze static map.	8.2
tadm	Twiss analyze dynamic map.	8.3
rasm	Resonance analyze static map.	8.4
radm	Resonance analyze dynamic map.	8.5
tbas	Translate basis.	8.6
exp	Compute exponential.	8.7
snor	Static normal form analysis.	8.8
dnor	Dynamic normal form analysis.	8.9
sia	Static invariant analysis.	8.10
dia	Dynamic invariant analysis.	8.11
psnf	Compute power of static normal form.	8.12
pdnf	Compute power of dynamic normal form.	8.13
gbuf	Get buffer contents.	8.14
fasm	Fourier analyze static map.	8.15
fadm	Fourier analyze dynamic map.	8.16

Type Code	Command	Subsection
amap	Apply map to a function or moments.	8.17
smul	Multiply a polynomial by a scalar.	8.18
padd	Add two polynomials.	8.19
pmul	Multiply two polynomials.	8.20
pb	Poisson bracket two polynomials	8.21
pold	Polar decomposition of a map.	8.22
pval	Evaluate a polynomial.	8.23
cxp	Compute Chromatic Expansion.	8.24
tqm	Transfer quadratic moments.	8.25
sq	Select quantities.	8.26
wsq	Write selected quantities.	8.27
ctr	Change tune range.	8.28
asni	Apply script N inverse.	8.29
pnlp	Compute power of nonlinear part.	8.30
csym	Check symplectic condition.	8.31
psp	Polynomial scalar product.	8.32
mn	Compute matrix norm.	8.33
bgen	Generate beam.	8.34
tic	Translate initial conditions.	8.35
ppa	Principal planes analysis.	8.36
moma	Moment analysis.	8.37
geom	Compute geometry of a loop.	8.38
fwa	Copy file to working array.	8.39
merf	Merge files.	8.40
lnf	Compute logarithm of normal form.	8.41

Note that the type codes are given in lower case. If entries are made in upper case, they are automatically converted to lower case by PREP and MaryLie.

The purpose of this section is to outline the use of these advanced commands by MaryLie, and to describe the parameters required to specify these commands in the #menu component of the Master Input File.

8.1 Closed Orbit Data

Type Code: cod

Required Parameters:

1. IOPT (controls meaning of the second parameter, EPSILON)
   = 1 if EPSILON = P_τ.
   = 2 if EPSILON = δ. See section 4.1.2.
2. EPSILON (value of expansion parameter).
3. IDATA
   = 0 to not put out analysis data.
   = 1 to put out closed orbit information.
   = 2 to put out transfer matrix expansion.
   = 3 to put out closed orbit information and transfer matrix expansion.
4. IPMAPS
   = 0 to not put out maps.
   = 1 to put out linear and nonlinear factors of closed-orbit transfer map.
   = 2 to put out transformation map $\mathcal{T}$.
   = 3 to put out factors and transformation map.
5. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
   Note: When ISEND = -1, -2, or -3, the result is the same as when ISEND = $+1$, $+2$, or $+3$, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
6. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps on file IFILE.

Example:

```
offporb    cod
2 , .001 , 3 , 3 , 1 , 0
```

This specifies a command with the user given name *offporb*. It prints out closed orbit information, the transfer matrix expansion, and all maps.

Description:

At any given location (Poincare surface of section) in a static (no RF) ring, the closed orbit has an expansion of the form

$$X(\epsilon) = \epsilon D_x^{(1)} + \epsilon^2 D_x^{(2)} + \epsilon^3 D_x^{(3)} + \cdots, \tag{8.1.1}$$

$$P_x(\epsilon) = \epsilon E_x^{(1)} + \epsilon^2 E_x^{(2)} + \epsilon^3 E_x^{(3)} + \cdots, \tag{8.1.2}$$

$$Y(\epsilon) = \epsilon D_y^{(1)} + \epsilon^2 D_y^{(2)} + \epsilon^3 D_y^{(3)} + \cdots, \tag{8.1.3}$$

$$P_y(\epsilon) = \epsilon E_y^{(1)} + \epsilon^2 E_y^{(2)} + \epsilon^3 E_y^{(3)} + \cdots. \tag{8.1.4}$$

The quantities $D^{(1)}$, $D^{(2)}$, $\cdots$ are first, second, and third-order dispersion functions, etc. The quantities $E^{(1)}$, $E^{(2)}$, $\cdots$ are their momentum counterparts (sometimes called "D primed"). Here the expansion parameter ϵ is given by the relation

$$\epsilon = P_\tau \text{ if IOPT } = 1, \tag{8.1.5}$$

$$\epsilon = \delta \text{ if IOPT } = 2. \tag{8.1.6}$$

Similarly, the scaled *differential* time of flight for the closed orbit has an expansion of the form

$$\tau(\epsilon) = \epsilon\tau^{(1)} + \epsilon^2\tau^{(2)} + \epsilon^3\tau^{(3)} + \cdots. \tag{8.1.7}$$

Unlike the dispersion functions, the values of the quantities $\tau^{(1)}$, $\tau^{(2)}$, $\cdots$ are independent of the ring location at which they are computed. Note that expansions of the form (8.1.1) through (8.1.4) and (8.1.7) for the case IOPT=1 are related to those for the case IOPT=2, and vice versa, as a result of (4.1.9) and (4.1.10).

The *actual* (as contrasted to differential) time of flight for the *design* closed orbit can be found using the *pli* command. See section 7.43. Let $\tau^{(0)}$ be the scaled actual time of flight for the design closed orbit, and let $\tau_{\text{tot}}(\epsilon)$ be the scaled *total* time of flight for the general closed orbit. Then there is the relation

$$\tau_{\text{tot}}(\epsilon) = \tau^{(0)} + \tau(\epsilon) = \tau^{(0)} + \epsilon\tau^{(1)} + \epsilon^2\tau^{(2)} + \epsilon^3\tau^{(3)} + \cdots. \tag{8.1.8}$$

There is no standard terminology for the quantities $\tau^{(1)}$, $\tau^{(2)}$, $\cdots$. They are variously called *phase slip* factors or *temporal dispersion* factors.

There is an intimate relation between the phase slip factors and the so called momentum compaction factors. Let $C(\epsilon)$ be the circumference of the general closed orbit. Evidently there is the relation

$$C(\epsilon) = v(\epsilon)\tau_{\text{tot}}(\epsilon)(\ell/c). \tag{8.1.9}$$

where $v(\epsilon)$ is the particle velocity on the general closed orbit. [Recall that τ_{tot} is the *scaled* time of flight. See (4.1.8).] The quantities $C(\epsilon)$ and $v(\epsilon)$ also have ϵ expansions of the form

$$C(\epsilon) = C^{(0)} + \epsilon C^{(1)} + \epsilon^2 C^{(2)} + \cdots, \tag{8.1.10}$$

$$v(\epsilon) = v^{(0)} + \epsilon v^{(1)} + \epsilon^2 v^{(2)} + \cdots. \tag{8.1.11}$$

Here $C^{(0)}$ and $v^{(0)}$ are the circumference of and velocity on the design orbit, respectively. The circumference of the design orbit can be found using the *pli* command. The velocity expansion coefficients $v^{(0)}$, $v^{(1)}$, $\cdots$ can be obtained from (4.1.1) through (4.1.8). For v as a function of P_τ one finds the expression

$$v = c\{1 - 1/[\gamma_0^2(1 - P_\tau\beta_0)^2]\}^{1/2}. \tag{8.1.12}$$

Correspondingly for $\epsilon = P_\tau$ (IOPT=1) the expansion coefficients are

$$v^{(0)} = \beta_0 c, \tag{8.1.13}$$

$$v^{(1)} = -v^{(0)}/(\beta_0\gamma_0^2), \tag{8.1.14}$$

$$v^{(2)} = v^{(0)}(\beta_0^4\gamma_0^2 + \beta_0^2 - 4\beta_0^2\gamma_0^2 - 1)/(2\beta_0^2\gamma_0^4), \tag{8.1.15}$$

$$v^{(3)} = v^{(0)}(\beta_0^2 - 4\beta_0^2\gamma_0^2 + 5\beta_0^4\gamma_0^2 - 8\beta_0^4\gamma_0^4 + 4\beta_0^6\gamma_0^4 - 1)/(2\beta_0^3\gamma_0^6), \text{ etc.} \tag{8.1.16}$$

For v as a function of δ one finds the expression

$$v = \beta_0 c(1 + \delta)(1 + 2\beta_0^2\delta + \beta_0^2\delta^2)^{-1/2}. \tag{8.1.17}$$

Correspondingly for $\epsilon = \delta$ (IOPT=2) the expansion coefficients are

$$v^{(0)} = \beta_0 c, \tag{8.1.18}$$

$$v^{(1)} = v^{(0)}(1 - \beta_0^2) = v^{(0)}/\gamma_0^2, \tag{8.1.19}$$

$$v^{(2)} = -3v^{(0)}\beta_0^2/(2\gamma_0^2), \tag{8.1.20}$$

$$v^{(3)} = 5v^{(0)}\beta_0^6/(2\gamma_0^2) - 15v^{(0)}\beta_0^4/(8\gamma_0^4), \text{ etc.} \tag{8.1.21}$$

Note that $v^{(1)}$, $v^{(2)}$, $\cdots$ all vanish, as expected, in the relativistic limit $\beta_0 \to 1$, $\gamma_0 \to \infty$. Inserting the expansions (8.1.8), (8.1.10), and (8.1.11) into (8.1.9), and equating powers of ϵ, give the relations

$$C^{(0)} = (\ell/c)v^{(0)}\tau^{(0)}, \tag{8.1.22}$$

$$C^{(1)} = (\ell/c)(v^{(0)}\tau^{(1)} + v^{(1)}\tau^{(0)}), \tag{8.1.23}$$

$$C^{(2)} = (\ell/c)(v^{(0)}\tau^{(2)} + v^{(1)}\tau^{(1)} + v^{(2)}\tau^{(0)}), \tag{8.1.24}$$

$$C^{(3)} = (\ell/c)(v^{(0)}\tau^{(3)} + v^{(1)}\tau^{(2)} + v^{(2)}\tau^{(1)} + v^{(3)}\tau^{(0)}), \text{ etc.} \tag{8.1.25}$$

Thus, if the $\tau^{(i)}$ are known, the $C^{(i)}$ can be found.

In the case $\epsilon = \delta$ (IOPT=2), the coefficients $C^{(1)}$, $C^{(2)}$, $\cdots$ may be called momentum compaction factors. For example, when $\epsilon = \delta$, $C^{(1)}$ is given by the relation

$$C^{(1)} = (\ell/c)(v^{(0)}\tau^{(1)} + v^{(1)}\tau^{(0)}) = (\ell/c)[v^{(0)}\tau^{(1)} + (v^{(0)}/\gamma_0^2)\tau^{(0)}]. \tag{8.1.26}$$

This relation can be rewritten in the form

$$\tau^{(1)}/\tau^{(0)} = cC^{(1)}/(\ell v^{(0)}\tau^{(0)}) - 1/\gamma_0^2 = C^{(1)}/C^{(0)} - 1/\gamma_0^2. \tag{8.1.27}$$

Here use has been made of (8.1.22). The quantity $C^{(1)}/C^{(0)}$ is commonly called the (first order) *momentum compaction*.

Consider a ring. Suppose γ_0 is varied (particles are accelerated or decelerated) while at the same time the strengths of all magnets are scaled in such a way that the design orbit remains unchanged. For many rings (but not all), the momentum compaction $C^{(1)}/C^{(0)}$ is positive and largely unaffected by the changes in γ_0 and magnet strengths just described. In this case it is useful to define a quantity γ_t, called the *transition* gamma, by the relation

$$\gamma_t^2 = C^{(0)}/C^{(1)}. \tag{8.1.28}$$

With this definition (8.1.27) takes the form

$$\tau^{(1)}/\tau^{(0)} = 1/\gamma_t^2 - 1/\gamma_0^2. \tag{8.1.29}$$

Evidently for such a ring the phase slip factor $\tau^{(1)}/\tau^0$ is negative for $\gamma_0 < \gamma_t$, zero when $\gamma_0 = \gamma_t$, and positive when $\gamma_0 > \gamma_t$. Since the sign of $\tau^{(1)}/\tau^{(0)}$ affects the stability of synchrotron oscillations, it is necessary for such rings to reverse the RF phase as particles are accelerated through transition. See sections 2.6 and 6.13. Of course, (8.1.29) remains true even if γ_t does change with γ_0. However, it is then no longer useful for estimating the energy at which transition actually occurs. We also note that it is possible to design a ring for which the momentum compaction is *negative*. For such a ring transition never occurs.

Some codes do not have a provision for handling the full 6-dimensional phase space, but do compute the (first-order) momentum compaction $C^{(1)}/C^{(0)}$; and from $C^{(1)}/C^{(0)}$ one can find the phase slip factor $\tau^{(1)}/\tau^{(0)}$ using (8.1.27). For most purposes, such as computing time of flight in spectrometers or synchrotron oscillations in dynamic lattices with RF, it is more useful to have direct access to phase slip information. (See, for example, section 10.6.2, where a complete third-order achromat is constructed.) For this reason *cod* calculates and displays the quantities $\tau^{(0)}$, $\tau^{(1)}$, $\tau^{(2)}$, $\cdots$ directly. However, as a convenience to the user, it also calculates and displays the quantities $C^{(0)}$, $C^{(1)}$, $C^{(2)}$ $\cdots$. Finally, for easy comparison with other codes, *cod* also calculates the first-order momentum compaction and the transition gamma. Note that the momentum compaction is given by $C^{(1)}/C^{(0)}$, and (1.28) holds, only in the case $\epsilon = \delta$.

If desired *cod* also provides, for the *transverse* variables (X, P_x, Y, P_y), an ϵ expansion of the linear (matrix) part of the transfer map about the closed orbit in the form

$$R_{Tr}(\epsilon) = R_{Tr}^{(0)} + \epsilon R_{Tr}^{(1)} + \epsilon^2 R_{Tr}^{(2)} + \cdots. \tag{8.1.30}$$

Here the subscript Tr stands for *transverse*, and indicates that the matrices $R_{Tr}^{(i)}$ should be 4×4. However, for programming simplicity, 6×6 matrices are used with 0 entries everywhere save for the upper left 4×4 blocks.

The computations associated with *cod* involve the determination of various maps. Some of these maps are placed in buffers where, if desired, they can be used as input for the further computations. These maps and buffers are listed below:

1. Buffer 1 contains the total transfer map about the closed orbit.

2. Buffer 2 contains $\mathcal{B}$, the betatron part of the map about the closed orbit.

3. Buffer 3 contains $\mathcal{C}$, the nonlinear part of the map about the closed orbit.

4. Buffer 4 contains in its matrix part the matrix $R_{Tr}(\epsilon)$ evaluated for the selected option and the specified value of ϵ. The array part of buffer 4 is empty.

5. Buffer 5 contains the transforming map $\mathcal{T}$ to the closed orbit.

6. Buffer 6 contains $\mathcal{N}$, the normal form for $\mathcal{M}$.

Terminology:

Let $\mathcal{M}$ be a static transfer map. Then $\mathcal{M}$ can be written in the form

$$\mathcal{M} = \mathcal{T}^{-1}\mathcal{B}\mathcal{C}\mathcal{T}. \tag{8.1.31}$$

The map $\mathcal{T}$ (strictly speaking, $\mathcal{T}^{-1}$) is the transforming map to the closed orbit. It is contained in buffer 5. The product $\mathcal{B}\mathcal{C}$ is the total transfer map about the closed orbit, and is contained in buffer 1. The maps $\mathcal{B}$ and $\mathcal{C}$ (contained in buffers 2 and 3) are the betatron and nonlinear parts, respectively, of the transfer map about the closed orbit. Specifically, the maps $\mathcal{T}$, $\mathcal{B}$, and $\mathcal{C}$ have the following properties:

- The map $\mathcal{T}$ is of the form
$$\mathcal{T} = e^{:t_2:}e^{:t_3:}e^{:t_4:}. \tag{8.1.32}$$
Here t_2 is linear in P_τ (and linear in the "geometric" coordinates X, P_x, etc.); t_3 is quadratic in P_τ (and linear in the geometric coordinates); and t_4 is cubic in P_τ (and linear in the geometric coordinates). All transverse closed-orbit dispersion information is derived from $\mathcal{T}$. This map is placed in buffer 5.

- The map $\mathcal{B}$ is of the form
$$\mathcal{B} = e^{:b_2:}e^{:b_3:}e^{:b_4:}. \tag{8.1.33}$$
Here b_2 contains only terms that are quadratic in the geometric coordinates or terms that are quadratic in P_τ; b_3 contains only terms that are linear in P_τ (and quadratic in the geometric coordinates) or terms that are cubic in P_τ; b_4 contains only terms that are quadratic in P_τ (and quadratic in the geometric coordinates) or terms that are quartic in P_τ. The map $\mathcal{B}$ evidently describes (since the geometric dependence of the b_n is purely quadratic) infinitesimal betatron oscillations about the closed orbit. The transverse transfer-matrix expansion $R_{Tr}(\epsilon)$ is derived from $\mathcal{B}$.

- The map $\mathcal{C}$ is of the form
$$\mathcal{C} = e^{:c_3:}e^{:c_4:}. \tag{8.1.34}$$
Here c_3 is independent of P_τ (contains only geometric coordinates); and c_4 contains only terms that are quartic in the geometric coordinates or terms that are linear in P_τ and cubic in the geometric coordinates.

The static transfer map $\mathcal{M}$ can also be written in the normal form factorization

$$\mathcal{M} = \mathcal{A}^{-1}\mathcal{N}\mathcal{A} \tag{8.1.35}$$

where $\mathcal{A}$ is the normalizing map and $\mathcal{N}$ is the normal form. The normal form $\mathcal{N}$ contains, among other things, time-of-flight (phase-slip) information for the closed orbit. This map is placed in buffer 6. For further discussion, see sections 8.2 and 8.8.

Note: All quantities computed by *cod* are available for fitting or optimization or scanning, etc., except for the closed orbit transfer matrix expansion (1.30). The (first order) momentum compaction and the transition gamma are placed in the first and second entries, respectively, of the extra array EX. See section 7.44. The other quantities are placed in specially reserved locations. See section 9.5. Finally, if desired, the closed orbit transfer matrix expansion can be made available for fitting etc. by the subsequent use of a *cxp* command. See section 8.24.

The MaryLie run shown in the Exhibit below illustrates the use of the *cod* command for both choices of IOPT, and demonstrates that the closed orbit claimed by *cod* does indeed close and have the predicted scaled time-of-flight deviation when raytraced.

```
#comment
  Exhibit 8.1.
  This is a test of the type code cod using the
  small static storage ring of Exhibit 2.5.1.
  It tests both cod options. Finally, it raytraces
  the closed orbit claimed by ecod, and verifies that
  the orbit does close and has the predicted time of
  flight deviation.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hdq      quad
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hcs      sext
   0.500000000000000      0.000000000000000E+00
```

```
  vcs      sext
   0.500000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  rt       rt
   -1.00000000000000       14.0000000000000       4.00000000000000
   0.000000000000000E+00   1.00000000000000       2.00000000000000
  coddata  ps1
  -3.882975000000000E-03  4.088627100000000E-04  0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  1.000000000000000E-03
  clear    iden
  ecod     cod
    1.00000000000000      1.000000000000000E-03   3.00000000000000
   0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
  mcod     cod
    2.00000000000000     -1.188970448993132E-03   3.00000000000000
   0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
#lumps
#loops
#labor
     1*fileout
     1*ring
     1*mcod
     1*ecod
     1*coddata
     1*rt
     1*fin

closed orbit analysis for static map
  P sub tau =   9.999999999999534E-004
  delta =  -1.188970448993132E-003

closed orbit data for epsilon defined in terms of momentum deviation:

location of closed orbit (x,px,y,py):
```

```
linear in epsilon expansion coefficients
  0.32666874D+01  -0.34410094D+00   0.00000000D+00   0.00000000D+00

quadratic in epsilon expansion coefficients
  0.72077913D+00  -0.18610788D+00   0.00000000D+00   0.00000000D+00

cubic in epsilon expansion coefficients
-0.47754332D+00   0.23372737D-01   0.00000000D+00   0.00000000D+00

location of closed orbit when epsilon = -0.11889704D-02
-0.38829750D-02   0.40886271D-03   0.00000000D+00   0.00000000D+00

time of flight and path length data:

transition gamma =  0.21863909D+01

first order momentum compaction =  0.20919167D+00

order by order scaled time of flight expansion coefficients:
        0th              1st              2nd              3rd
  0.10725501D+03  -0.89208118D+01   0.50849003D+02  -0.61731843D+02

order by order time of flight expansion coefficients:
        0th              1st              2nd              3rd
  0.35776420D-06  -0.29756625D-07   0.16961402D-06  -0.20591526D-06

order by order path length expansion coefficients:
        0th              1st              2nd              3rd
  0.90224000D+02   0.18874109D+02   0.12581320D+02  -0.13405701D+02

deviation values when epsilon = -0.11889704D-02
   scaled t-o-f        t-o-f          path length
  0.10678568D-01   0.35619869D-10  -0.22422950D-01

total values when epsilon = -0.11889704D-02
   scaled t-o-f        t-o-f          path length
  0.10726569D+03   0.35779982D-06   0.90201577D+02


twiss matrix for epsilon defined in terms of momentum deviation:

on energy/momentum betatron matrix

matrix for map is :

  8.79899E-01  6.43546E+00  0.00000E+00  0.00000E+00  0.00000E+00    0.00000E+00
 -2.82743E-01 -9.31451E-01  0.00000E+00  0.00000E+00  0.00000E+00   -1.11022E-16
  0.00000E+00  0.00000E+00 -9.04734E-01  6.22828E+00  0.00000E+00    0.00000E+00
  0.00000E+00  0.00000E+00 -2.82058E-01  8.36415E-01  0.00000E+00    0.00000E+00
  1.66533E-16  4.44089E-15  0.00000E+00  0.00000E+00  1.00000E+00    1.06047E+01
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00    1.00000E+00
```

```
matrix for epsilon correction

matrix for map is :

  5.98462E+00 -1.05160E+00  0.00000E+00  0.00000E+00  0.00000E+00   4.37861E-15
 -1.36443E-01  5.67524E+00  0.00000E+00  0.00000E+00  0.00000E+00  -6.51891E-16
  0.00000E+00  0.00000E+00  1.27183E+01  4.29923E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00 -1.01638E-01  1.37979E+01  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00


matrix for epsilon**2 correction

matrix for map is :

 -2.12574E+01 -1.07902E+02  0.00000E+00  0.00000E+00  0.00000E+00  -4.64722E-15
  4.82183E+00  8.99909E+00  0.00000E+00  0.00000E+00  0.00000E+00   2.56599E-15
  0.00000E+00  0.00000E+00  6.38093E+01 -5.52205E+02  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  2.48175E+01 -8.95622E+01  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00


value of betatron matrix when epsilon= -0.11889704D-02

matrix for map is :

  8.72754E-01  6.43655E+00  0.00000E+00  0.00000E+00  0.00000E+00  -5.21261E-18
 -2.82574E-01 -9.38186E-01  0.00000E+00  0.00000E+00  0.00000E+00  -1.10244E-16
  0.00000E+00  0.00000E+00 -9.19765E-01  6.22238E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00 -2.81902E-01  8.19883E-01  0.00000E+00   0.00000E+00
  1.66533E-16  4.44089E-15  0.00000E+00  0.00000E+00  1.00000E+00   1.06047E+01
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   1.00000E+00

closed orbit analysis for static map
  P sub tau =   1.000000000000000E-003
  delta =  -1.188970448993188E-003

closed orbit data for epsilon defined in terms of P sub tau:

location of closed orbit (x,px,y,py):

linear in epsilon expansion coefficients
-0.38833191D+01   0.40905468D+00   0.00000000D+00   0.00000000D+00

quadratic in epsilon expansion coefficients
  0.34374492D+00  -0.19191582D+00   0.00000000D+00   0.00000000D+00

cubic in epsilon expansion coefficients
  0.35402618D+00  -0.46168413D-01   0.00000000D+00   0.00000000D+00
```

```
location of closed orbit when epsilon =  0.10000000D-02
-0.38829750D-02   0.40886271D-03   0.00000000D+00   0.00000000D+00

time of flight and path length data:

transition gamma =  0.21863909D+01

first order momentum compaction =  0.20919167D+00

order by order scaled time of flight expansion coefficients:
        0th              1st              2nd              3rd
  0.10725501D+03   0.10604737D+02   0.73700580D+02   0.13086918D+03

order by order time of flight expansion coefficients:
        0th              1st              2nd              3rd
  0.35776420D-06   0.35373594D-07   0.24583867D-06   0.43653261D-06

order by order path length expansion coefficients:
        0th              1st              2nd              3rd
  0.90224000D+02  -0.22436855D+02   0.13880402D+02   0.24064698D+02

deviation values when epsilon =  0.10000000D-02
   scaled t-o-f        t-o-f          path length
  0.10678568D-01   0.35619869D-10  -0.22422950D-01

total values when epsilon =  0.10000000D-02
   scaled t-o-f        t-o-f          path length
  0.10726569D+03   0.35779982D-06   0.90201577D+02


twiss matrix for epsilon defined in terms of P sub tau:

on energy/momentum betatron matrix

matrix for map is :

  8.79899E-01  6.43546E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00
 -2.82743E-01 -9.31451E-01  0.00000E+00  0.00000E+00  0.00000E+00  -1.11022E-16
  0.00000E+00  0.00000E+00 -9.04734E-01  6.22828E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00 -2.82058E-01  8.36415E-01  0.00000E+00   0.00000E+00
  1.66533E-16  4.44089E-15  0.00000E+00  0.00000E+00  1.00000E+00   1.06047E+01
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   1.00000E+00


matrix for epsilon correction

matrix for map is :

 -7.11429E+00  1.25010E+00  0.00000E+00  0.00000E+00  0.00000E+00  -5.20514E-15
  1.62198E-01 -6.74652E+00  0.00000E+00  0.00000E+00  0.00000E+00   7.74944E-16
  0.00000E+00  0.00000E+00 -1.51190E+01 -5.11077E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  1.20823E-01 -1.64024E+01  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00
```

```
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00


matrix for epsilon**2 correction

matrix for map is :

 -3.12764E+01 -1.52265E+02  0.00000E+00  0.00000E+00  0.00000E+00  -7.47179E-15
  6.84220E+00  1.15447E+01  0.00000E+00  0.00000E+00  0.00000E+00   3.76082E-15
  0.00000E+00  0.00000E+00  8.75454E+01 -7.81241E+02  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  3.50921E+01 -1.29416E+02  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   0.00000E+00


value of betatron matrix when epsilon=  0.10000000D-02

matrix for map is :

  8.72754E-01  6.43655E+00  0.00000E+00  0.00000E+00  0.00000E+00  -5.21261E-18
 -2.82574E-01 -9.38186E-01  0.00000E+00  0.00000E+00  0.00000E+00  -1.10244E-16
  0.00000E+00  0.00000E+00 -9.19765E-01  6.22238E+00  0.00000E+00   0.00000E+00
  0.00000E+00  0.00000E+00 -2.81902E-01  8.19883E-01  0.00000E+00   0.00000E+00
  1.66533E-16  4.44089E-15  0.00000E+00  0.00000E+00  1.00000E+00   1.06047E+01
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   1.00000E+00

lengthy ray trace data

i = 1
rmat = -.388425170946838D-02
rf3 = 0.127366180805828D-05
rf4 = 0.497771285645497D-08
rf3^2/2 = -.193414162933386D-08
rf3^3/6 = -.101712468595459D-10
rf3f4 = -.561194977342628D-11

i = 2
rmat = 0.409136190557491D-03
rf3 = -.272133492428854D-06
rf4 = -.978203860541615D-09
rf3^2/2 = -.361182373079016D-09
rf3^3/6 = 0.214279953122663D-11
rf3f4 = -.127069182114673D-11

i = 3
rmat = 0.000000000000000D+00
rf3 = 0.000000000000000D+00
rf4 = 0.000000000000000D+00
rf3^2/2 = 0.000000000000000D+00
rf3^3/6 = 0.000000000000000D+00
rf3f4 = 0.000000000000000D+00

i = 4
```

```
rmat = 0.000000000000000D+00
rf3 = 0.000000000000000D+00
rf4 = 0.000000000000000D+00
rf3^2/2 = 0.000000000000000D+00
rf3^3/6 = 0.000000000000000D+00
rf3f4 = 0.000000000000000D+00

i = 5
rmat = 0.106041968397807D-01
rf3 = 0.742360623935188D-04
rf4 = 0.132932068635130D-06
rf3^2/2 = 0.219425058828633D-08
rf3^3/6 = -.449053718580645D-11
rf3f4 = 0.690563417159232D-11

i = 6
rmat = 0.100000000000000D-02
rf3 = 0.000000000000000D+00
rf4 = 0.000000000000000D+00
rf3^2/2 = 0.000000000000000D+00
rf3^3/6 = 0.000000000000000D+00
rf3f4 = 0.000000000000000D+00

initial conditions are: (dimensionless form)
-3.882975000000000E-003  4.088627100000000E-004
  0.000000000000000E+000  0.000000000000000E+000
  0.000000000000000E+000  1.000000000000000E-003

final conditions are: (dimensionless form)
-3.882975019872296E-003  4.088627185509358E-004
  0.000000000000000E+000  0.000000000000000E+000
  1.067856803090855E-002  1.000000000000000E-003

end of MARYLIE run
```

8.2 Twiss Analyze Static Map

Type Code: tasm

Required Parameters:

1. IOPT (controls meaning of the sixth parameter, EPSILON, in the parameter set IPSET).
   = 1 if EPSILON = P_τ.
   = 2 if EPSILON = δ. See section 4.1.2.
2. IPSET (may = 0, in which case the net effect is the same as using a pset with all zero entries).
3. IDATA
   = 0 to not compute analysis data.
   = 1 to compute tunes, tune expansions (chromaticities), squared betatron amplitudes, and anharmonicities.
   = 2 to compute Twiss parameter expansions, envelope coefficients, and envelopes.
   = 3 to compute eigenvector expansions.
   = 12 to compute items 1 and 2 above.
   = 13 to compute items 1 and 3 above.
   = 23 to compute items 2 and 3 above.
   = 123 to compute items 1, 2, and 3 above.
4. IPMAPS
   = 0 to not print maps.
   = 1 to print $\mathcal{A}_c$ and $\mathcal{N}$, the normalizing map with respect to the closed orbit and the normal form for the current transfer map.
   = 2 to print $\mathcal{B}$ and $\mathcal{A}_b$, the betatron factor for the current transfer map and the transforming map for the betatron factor.
   = 3 to do both items 1 and 2 above.
5. ISEND
   = 0 to print nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.

   Note: When ISEND = −1, −2, or −3, the result is the same as when ISEND = +1, +2, or +3, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.

6. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps $\mathcal{N}$ and the contents of all six buffers on file IFILE.

This command also requires additional parameters whose values are specified by the parameter set IPSET. Its contents are listed below.
Contents of IPSET:

1. X
2. P_X
3. Y
4. P_Y
5. 0
6. EPSILON

The quantities X, P_X, Y, P_Y specify the transverse coordinates of a phase-space point, and EPSILON specifies an energy or momentum deviation depending on the value of IOPT.

Example:

```
twissdat    tasm
  1 ,  1 ,  1 , 0 , 1 , 0
```

This specifies a command with the user given name *twissdat*. When invoked it computes tunes, chromaticities, squared betatron amplitudes, and anharmonicities. It does so using P_τ as the expansion parameter and the contents of pset 1 as phase-space coordinates. Results are to be written at the terminal.

Description:

The expansion for each tune is given in the form

$$T(\epsilon) = T^{(0)} + \epsilon T^{(1)} + \epsilon^2 T^{(2)} + \cdots . \tag{8.2.1}$$

Here the expansion parameter ϵ is given by the relations

$$\epsilon = P_\tau \text{ if IOPT} = 1, \tag{8.2.2}$$

$$\epsilon = \delta \text{ if IOPT} = 2. \tag{8.2.3}$$

The quantity $T^{(0)}$ is the fractional part of the tune for the design closed orbit, and lies in the range $0 < T^{(0)} < 1$ unless set otherwise by a command with the type code *ctr*. See section 8.28. Note that since the computation of tunes is based on the total transfer map, and not what went into it, the integer part of the tune must be determined by other means. See section 10.10. If the energy (momentum) of a particle differs slightly from the design value,

there is a unique off energy closed orbit for such a particle [assuming that the tunes $T^{(0)}$ do not have integer values]. See section 8.1. Consider a particle whose energy is slightly different from the design energy and whose initial conditions are near those for the corresponding off-energy closed orbit. Such a particle will make betatron oscillations about the off-energy closed orbit. The tunes of such particles are given by the expansion (8.2.1). The quantities $T^{(1)}$ and $T^{(2)}$ are called the first- and second-order *chromaticities*, respectively.

The above discussion refers to the case of infinitesimal betatron oscillation amplitudes about the (energy dependent) closed orbit. If the betatron amplitudes are finite, there are amplitude dependent corrections described by the *anharmonicities*. The map $\mathcal{M}$ can be written in the normal form factorization

$$\mathcal{M} = \mathcal{A}^{-1}\mathcal{N}\mathcal{A}. \tag{8.2.4}$$

Let $\boldsymbol{z}^{\text{in}}$ denote the coordinates of the phase-space point stored in the parameter set IPSET. Let $\boldsymbol{z}^{\text{tr}}$ be the transformed point given by the relation

$$\boldsymbol{z}^{\text{tr}} = \mathcal{A}^{-1}\boldsymbol{z}^{\text{in}}. \tag{8.2.5}$$

[When IOPT = 2, in which case $\epsilon = \delta$, P_τ is first computed using (4.1.9), and z_6^{in} is then set to $z_6^{in} = P_\tau$.] The *horizontal* and *vertical* squared betatron eigen amplitudes (about the closed orbit) are given the relations

$$ha2 = (z_1^{\text{tr}})^2 + (z_2^{\text{tr}})^2, \tag{8.2.6}$$

$$va2 = (z_3^{\text{tr}})^2 + (z_4^{\text{tr}})^2. \tag{8.2.7}$$

If anharmonicities are taken into account, the *finite* betatron *amplitude* fractional *horizontal* and *vertical* tunes are given by the relations

$$T_h^{fa}(\epsilon) = T_h^{(0)} + \epsilon T_h^{(1)} + \epsilon^2 T_h^{(2)} + (hh)(ha2) + (hv)(va2), \tag{8.2.8}$$

$$T_v^{fa}(\epsilon) = T_v^{(0)} + \epsilon T_v^{(1)} + \epsilon^2 T_v^{(2)} + (hv)(ha2) + (vv)(va2). \tag{8.2.9}$$

Here the quantities hh, vv, and hv are the anharmonicities.

The twiss parameters are computed as follows: Let $\mathcal{A}_b^{-1}$ be the inverse of the normalizing map for the betatron factor $\mathcal{B}$. (See the definitions below.) When this map acts on the quantity $(P_x^2 + X^2)$ the result is a polynomial of the form

$$I_x = \mathcal{A}_b^{-1}(P_x^2 + X^2) = \beta P_x^2 + 2\alpha X P_x + \gamma X^2 + \text{ additional terms} \tag{8.2.10}$$

where α, β, γ are the horizontal twiss parameters. Analogous results hold for the vertical plane. The additional terms describe chromatic corrections to the twiss parameters and possible effects of coupling between the horizontal and vertical planes. In fact, the quantities I_x and I_y, which MaryLie calls Twiss invariants, are generalizations of the usual Courant-Snyder invariants that take into account chromatic and coupling effects. That is, there are the relations

$$\mathcal{B} I_{x,y} = I_{x,y}. \tag{8.2.11}$$

When a command with type code *tasm* is invoked, twiss parameters are computed and made available for fitting, and I_x and I_y are placed in buffers 5 and 6. When IDATA = 2 (or 23 or 123) the twiss parameters and these invariants are written out.

More specifically, in the absence of coupling between the horizontal and vertical planes, I_x is of the form

$$I_x = (\beta_0 + \epsilon\beta_1 + \epsilon^2\beta_2)X^2 + 2(\alpha_0 + \epsilon\alpha_1 + \epsilon^2\alpha_2)XP_x + (\gamma_0 + \epsilon\gamma_1 + \epsilon^2\gamma_2)Y^2, \tag{8.2.12}$$

where, in the canonical coordinates employed,

$$\epsilon = P_\tau. \tag{8.2.13}$$

There is a similar expression for I_y. Thus, the Twiss parameters have expansions of the form

$$\beta(\epsilon) = \beta_0 + \epsilon\beta_1 + \epsilon\beta_2 + \cdots, \tag{8.2.14}$$

and *tasm* provides such expansions for either

$$\epsilon = P_\tau \text{ or } \epsilon = \delta. \tag{8.2.15}$$

These expansions are also evaluated for the specified value of ϵ. In addition, for fitting, writing, etc., the quantities α_0, β_0, γ_0 and α, β, γ evaluated for the specified value of ϵ are placed in the array *tp*. See Section 9.5.

Section 8.1 described the ϵ expansion of $R_{Tr}(\epsilon)$, the matrix part of the transfer map about the closed orbit. See (8.1.30). On the assumption that the eigenvalues of $R_{Tr}(\epsilon)$ lie on the unit circle and are distinct, there is a normalizing matrix $A_{Tr}(\epsilon)$ associated with $\mathcal{A}_b$ and a normal form matrix $N_{Tr}(\epsilon)$ associated with $\mathcal{N}_b$ such that

$$R_{Tr}(\epsilon)A_{Tr}(\epsilon) = A_{Tr}(\epsilon)N_{Tr}(\epsilon). \tag{8.2.16}$$

The normal form matrix consists of 2×2 blocks on the diagonal of the form (6.15.1) with $\alpha = 0$ and $\beta = \gamma = 1$. The phase advances $\phi(\epsilon)$ for these matrices have an ϵ expansion given by the relation

$$\phi(\epsilon) = 2\pi T(\epsilon) \tag{8.2.17}$$

and (8.2.1).

Consider betatron oscillations about the closed orbit, and suppose nonlinear effects for these oscillations are neglected. That is, only the effect of $\mathcal{B}$ is taken into account, and the effect of $\mathcal{C}$ is ignored. The map $\mathcal{N}_b$ produces motion on a torus with radii $ha = \sqrt{ha2}$ and $va = \sqrt{va2}$. The map $\mathcal{A}_b$ transforms this motion to the motion produced by $\mathcal{B}$. Let z^b denote phase-space coordinates describing deviations from (betatron oscillations about) the closed orbit. This is the motion produced by $\mathcal{B}$. Then, turn by turn, there are relations of the form

$$\begin{aligned} z_i^b(\chi_h, \chi_v) &= ha\{[A_{\mathrm{Tr}}(\epsilon)]_{i1}\cos\chi_h + [A_{\mathrm{Tr}}(\epsilon)]_{i2}\sin\chi_h\} \\ &+ va\{[A_{\mathrm{Tr}}(\epsilon)]_{i3}\cos\chi_v + [A_{\mathrm{Tr}}(\epsilon)]_{i4}\sin\chi_v\} \end{aligned} \tag{8.2.18}$$

where χ_h, χ_v are phase factors that vary from turn to turn.

Evidently a term of the form

$$[A_{\rm Tr}(\epsilon)]_{i1}\cos\chi_h + [A_{\rm Tr}(\epsilon)]_{i2}\sin\chi_h$$

may be viewed as the scalar product of a vector with entries $[A_{\rm Tr}(\epsilon)]_{i1}$, $[A_{\rm Tr}(\epsilon)]_{i2}$ and a vector with entries $\cos\chi_h$, $\sin\chi_h$. Therefore, by Schwarz's inequality, there is the result

$$|[A_{\rm Tr}(\epsilon)]_{i1}\cos\chi_h + [A_{\rm Tr}(\epsilon)]_{i2}\sin\chi_h| \le \sqrt{\{[A_{\rm Tr}(\epsilon)]_{i1}\}^2 + \{[A_{\rm Tr}(\epsilon)]_{i2}\}^2}. \tag{8.2.19}$$

Define what will be called *envelope coefficients* by the relations

$$xehc = \sqrt{\{[A_{\rm Tr}(\epsilon)]_{11}\}^2 + \{[A_{\rm Tr}(\epsilon)]_{12}\}^2}, \tag{8.2.20}$$

$$xevc = \sqrt{\{[A_{\rm Tr}(\epsilon)]_{13}\}^2 + \{[A_{\rm Tr}(\epsilon)]_{14}\}^2}, \tag{8.2.21}$$

$$pxehc = \sqrt{\{[A_{\rm Tr}(\epsilon)]_{21}\}^2 + \{[A_{\rm Tr}(\epsilon)]_{22}\}^2}, \tag{8.2.22}$$

$$pxevc = \sqrt{\{[A_{\rm Tr}(\epsilon)]_{23}\}^2 + \{[A_{\rm Tr}(\epsilon)]_{24}\}^2}, \text{ etc.} \tag{8.2.23}$$

Then there are the inequalities

$$|X^b| \le xe = xehc\ ha + xevc\ va, \tag{8.2.24}$$

$$|P_x^b| \le pxe = pxehc\ ha + pxevc\ va, \tag{8.2.25}$$

$$|Y^b| \le ye = yehc\ ha + yevc\ va, \tag{8.2.26}$$

$$|P_y^b| \le pye = pyehc\ ha + pyevc\ va. \tag{8.2.27}$$

With some thought it becomes evident that, if the tunes are not resonant (irrational and incommensurate) so that all phases in (8.2.18) are ergodic, the inequalities (8.2.24) through (8.2.27) can be saturated. Thus these relations provide bounds (envelopes) on the betatron motion that are met but never exceeded. For the quantities $xe, \cdots pye$ we will use the term *envelopes*. The envelope coefficients $xehc, \cdots pyevc$ and the envelopes $xe, \cdots pye$ are also printed out when IDATA = 2. Evidently the envelope coefficients can be viewed as generalizations of the β and γ functions that take into account chromatic and coupling effects. Indeed, in the absence of such effects there are the relations

$$xehc = \sqrt{\beta_0}, \tag{8.2.28}$$

$$pxehc = \sqrt{\gamma_0}, \text{ etc.}, \tag{8.2.29}$$

where β_0, γ_0 are the horizontal on energy/momentum twiss parameters, etc.

It can be shown that the colunns of $A_{Tr}(\epsilon)$ are composed of the real and imaginary parts of the eigenvectors of $R_{Tr}(\epsilon)$. Consequently, an expansion of the form

$$A_{Tr}(\epsilon) = A_{Tr}^{(0)} + \epsilon A_{Tr}^{(1)} + \epsilon^2 A_{Tr}^{(2)} + \cdots \tag{8.2.30}$$

provides an ϵ expansion of the eigenvectors of $R_{Tr}(\epsilon)$. It is this expansion that is provided when IDATA = 3.

Finally we remark that MaryLie takes into account any coupling between horizontal and vertical planes that might be present. Thus, what are referred to as tunes, chromaticities, betatron amplitudes, and anharmonicities are in fact eigentunes, eigenchromaticities, eigen betatron amplitudes, and eigenanharmonicities. The labels "horizontal" and "vertical" are assigned in such a way that the various quantities go over properly and continuously to their appropriately named quantities in the limit of vanishing coupling.

When a command with type code *tasm* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}_c$, the normalizing map with respect to the closed orbit.
2. Buffer 2 contains $\mathcal{B}$, the betatron factor of the current transfer map.
3. Buffer 3 contains $\mathcal{A}_b$, the normalizing map for the betatron factor.
4. Buffer 4 contains $\mathcal{A}$, the full normalizing map.
5. Buffer 5 contains the identity matrix in its matrix part, and the x quadratic betatron (twiss) invariant in its polynomial part.
6. Buffer 6 contains the identity matrix in its matrix part, and the y quadratic betatron (twiss) invariant in its polynomial part.

Terminology:

For a definition of the maps $\mathcal{T}$, $\mathcal{B}$, and $\mathcal{C}$, see section 8.1. The maps $\mathcal{N}$, $\mathcal{A}_c$, and $\mathcal{A}_b$ are described below:

- The map $\mathcal{M}$ can be written in the normal form factorization:

  $$\mathcal{M} = \mathcal{A}^{-1}\mathcal{N}\mathcal{A}.$$

  The type code *tasm* can be used to compute the normal form map $\mathcal{N}$ and to print it out and/or write it to an external file. It is not placed in a buffer by *tasm*, but is so available in *snor* and *sia*. The full normalizing map $\mathcal{A}$ is available from *tasm* in buffer 4. It is also available from *rasm*, *snor,* and *sia*. See sections 8.4, 8.8, and 8.10.

- The product $\mathcal{BC}$ (the transfer map about the closed orbit) can be written in the normal form factorization

  $$\mathcal{BC} = \mathcal{A}_c^{-1}\mathcal{N}\mathcal{A}_c.$$

  The map $\mathcal{A}_c$ is the normalizing map with respect to the closed orbit. Evidently one has the relation

  $$\mathcal{A} = \mathcal{A}_c\mathcal{T}.$$

- The betatron factor $\mathcal{B}$ can itself be written in a normal form factorization

  $$\mathcal{B} = \mathcal{A}_b^{-1}\mathcal{N}_b\mathcal{A}_b$$

The normal form factor $\mathcal{N}_b$ is similar to $\mathcal{N}$ except that it contains no anharmonicity terms. (Recall that $\mathcal{B}$ describes only infinitesimal oscillations about the closed orbit.) The map $\mathcal{A}_b$ is the normalizing map for the betatron factor. It is used internally by *tasm* to compute twiss parameters and eigenvector expansions. The map $\mathcal{N}_b$ is not directly available from *tasm*, but can be computed from $\mathcal{A}_b$ and $\mathcal{B}$, which are available in buffers.

Note: All quantities computed by *tasm* are available for fitting or optimizing or scanning, etc. except for the eigenvector expansion (2.26). However, if desired, it can be made so available by the subsequent use of a *cxp* command. See section 8.24.

The MaryLie run shown in the Exhibit below illustrates the use of a *tasm* command. Several types of data are displayed.

```
#comment
  Exhibit 8.2.
  This is a test of the type code tasm.  A command with type
  code tasm is first applied to a map produced with a twsm command
  to demonstrate consistency.  Then it is applied to the one-turn
  map for the small static storage ring of Exhibit 2.5.1.  Both
  tasm options are illustrated.  Note that the relations (2.24),(2.25),
etc.
  hold for the case of the simple map produced by the twsm command.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  arot     arot
    5.00000000000000
  twsx     twsm
    1.00000000000000       60.0000000000000       1.00000000000000
    2.00000000000000
  twsy     twsm
    2.00000000000000       72.0000000000000       3.00000000000000
    4.00000000000000
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
```

```
  hdq      quad
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hcs      sext
   0.500000000000000      0.000000000000000E+00
  vcs      sext
   0.500000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  tasmpt   tasm
    1.00000000000000       1.00000000000000       123.000000000000
   0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  tasmd    tasm
    2.00000000000000       1.00000000000000       123.000000000000
   0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  ptdata   ps1
   1.000000000000000E-02  0.000000000000000E+00  1.000000000000000E-02
   0.000000000000000E+00  0.000000000000000E+00  1.000000000000000E-03
  ddata    ps1
   1.000000000000000E-02  0.000000000000000E+00  1.000000000000000E-02
   0.000000000000000E+00  0.000000000000000E+00 -1.188970448993188E-03
  clear    iden
  zer      zer
   0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
   0.000000000000000E+00
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
#lumps
#loops
#labor
     1*fileout
     1*zer
     1*twsx
     1*twsy
     1*ptdata
     1*tasmpt
     1*clear
```

```
     1*ring
     1*tasmpt
     1*ddata
     1*tasmd
     1*fin

twiss analysis of static map
  P sub tau =   1.000000000000000E-003
  delta =  -1.188970448993188E-003

tunes, chromaticities, etc. for epsilon defined in terms of P sub tau:

horizontal tune =  0.166666666666667
first order horizontal chromaticity =  0.000000000000000E+000
second order horizontal chromaticity =  0.000000000000000E+000
horizontal tune when epsilon =  1.000000000000000E-003
  0.166666666666667

vertical tune =  0.200000000000000
first order vertical chromaticity =  0.000000000000000E+000
second order vertical chromaticity =  0.000000000000000E+000
vertical tune when epsilon =  1.000000000000000E-003
  0.200000000000000

tune separation when epsilon =  1.000000000000000E-003
-3.333333333333341E-002

anharmonicities
  hh=  0.000000000000000E+000
  hv=  0.000000000000000E+000
  vv=  0.000000000000000E+000

squared betatron amplitudes
  ha2=   9.999999999999998E-005
  va2=   2.499999999999999E-004
finite amplitude horizontal tune =   0.166666666666667
finite amplitude vertical tune =   0.200000000000000
finite amplitude tune separation =  -3.333333333333341E-002

twiss parameters, invariants, and envelopes

horizontal parameters

full twiss invariant written as a map

nonzero elements in generating polynomial are :

  f(  7)=f( 20 00 00 )=  1.0000000000000
  f(  8)=f( 11 00 00 )=  2.0000000000000
  f( 13)=f( 02 00 00 )=  2.0000000000000

diagonal terms (alpha,beta,gamma)
```

```
on energy terms
   1.00000000000000        2.00000000000000        1.00000000000000

linear in epsilon expansion coefficients
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

quadratic in epsilon expansion coefficients
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

diagonal terms when epsilon =   1.000000000000000E-003
   1.00000000000000        2.00000000000000        1.00000000000000

vertical parameters

full twiss invariant written as a map

nonzero elements in generating polynomial are :

  f( 18)=f( 00 20 00 )=  2.5000000000000
  f( 19)=f( 00 11 00 )=  6.0000000000000
  f( 22)=f( 00 02 00 )=  4.0000000000000

diagonal terms (alpha,beta,gamma)

on energy terms
   3.00000000000000        4.00000000000000        2.50000000000000

linear in epsilon expansion coefficients
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

quadratic in epsilon expansion coefficients
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

diagonal terms when epsilon =   1.000000000000000E-003
   3.00000000000000        4.00000000000000        2.50000000000000

envelopes when epsilon =  1.000000000000000E-003

horizontal envelope coefficients (xehc,xevc;pxehc,pxevc)
   1.41421356237310       0.000000000000000E+000
   1.00000000000000       0.000000000000000E+000

vertical envelope coefficients (yehc,yevc;pyehc,pyevc)
  0.000000000000000E+000   2.00000000000000
  0.000000000000000E+000   1.58113883008419

betatron amplitudes
  ha=   9.999999999999998E-003
  va=   1.581138830084189E-002

finite amplitude envelopes (xe,pxe;ye,pye)
  1.414213562373095E-002  9.999999999999997E-003
  3.162277660168377E-002  2.499999999999999E-002
```

```
eigenvector expansion when epsilon =  1.000000000000000E-003

on energy matrix of eigenvectors

matrix for map is :

  1.41421E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -7.07107E-01  7.07107E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -1.50000E+00  5.00000E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00


epsilon correction

matrix for map is :

  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


epsilon**2 correction

matrix for map is :

  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

matrix of eigenvectors when epsilon=  1.000000000000000E-003

matrix for map is :

  1.41421E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -7.07107E-01  7.07107E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -1.50000E+00  5.00000E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

twiss analysis of static map
  P sub tau =   1.000000000000000E-003
  delta =  -1.188970448993188E-003
```

```
tunes, chromaticities, etc. for epsilon defined in terms of P sub tau:

horizontal tune =  0.254102811747596
first order horizontal chromaticity =   1.10337520026811
second order horizontal chromaticity =   1.66933106836440
horizontal tune when epsilon =  1.000000000000000E-003
  0.255207856278933

vertical tune =  0.255437705851761
first order vertical chromaticity =   2.50986502239160
second order vertical chromaticity =   4.01031685515868
vertical tune when epsilon =  1.000000000000000E-003
  0.257951581191008

tune separation when epsilon =  1.000000000000000E-003
-2.743724912075307E-003

anharmonicities
  hh=  7.695357766309278E-002
  hv=  0.267563462613096
  vv=  0.180267510644177

squared betatron amplitudes
  ha2=   4.547438572252656E-005
  va2=   2.851973842113647E-005
finite amplitude horizontal tune =   0.255218986535571
finite amplitude vertical tune =   0.257968889657362
finite amplitude tune separation =  -2.749903121790687E-003

twiss parameters, invariants, and envelopes

horizontal parameters

full twiss invariant written as a map

nonzero elements in generating polynomial are :

  f(  7)=f( 20 00 00 )= 0.28283741725472
  f(  8)=f( 11 00 00 )=  1.8119521227272
  f( 13)=f( 02 00 00 )=  6.4375945779623
  f( 33)=f( 20 00 01 )=-0.11169297025730
  f( 38)=f( 11 00 01 )=-4.39995277499051E-02
  f( 53)=f( 02 00 01 )=  2.4012789660730
  f(104)=f( 20 00 02 )= 8.98597094751910E-03
  f(119)=f( 11 00 02 )=  1.1902837331380
  f(154)=f( 02 00 02 )=  4.5581327706390

diagonal terms (alpha,beta,gamma)

on energy terms
  0.905976061363577        6.43759457796235       0.282837417254720

linear in epsilon expansion coefficients
```

```
-2.199976387495256E-002   2.40127896607300      -0.111692970257297

quadratic in epsilon expansion coefficients
  0.595141866569002        4.55813277063904       8.985970947519101E-003

diagonal terms when epsilon =   1.000000000000000E-003
  0.905954656741569        6.44000041506119       0.282725733270434

vertical parameters

full twiss invariant written as a map

nonzero elements in generating polynomial are :

  f( 18)=f( 00 20 00 )= 0.28222227408050
  f( 19)=f( 00 11 00 )= -1.7421650477668
  f( 22)=f( 00 02 00 )=  6.2319133709247
  f( 67)=f( 00 20 01 )= 3.12262681415022E-02
  f( 70)=f( 00 11 01 )= 0.34510852165303
  f( 76)=f( 00 02 01 )= -1.7547070004441
  f(184)=f( 00 20 02 )= 0.24044077217539
  f(190)=f( 00 11 02 )=-0.85695095068877
  f(200)=f( 00 02 02 )= -2.3646721152833

diagonal terms (alpha,beta,gamma)

on energy terms
-0.871082523883395        6.23191337092474       0.282222274080502

linear in epsilon expansion coefficients
  0.172554260826516       -1.75470700044406       3.122626814150218E-002

quadratic in epsilon expansion coefficients
-0.428475475344385       -2.36467211528335       0.240440772175391

diagonal terms when epsilon =   1.000000000000000E-003
-0.870910398098044        6.23015629925218       0.282253740789415

envelopes when epsilon =  1.000000000000000E-003

horizontal envelope coefficients (xehc,xevc;pxehc,pxevc)
   2.53771559003073       0.000000000000000E+000
  0.531719600258752       0.000000000000000E+000

vertical envelope coefficients (yehc,yevc;pyehc,pyevc)
  0.000000000000000E+000   2.49602810476500
  0.000000000000000E+000  0.531275578974176

betatron amplitudes
  ha=   6.743469857760659E-003
  va=   5.340387478557756E-003

finite amplitude envelopes (xe,pxe;ye,pye)
```

```
  1.711300858894151E-002  3.585635097125439E-003
  1.332975723681525E-002  2.837217449617210E-003

eigenvector expansion when epsilon =  1.000000000000000E-003

on energy matrix of eigenvectors

matrix for map is :

  2.53724E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00


epsilon correction

matrix for map is :

  4.73207E-01  4.56623E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  4.33537E-03 -1.37768E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -3.51450E-01  9.07617E-02  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -3.45609E-02  6.90817E-02  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


epsilon**2 correction

matrix for map is :

  8.13029E-01  5.87798E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -1.05289E-01 -1.08092E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -5.00009E-01  9.47381E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  8.21307E-02  2.21126E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

matrix of eigenvectors when epsilon=  1.000000000000000E-003

matrix for map is :

  2.53772E+00  4.56682E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -3.57067E-01  3.93991E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.49603E+00  9.17091E-05  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  3.48904E-01  4.00649E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

twiss analysis of static map
  P sub tau =   1.000000000000000E-003
```

```
  delta =  -1.188970448993188E-003

tunes, chromaticities, etc. for epsilon defined in terms of momentum
deviation:

horizontal tune =  0.254102811747596
first order horizontal chromaticity = -0.928170393454570
second order horizontal chromaticity =   1.04559371148684
horizontal tune when epsilon = -1.188970448993188E-003
  0.255207857021356

vertical tune =  0.255437705851761
first order vertical chromaticity =  -2.11132387676014
second order vertical chromaticity =   2.52919932812798
vertical tune when epsilon = -1.188970448993188E-003
  0.257951582953956

tune separation when epsilon = -1.188970448993188E-003
-2.743725932599583E-003

anharmonicities
  hh=  7.695357766309278E-002
  hv=  0.267563462613096
  vv=  0.180267510644177

squared betatron amplitudes
  ha2=   4.547438572252656E-005
  va2=   2.851973842113647E-005
finite amplitude horizontal tune =   0.255218987277994
finite amplitude vertical tune =   0.257968891420309
finite amplitude tune separation =  -2.749904142314963E-003

twiss parameters, invariants, and envelopes

horizontal parameters

full twiss invariant written as a map

nonzero elements in generating polynomial are :

  f(  7)=f( 20 00 00 )= 0.28283741725472
  f(  8)=f( 11 00 00 )=  1.8119521227272
  f( 13)=f( 02 00 00 )=  6.4375945779623
  f( 33)=f( 20 00 01 )=-0.11169297025730
  f( 38)=f( 11 00 01 )=-4.39995277499051E-02
  f( 53)=f( 02 00 01 )=  2.4012789660730
  f(104)=f( 20 00 02 )= 8.98597094751910E-03
  f(119)=f( 11 00 02 )=  1.1902837331380
  f(154)=f( 02 00 02 )=  4.5581327706390

diagonal terms (alpha,beta,gamma)

on momentum terms
```

```
  0.905976061363577        6.43759457796235       0.282837417254720

linear in epsilon expansion coefficients
  1.850642418531839E-002  -2.01998018642486       9.395725780734790E-002

quadratic in epsilon expansion coefficients
  0.423848225123026        2.93020563110399       2.009371416766704E-002

diagonal terms when epsilon =  -1.188970448993188E-003
  0.905954656945457        6.44000041699888       0.282725733257212

vertical parameters

full twiss invariant written as a map

nonzero elements in generating polynomial are :

  f( 18)=f( 00 20 00 )= 0.28222227408050
  f( 19)=f( 00 11 00 )= -1.7421650477668
  f( 22)=f( 00 02 00 )=  6.2319133709247
  f( 67)=f( 00 20 01 )= 3.12262681415022E-02
  f( 70)=f( 00 11 01 )= 0.34510852165303
  f( 76)=f( 00 02 01 )= -1.7547070004441
  f(184)=f( 00 20 02 )= 0.24044077217539
  f(190)=f( 00 11 02 )=-0.85695095068877
  f(200)=f( 00 02 02 )= -2.3646721152833

diagonal terms (alpha,beta,gamma)

on momentum terms
-0.871082523883395        6.23191337092474       0.282222274080502

linear in epsilon expansion coefficients
-0.145154391837601        1.47607730045400      -2.626785302041698E-002

quadratic in epsilon expansion coefficients
-0.324423089619911       -1.45754646723180       0.166304273397744

diagonal terms when epsilon =  -1.188970448993188E-003
-0.870910398221896        6.23015629817245       0.282253740877659

envelopes when epsilon = -1.188970448993188E-003

horizontal envelope coefficients (xehc,xevc;pxehc,pxevc)
   2.53771559038294       0.000000000000000E+000
  0.531719600240117       0.000000000000000E+000

vertical envelope coefficients (yehc,yevc;pyehc,pyevc)
  0.000000000000000E+000   2.49602810453957
  0.000000000000000E+000  0.531275579055280

betatron amplitudes
  ha=   6.743469857760659E-003
```

```
  va=   5.340387478557756E-003

finite amplitude envelopes (xe,pxe;ye,pye)
  1.711300859131663E-002  3.585635096999775E-003
  1.332975723561137E-002  2.837217450050337E-003

eigenvector expansion when epsilon = -1.188970448993188E-003

on momentum matrix of eigenvectors

matrix for map is :

  2.53724E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00


epsilon correction

matrix for map is :

 -3.98066E-01 -3.84116E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -3.64696E-03  1.15892E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.95644E-01 -7.63497E-02  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.90730E-02 -5.81122E-02  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


epsilon**2 correction

matrix for map is :

  5.17137E-01 -1.45565E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -7.50390E-02 -5.95479E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -3.10606E-01  6.59239E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  6.23685E-02  1.47981E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

matrix of eigenvectors when epsilon= -1.188970448993188E-003

matrix for map is :

  2.53772E+00  4.56682E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -3.57067E-01  3.93991E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  2.49603E+00  9.17094E-05  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  3.48904E-01  4.00649E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
end of MARYLIE run
```

8.3 Twiss Analyze Dynamic Map

Type Code: tadm

Required Parameters:

1. IPSET (may = 0, in which case the net effect is the same as using a pset with all zero entries).
2. IDATA
   = 0 to not compute analysis data.
   = 1 to compute tunes, squared betatron amplitudes, and anharmonicities.
   = 2 to compute Twiss parameters, envelope coefficients, and envelopes.
   = 3 to compute eigenvectors.
   = 12 to compute items 1 and 2 above.
   = 13 to compute items 1 and 3 above.
   = 23 to compute items 2 and 3 above.
   = 123 to compute items 1, 2, and 3 above.
3. IPMAPS
   = 0 to not print maps.
   = 1 to print the normalizing map $\mathcal{A}$.
   = 2 to print the normal form map $\mathcal{N}$.
   = 3 to print normalizing and normal form maps.
4. ISEND
   = 0 to print nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
   Note: When ISEND = -1, -2, or -3, the result is the same as when ISEND = $+1$, $+2$, or $+3$, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
5. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps on file IFILE.

This command also requires additional parameters whose values are specified by the parameter set IPSET. Its contents are listed below.
Contents of IPSET:

1. X
2. P_X

3. Y
4. P_Y
5. τ
6. P_τ

These quantities specify the coordinates of a phase-space point.

Example:

```
syntune    tadm
 1 , 1 , 0 , 1 , 0
```

This specifies a command with the user given name *syntune*. When invoked it computes tunes, squared betatron amplitudes, and anharmonicities. Results are written at the terminal.

Description:

The map $\mathcal{M}$ can be written in the normal form factorization

$$\mathcal{M} = \mathcal{A}^{-1}\mathcal{N}\mathcal{A}. \tag{8.3.1}$$

Let $\boldsymbol{z}^{\rm in}$ denote the coordinates of the phase-space point stored in the parameter set IPSET. Let $\boldsymbol{z}^{\rm tr}$ be the transformed point given by the relation

$$\boldsymbol{z}^{\rm tr} = \mathcal{A}^{-1}\boldsymbol{z}^{\rm in}. \tag{8.3.2}$$

The *horizontal*, *vertical*, and *temporal* squared eigen betatron amplitudes are given by the relations

$$ha2 = (z_1^{\rm tr})^2 + (z_2^{\rm tr})^2, \tag{8.3.3}$$

$$va2 = (z_3^{\rm tr})^2 + (z_4^{\rm tr})^2, \tag{8.3.4}$$

$$ta2 = (z_5^{\rm tr})^2 + (z_6^{\rm tr})^2. \tag{8.3.5}$$

If anharmonicities are taken into account, the *finite* betatron *amplitude* fractional *horizontal*, *vertical*, and *temporal* (synchrotron) tunes are given by the relations

$$T_h^{fa} = T_h + (hh)(ha2) + (hv)(va2) + (ht)(ta2), \tag{8.3.6}$$

$$T_v^{fa} = T_v + (hv)(ha2) + (vv)(va2) + (vt)(ta2), \tag{8.3.7}$$

$$T_t^{fa} = T_t + (ht)(ha2) + (vt)(va2) + (tt)(ta2). \tag{8.3.8}$$

Here the quantities hh, vv, tt, hv, ht, and vt are the anharmonicities.

The Twiss parameters are computed as follows: Let $\mathcal{A}_2^{-1}$ be the inverse of the linear part of the normalizing map $\mathcal{A}$. When this map acts on the quantity $(P_x^2 + X^2)$ the result is a quadratic polynomial of the form

$$I_x = \mathcal{A}_2^{-1}(P_x^2 + X^2) = \beta P_x^2 + 2\alpha X P_x + \gamma X^2 + \text{ additional terms} \tag{8.3.9}$$

where α, β, γ are the horizontal Twiss parameters. Analogous results hold for the vertical and temporal planes. The additional terms describe possible effects of coupling between planes. In fact, the quantities I_x, I_y, and I_τ, which MaryLie calls Twiss invariants, are generalizations of the usual Courant-Snyder invariants that take into account coupling effects. That is, these are the relations

$$\mathcal{M}_2 I_{x,y,\tau} = I_{x,y,\tau} \tag{8.3.10}$$

where $\mathcal{M}_2$ is the linear part of $\mathcal{M}$.

When a command with type code *tadm* is invoked, Twiss parameters are computed and made available for fitting, and I_x, I_y, and I_τ are placed in buffers 4 through 6. When IDATA = 2 (or 23 or 123) the Twiss parameters and these invariants are written out.

Suppose that nonlinear effects are neglected so that only the effects of $\mathcal{M}_2$ are taken into account. This map is brought to normal form by the normalizing map $\mathcal{A}_2$. Let A be the matrix associated with $\mathcal{A}_2$. In this case linear betatron oscillations about the design orbit are described by relations of the form

$$\begin{aligned} z_i(\chi_h, \chi_v, \chi_\tau) &= ha[A_{i1}\cos\chi_h + A_{i2}\sin\chi_h] \\ &+ va[A_{i3}\cos\chi_v + A_{i4}\sin\chi_v] \\ &+ ta[A_{i5}\cos\chi_\tau + A_{i6}\sin\chi_\tau] \end{aligned} \tag{8.3.11}$$

where χ_h, χ_v, χ_τ are phase factors that vary from turn to turn and ha, va, and ta are the horizontal, vertical, and temporal betatron eigen amplitudes. Correspondingly there are envelope coefficients such that

$$|X| \le xe = xehc\ ha + xevc\ va + xetc\ ta, \tag{8.3.12}$$

$$|P_x| \le xe = pxehc\ ha + pxevc\ va + pxetc\ ta, \tag{8.3.13}$$

$$|Y| \le ye = yehc\ ha + yevc\ va + yetc\ ta, \tag{8.3.14}$$

$$|P_y| \le pye = pyehc\ ha + pyevc\ va + pyetc\ ta, \tag{8.3.15}$$

$$|\tau| \le te = tehc\ ha + tevc\ va + tetc\ ta, \tag{8.3.16}$$

$$|P_\tau| \le pte = ptehc\ ha + ptevc\ va + ptetc\ ta. \tag{8.3.17}$$

These relations provide bounds on the betatron motion about the closed orbit that are met but never exceeded. (See section 8.2 for an analogous discussion of the static case.) The envelope coefficients $xehc, \cdots ptetc$ and the envelopes $xe, \cdots pte$ are also printed out when IDATA = 2. Evidently the envelope coefficients are generalizations of the β and γ functions that take into account coupling effects. Indeed, in the absence of such effects there are the relations (2.24), (2.25), etc.

Finally, let R be the matrix associated with $\mathcal{M}_2$. It can be shown that the columns of A are composed of the real and imaginary parts of the eigenvectors of R. The matrix A of eigenvectors is printed out when IDATA = 3.

We remark that MaryLie takes into account any coupling between horizontal, vertical, and temporal planes that might be present. Thus, what are referred to as tunes, betatron amplitudes, and anharmonicities are in fact eigentunes, eigen betatron amplitudes, and eigen-

anharmonicities. The labels "horizontal", "vertical", and "temporal" are assigned in such a way that the various quantities go over properly and continuously to their appropriately named quantities in the limit of vanishing coupling. Transverse tunes are taken to lie in the range $(0, 1)$. Finally, since the temporal tune is usually near zero and can be negative, it is taken to lie in the range $-(1/2) < T_t < (1/2)$. These ranges can be changed by using a command with type code *ctr*. See section 8.28.

When a command with type code *tadm* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}_2$, the linear part of the normalizing map.
2. Buffer 2 contains the full normalizing map $\mathcal{A}$.
3. Buffer 3 contains $\mathcal{N}$.
4. Buffer 4 contains the identity matrix in its matrix part, and the x quadratic synchro-betatron (twiss) invariant in its array part.
5. Buffer 5 contains the identity matrix in its matrix part, and the y quadratic synchro-betatron (twiss) invariant in its array part.
6. Buffer 6 contains the identity matrix in its matrix part, and the τ quadratic synchro-betatron (twiss) invariant in its array part.

The MaryLie run shown in the Exhibit below illustrates the use of a *tadm* command. Several types of data are displayed.

```
#comment
 Exhibit 8.3.
 This is a test of the type code tadm.  A command with type
 code tadm is first applied to a map produced with a twsm command
 to demonstrate consistency.  Then it is applied to the one-turn
 map for the small dynamic storage ring of Exhibit 2.6.1.  Note that the
 relations (2.24),(2.25), etc. hold for the case of the simple map
 produced by the twsm command.
#beam
  4.78740000023600
  2807.64374771200
  1.00000000000000
  1.00000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
```

```
   90.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.6094450000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.0110200000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       8.91000000000000
 vcs      sext
  0.200000000000000      -31.2000000000000
 cvty     srfc
  -350000.000000000       502000000.000000
 tadm     tadm
   1.00000000000000       123.000000000000      0.000000000000000E+00
   1.00000000000000      0.000000000000000E+00
 ps1      ps1
  1.000000000000000E-02  0.000000000000000E+00  1.000000000000000E-02
  0.000000000000000E+00  0.000000000000000E+00  5.000000000000000E-03
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 iden     iden
 twsx     twsm
   1.00000000000000       20.0000000000000       1.00000000000000
   2.00000000000000
 twsy     twsm
   2.00000000000000       30.0000000000000       3.00000000000000
   4.00000000000000
 twst     twsm
   3.00000000000000       72.0000000000000       5.00000000000000
   6.00000000000000
 zer      zer
  0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
  0.000000000000000E+00
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 half
     1*arc         1*cvty        1*arc
 statring
     4*arc
 dynring
     2*half
 test
     1*twsx        1*twsy        1*twst
#lumps
```

```
#loops
#labor
    1*fileout
    1*zer
    1*test
    1*ps1
    1*tadm
    1*iden
    1*dynring
    1*tadm
    1*fin

twiss analysis of dynamic map

horizontal tune =  5.555555555555557E-002
vertical tune =  8.333333333333336E-002
temporal tune =  0.200000000000000

anharmonicities
 hh=  0.000000000000000E+000
 vv=  0.000000000000000E+000
 tt=  0.000000000000000E+000
 hv=  0.000000000000000E+000
 ht=  0.000000000000000E+000
 vt=  0.000000000000000E+000

squared betatron amplitudes and finite amplitude tunes
 ha2=   9.999999999999996E-005
 va2=   2.500000000000003E-004
 ta2=   1.500000000000000E-004
 finite amplitude horizontal tune =   5.555555555555557E-002
 finite amplitude vertical tune =   8.333333333333336E-002
 finite amplitude temporal tune =   0.200000000000000

horizontal twiss parameters

diagonal terms (alpha,beta,gamma)
 0.999999999999999        2.00000000000000        1.00000000000000

full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )=  1.0000000000000
 f(  8)=f( 11 00 00 )=  2.0000000000000
 f( 13)=f( 02 00 00 )=  2.0000000000000

vertical twiss parameters

diagonal terms (alpha,beta,gamma)
  3.00000000000000        4.00000000000000        2.50000000000000

full twiss invariant written as a map
```

```
nonzero elements in generating polynomial are :

 f( 18)=f( 00 20 00 )=  2.5000000000000
 f( 19)=f( 00 11 00 )=  6.0000000000000
 f( 22)=f( 00 02 00 )=  4.0000000000000

temporal twiss parameters

diagonal terms (alpha,beta,gamma)
  5.00000000000000        6.00000000000000        4.33333333333333

full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f( 25)=f( 00 00 20 )=  4.3333333333333
 f( 26)=f( 00 00 11 )=  10.000000000000
 f( 27)=f( 00 00 02 )=  6.0000000000000

horizontal envelope coefficients (xehc,xevc,xetc;pxehc,pxevc,pxetc)
  1.41421356237309       0.000000000000000E+000  0.000000000000000E+000
  1.00000000000000       0.000000000000000E+000  0.000000000000000E+000

vertical envelope coefficients (yehc,yevc,yetc;pyehc,pyevc,pyetc)
 0.000000000000000E+000   2.00000000000000       0.000000000000000E+000
 0.000000000000000E+000   1.58113883008419       0.000000000000000E+000

temporal envelope coefficients (tehc,tevc,tetc;ptehc,ptevc,ptetc)
 0.000000000000000E+000  0.000000000000000E+000   2.44948974278318
 0.000000000000000E+000  0.000000000000000E+000   2.08166599946613

betatron amplitudes
 ha=   9.999999999999998E-003
 va=   1.581138830084191E-002
 ta=   1.224744871391589E-002

finite amplitude envelopes (xe,pxe;ye,pye;te,pte)
 1.414213562373095E-002  9.999999999999997E-003
 3.162277660168382E-002  2.500000000000003E-002
 3.000000000000000E-002  2.549509756796391E-002

matrix of eigenvectors

matrix for map is :

 1.41421E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00
-7.07107E-01  7.07107E-01  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  2.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00 -1.50000E+00  5.00000E-01  0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  2.44949E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -2.04124E+00 4.08248E-01
```

```
twiss analysis of dynamic map

horizontal tune =  0.176613318494107
vertical tune =  0.118740859382932
temporal tune = -1.731496642191732E-002

anharmonicities
 hh=   26.3223407503989
 vv=   68.6690650861266
 tt=   2.56408121701950
 hv=  -161.040347792620
 ht=  0.826493176425386
 vt=  0.594623320211105

squared betatron amplitudes and finite amplitude tunes
 ha2=   2.826871277148521E-005
 va2=   7.225033734487645E-005
 ta2=   5.362590351310147E-004
 finite amplitude horizontal tune =   0.166165412163433
 finite amplitude vertical tune =   0.119468691292155
 finite amplitude temporal tune =  -1.587362906879054E-002

horizontal twiss parameters

diagonal terms (alpha,beta,gamma)
 4.469108299671297E-016   8.48042374504464       0.117484536588254

full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.11748453658825
 f( 12)=f( 10 00 01 )= 0.42577475913485
 f( 13)=f( 02 00 00 )=  8.4804237450446
 f( 16)=f( 01 00 10 )=-1.72765145618715E-02
 f( 25)=f( 00 00 20 )= 8.79902833808772E-06
 f( 27)=f( 00 00 02 )= 0.38576171550066

vertical twiss parameters

diagonal terms (alpha,beta,gamma)
 6.498022550076755E-016   2.08033711122333       0.480691323826819

full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f( 18)=f( 00 20 00 )= 0.48069132382682
 f( 22)=f( 00 02 00 )=  2.0803371112233

temporal twiss parameters

diagonal terms (alpha,beta,gamma)
```

```
 7.552500892933016E-015   21.0746881490542       4.727560601191217E-002

full twiss invariant written as a map

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 2.18664519385818E-05
 f( 12)=f( 10 00 01 )=-4.29338400580191E-02
 f( 13)=f( 02 00 00 )= 0.15522995115863
 f( 16)=f( 01 00 10 )= 0.17133114150351
 f( 25)=f( 00 00 20 )= 4.72756060119122E-02
 f( 27)=f( 00 00 02 )=  21.074688149054

horizontal envelope coefficients (xehc,xevc,xetc;pxehc,pxevc,pxetc)
  2.91211671212619       5.380753593289051E-015  0.393992323730586
 0.342760173573673       4.499240444867006E-016  4.676157817972119E-003

vertical envelope coefficients (yehc,yevc,yetc;pyehc,pyevc,pyetc)
 1.137204921262785E-018   1.44233737773911       6.875296588310090E-016
 0.000000000000000E+000  0.693319063510314       0.000000000000000E+000

temporal envelope coefficients (tehc,tevc,tetc;ptehc,ptevc,ptetc)
 0.621097186840077       1.953904227786831E-015   4.59071760719979
 2.966315616735299E-003  5.925370506000332E-016  0.217429542638327

betatron amplitudes
 ha=   5.316832964414550E-003
 va=   8.500019843793099E-003
 ta=   2.315726743661727E-002

finite amplitude envelopes (xe,pxe;ye,pye;te,pte)
 2.460702373985856E-002  1.930685626911570E-003
 1.225989633222693E-002  5.893225797917717E-003
 0.109610745353010       5.050845472151043E-003

matrix of eigenvectors

matrix for map is :

 2.91212E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -3.93992E-01
 0.00000E+00  3.42760E-01  0.00000E+00  0.00000E+00  4.67616E-03  0.00000E+00
 0.00000E+00  0.00000E+00  1.44234E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  6.93319E-01  0.00000E+00  0.00000E+00
 0.00000E+00 -6.21097E-01  0.00000E+00  0.00000E+00  4.59072E+00  0.00000E+00
 2.96632E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  2.17430E-01

end of MARYLIE run
```

8.4 Resonance Analyze Static Map

Type Code: rasm

Required Parameters:

1. IOPT
   = 0 to retain third order terms when computing fourth order terms.
   = 1 to remove third order terms when computing fourth order terms.
   Note: When IOPT = −1, the result is the same as when IOPT = +1 except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
2. ITHIRD
   = 0 to not compute analysis data.
   = 1 to write analysis data for cubic terms at the terminal.
   = 2 to write analysis data for cubic terms on file 12.
   = 3 to write analysis data at the terminal and on file 12.
3. IFOURTH
   = 0 to not compute analysis data.
   = 1 to write analysis data for quartic terms at the terminal.
   = 2 to write analysis data for quartic terms on file 12.
   = 3 to write analysis data at the terminal and on file 12.
4. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps on file IFILE.

Example:

```
statres    rasm
 1 , 1 , 1 , 0
```

This specifies a command with the user given name *statres*. When invoked it removes third order "nonresonant" terms and analyzes the remaining third order and all the fourth order terms.

Description:

Suppose $\mathcal{M}$ is the map to be analyzed and IOPT = 0. Then this command finds a normalizing map

$$\mathcal{A} = \mathcal{A}_2$$

such that the purified map $\mathcal{N}$ has the form

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1} = \mathcal{N}_2(\exp : g_3 :)(\exp : g_4 :).$$

Here $\mathcal{N}_2$ is a linear map in static normal form, and g_3 and g_4 are displayed in a static resonance basis. Thus, the components of g_3 and g_4 in this basis are a measure of the strengths of third order and fourth order resonance driving terms.

In the case that IOPT = 1, this command finds a normalizing map

$$\mathcal{A} = \mathcal{A}_2\mathcal{A}_3$$

such that the purified map $\mathcal{N}$ has the form

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1} = \mathcal{N}_2\mathcal{N}_3 \exp(: h_4 :).$$

Here $\mathcal{N}_2$ is the same linear map in static normal form and $\mathcal{N}_3$ has third order generators free of "nonresonant" terms. The $\mathcal{N}_3$ generator and h_4 are displayed in a static resonance basis.

When a command with type code *rasm* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}$, the normalizing map.
2. Buffer 2 contains the purified map in the cartesian basis.
3. Buffer 3 contains the purified map in the static resonance basis.
4. Buffer 4 is empty.
5. Buffer 5 is empty.
6. Buffer 6 is empty.

The MaryLie run shown in the Exhibit below illustrates the use of a *rasm* command with both options.

```
#comment
  Exhibit 8.4.
  This MaryLie run illustrates the use of the type code rasm with
  both options for the small static storage ring of Exhibit 2.5.3.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
```

```
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       3.13000000000000       1.00000000000000
    1.00000000000000
  hdq      quad
   0.500000000000000      -1.92000000000000       1.00000000000000
    1.00000000000000
  hcs      sext
   0.500000000000000       2.65000000000000
  vcs      sext
   0.500000000000000      -5.01000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  raysin   rt
    13.0000000000000       14.0000000000000      -1.00000000000000
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  track    rt
    13.0000000000000       14.0000000000000       5.00000000000000
    500.000000000000       1.00000000000000      0.000000000000000E+00
  chrom    tasm
    2.00000000000000      1.000000000000000E-03   1.00000000000000
   0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  rasm0    rasm
   0.000000000000000E+00   1.00000000000000       1.00000000000000
   0.000000000000000E+00
  rasm1    rasm
    1.00000000000000       1.00000000000000       1.00000000000000
   0.000000000000000E+00
  zer      zer
   0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
   0.000000000000000E+00
  iden     iden
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
#lumps
#loops
```

```
#labor
     1*fileout
     1*zer
     1*half
     1*rasm0
     1*rasm1
     1*fin

resonance analysis of static map

requested resonance driving terms written as a map

nonzero elements in generating polynomial in
the static resonance basis are :

f( 28)=f( R11001 )=-1.89854336896950E-03
f( 29)=f( R00111 )=-3.32599139784473E-03
f( 30)=f( R00003 )= -10.672804169663
f( 35)=f( R20001 )=-0.50816385804132
f( 36)=f( I20001 )=  4.5403556514585
f( 37)=f( R00201 )=  3.0179757220971
f( 38)=f( I00201 )= -4.9100782402997
f( 43)=f( R21000 )= -1.4320242663414
f( 44)=f( I21000 )= -1.0310688540970
f( 47)=f( R10110 )=  1.0469812603412
f( 48)=f( I10110 )= -3.3217267669329
f( 51)=f( R30000 )= 0.80065961817555
f( 52)=f( I30000 )=-0.25915966558953
f( 57)=f( R10200 )= -2.1145185575516
f( 58)=f( I10200 )= -1.5130583452011
f( 61)=f( R01200 )=  1.7109740059006
f( 62)=f( I01200 )= 0.23837483457704
f( 84)=f( R11002 )= -61.269826367022
f( 85)=f( R00112 )= -25.102617348091
f( 86)=f( R00004 )= -37.099234810374
f( 87)=f( R22000 )= -7.9191307679030
f( 88)=f( R00220 )= -3.9867993263574
f( 89)=f( R11110 )= -7.0641577051297
f( 94)=f( R20002 )=  26.584896743788
f( 95)=f( I20002 )=  5.0352314282145
f( 96)=f( R00202 )=  9.2911532199116
f( 97)=f( I00202 )= -4.7843049980817
f(102)=f( R21001 )=  8.6009141902677
f(103)=f( I21001 )=  19.178343034727
f(106)=f( R10111 )=  4.1850829799698
f(107)=f( I10111 )=  2.3309988940022
f(110)=f( R30001 )= -6.7835204606562
f(111)=f( I30001 )= -9.5166636493808
f(116)=f( R10201 )= -6.7840050457633
f(117)=f( I10201 )=  2.2717178534445
f(120)=f( R01201 )= -5.9188086053657
f(121)=f( I01201 )=  4.9610563446273
f(122)=f( R31000 )=  3.4194320637620
```

```
f(123)=f( I31000 )=-0.99837297049187
f(124)=f( R00310 )=  3.2504355552144
f(125)=f( I00310 )=  3.0853610888697
f(126)=f( R20110 )=  5.1240890496662
f(127)=f( I20110 )= 0.74149732959070
f(128)=f( R11200 )= -1.5843167123449
f(129)=f( I11200 )= -6.3971418446779
f(138)=f( R40000 )= 0.37082739835335
f(139)=f( I40000 )=  2.5633143287422
f(140)=f( R00400 )=  1.8038388937664
f(141)=f( I00400 )= -1.7139352046444
f(150)=f( R20200 )=  2.5856472940933
f(151)=f( I20200 )=  4.3355174398109
f(152)=f( R20020 )=-0.54685437008740
f(153)=f( I20020 )=  5.2769019327362

resonance analysis of static map

third order terms removed

requested resonance driving terms written as a map

nonzero elements in generating polynomial in
the static resonance basis are :

f( 28)=f( R11001 )=-1.89854336897493E-03
f( 29)=f( R00111 )=-3.32599139785283E-03
f( 30)=f( R00003 )= -10.672804169663
f( 84)=f( R11002 )= -72.776585838634
f( 85)=f( R00112 )=  54.065292108939
f( 86)=f( R00004 )= -37.099234810374
f( 87)=f( R22000 )= -60.235266277165
f( 88)=f( R00220 )=-0.96214038597164
f( 89)=f( R11110 )= -9.0080359795728
f( 94)=f( R20002 )=  26.583833038774
f( 95)=f( I20002 )=  5.0447354474265
f( 96)=f( R00202 )=  9.2433061848593
f( 97)=f( I00202 )= -4.7064605396411
f(102)=f( R21001 )=  15.580571180531
f(103)=f( I21001 )=  32.720123059056
f(106)=f( R10111 )=  36.819927586957
f(107)=f( I10111 )= -114.08629573700
f(110)=f( R30001 )= -9.6117592954547
f(111)=f( I30001 )= -6.2709199935098
f(116)=f( R10201 )= -37.047065817947
f(117)=f( I10201 )= -38.087341467052
f(120)=f( R01201 )=  43.690060641248
f(121)=f( I01201 )=  10.886726350108
f(122)=f( R31000 )=  47.087258484683
f(123)=f( I31000 )= -60.423298401693
f(124)=f( R00310 )=  5.0107234448523
f(125)=f( I00310 )= -1.1073191899229
f(126)=f( R20110 )= -32.266751697609
```

```
f(127)=f( I20110 )= -56.043259583345
f(128)=f( R11200 )= -3.5204349928703
f(129)=f( I11200 )= -6.3161500914910
f(138)=f( R40000 )=  36.726701635870
f(139)=f( I40000 )=  14.248730865804
f(140)=f( R00400 )= 0.59862092808745
f(141)=f( I00400 )= -2.8583375517423
f(150)=f( R20200 )= -30.679200951848
f(151)=f( I20200 )=  8.7374523668605
f(152)=f( R20020 )=  31.312931864786
f(153)=f( I20020 )= -37.043419620519
```

8.5 Resonance Analyze Dynamic Map

Type Code: radm

Required Parameters:

1. IOPT
   = 0 to retain third order terms when computing fourth order terms.
   = 1 to remove third order terms when computing fourth order terms.
   Note: When IOPT = −1, the result is the same as when IOPT = +1 except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
2. ITHIRD
   = 0 to not compute analysis data.
   = 1 to write analysis data for cubic terms at the terminal.
   = 2 to write analysis data for cubic terms on file 12.
   = 3 to write analysis data at the terminal and on file 12.
3. IFOURTH
   = 0 to not compute analysis data.
   = 1 to write analysis data for quartic terms at the terminal.
   = 2 to write analysis data for quartic terms on file 12.
   = 3 to write analysis data at the terminal and on file 12.
4. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps on file IFILE.

Example:

```
dynres    radm
 1 , 1 , 1 , 0
```

This specifies a command with the user given name *dynres*. When invoked it removes third order "nonresonant" terms and analyzes the remaining third order and all the fourth order terms.

Description:

Suppose $\mathcal{M}$ is the map to be analyzed and IOPT = 0. Then this command finds a normalizing map

$$\mathcal{A} = \mathcal{A}_2$$

such that the purified map $\mathcal{N}$ has the form

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1} = \mathcal{N}_2(\exp : g_3 :)(\exp : g_4 :).$$

Here $\mathcal{N}_2$ is a linear map in dynamic normal form, and g_3 and g_4 are displayed in a dynamic resonance basis. Thus, the components of g_3 and g_4 in this basis are a measure of the strengths of third order and fourth order resonance driving terms.

In the case that IOPT = 1, this command finds a normalizing map

$$\mathcal{A} = \mathcal{A}_2\mathcal{A}_3$$

such that the purified map $\mathcal{N}$ has the form

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1} = \mathcal{N}_2\mathcal{N}_3 \exp(: h_4 :).$$

Here $\mathcal{N}_2$ is the same linear map in dynamic normal form and $\mathcal{N}_3$ has third order generators free of "nonresonant" terms. The $\mathcal{N}_3$ generator and h_4 are displayed in a dynamic resonance basis.

When a command with type code *radm* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}$, the normalizing map.
2. Buffer 2 contains the purified map in the cartesian basis.
3. Buffer 3 contains the purified map in the dynamic resonance basis.
4. Buffer 4 is empty.
5. Buffer 5 is empty.
6. Buffer 6 is empty.

The MaryLie run shown in the Exhibit below illustrates the use of a *radm* command with both options.

```
#comment
 Exhibit 8.5.
 This MaryLie run illustrates the use of the type code radm with
 both options for the small dynamic storage ring of Exhibit 2.6.1.
#beam
  4.78740000023600
  2807.64374771200
  1.00000000000000
  1.00000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
```

```
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.6094450000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.0110200000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       8.91000000000000
 vcs      sext
  0.200000000000000      -31.2000000000000
 cvty     srfc
  -350000.000000000       502000000.000000
 tadm     tadm
   1.00000000000000       123.000000000000       3.00000000000000
   1.00000000000000      0.000000000000000E+00
 radm0    radm
  0.000000000000000E+00   1.00000000000000       1.00000000000000
  0.000000000000000E+00
 radm1    radm
   1.00000000000000       1.00000000000000       1.00000000000000
  0.000000000000000E+00
 ps1      ps1
  1.000000000000000E-02  0.000000000000000E+00  7.000000000000000E-03
  0.000000000000000E+00  2.000000000000000E-02  0.000000000000000E+00
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 iden     iden
 twsx     twsm
   1.00000000000000       20.0000000000000       1.00000000000000
   2.00000000000000
 twsy     twsm
   2.00000000000000       30.0000000000000       3.00000000000000
   4.00000000000000
 twst     twsm
   3.00000000000000       40.0000000000000       5.00000000000000
   6.00000000000000
 zer      zer
  0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
  0.000000000000000E+00
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 half
     1*arc         1*cvty        1*arc
```

```
 statring
     4*arc
 dynring
     2*half
 test
     1*twsx        1*twsy        1*twst
#lumps
#loops
#labor
    1*fileout
    1*zer
    1*dynring
    1*radm0
    1*radm1
    1*fin

resonance analysis of dynamic map

requested resonance driving terms written as a map

nonzero elements in generating polynomial in
the dynamic resonance basis are :

f( 28)=f( R110010 )=-2.19978944080518E-02
f( 29)=f( I110010 )=-5.09138400608650E-04
f( 30)=f( R001110 )=-2.96825188048692E-02
f( 31)=f( I001110 )=-4.17347745361637E-03
f( 32)=f( R000030 )=-9.34191231733381E-04
f( 33)=f( I000030 )= 7.03845540004149E-03
f( 34)=f( R000021 )= 9.43941677616657E-04
f( 35)=f( I000021 )=-2.14121831789795E-02
f( 36)=f( R100011 )= 8.95384071260254E-03
f( 37)=f( I100011 )= 3.07199503455062E-02
f( 40)=f( R100020 )=-3.11239819497490E-04
f( 41)=f( I100020 )=-1.67344779116342E-02
f( 42)=f( R100002 )=-8.01293213911895E-03
f( 43)=f( I100002 )=-1.33666065665717E-02
f( 48)=f( R200010 )= 0.41108879852808
f( 49)=f( I200010 )=-0.18246081440543
f( 50)=f( R200001 )=-0.37740725948694
f( 51)=f( I200001 )= 0.22980078877788
f( 52)=f( R002010 )= 1.48751536754900E-02
f( 53)=f( I002010 )=-3.57544433030030E-02
f( 54)=f( R002001 )=-0.15114599503428
f( 55)=f( I002001 )=-1.65453775570663E-02
f( 64)=f( R210000 )= 0.55104196732507
f( 65)=f( I210000 )= 0.57936693933081
f( 68)=f( R101100 )=-0.20466033519870
f( 69)=f( I101100 )=  2.6432573016939
f( 72)=f( R300000 )= 0.56757237750316
f( 73)=f( I300000 )=-0.41146371681522
f( 78)=f( R102000 )= -6.7097552908067
f( 79)=f( I102000 )= -4.8168794964139
```

```
f( 82)=f( R100200 )= -7.4833147425705
f( 83)=f( I100200 )=  5.7220175717175
f( 84)=f( R220000 )= -42.022100162916
f( 85)=f( R002200 )=-0.80872037224564
f( 86)=f( R000022 )= -4.0212701553502
f( 87)=f( R111100 )=  26.366718018250
f( 88)=f( R110011 )= -2.8323490778510
f( 89)=f( R001111 )= -1.9552087459250
f( 90)=f( R110020 )=  2.2047093416394
f( 91)=f( I110020 )= 0.24401460453064
f( 92)=f( R001120 )=  1.9513707839233
f( 93)=f( I001120 )= 0.20936652582377
f( 94)=f( R000040 )= -1.3010466860427
f( 95)=f( I000040 )=-0.28767620389856
f( 96)=f( R000031 )= -5.2812233275056
f( 97)=f( I000031 )=-0.57677384279006
f( 98)=f( R100030 )=-0.27101008832195
f( 99)=f( I100030 )=-0.67471520221043
f(100)=f( R100003 )=-0.45295950895019
f(101)=f( I100003 )=-0.51001839825339
f(106)=f( R100021 )= -1.0308087531544
f(107)=f( I100021 )= -1.8440146050199
f(108)=f( R100012 )= -1.1973317106911
f(109)=f( I100012 )= -1.7475090204547
f(114)=f( R200011 )= -1.1209188570086
f(115)=f( I200011 )=  1.6721259213374
f(116)=f( R002011 )=-6.65611323781401E-02
f(117)=f( I002011 )= 0.23348184662992
f(118)=f( R200020 )= 0.82344475573375
f(119)=f( I200020 )= -1.0287039608324
f(120)=f( R200002 )= 0.52645811742143
f(121)=f( I200002 )= -1.0787281509191
f(122)=f( R002020 )= 3.51429623225890E-02
f(123)=f( I002020 )=-8.90361294716686E-02
f(124)=f( R002002 )= 3.05021551261502E-02
f(125)=f( I002002 )=-0.15195069460009
f(138)=f( R210010 )=-0.21875422062686
f(139)=f( I210010 )= 0.66047512266906
f(140)=f( R210001 )=-9.34496279974797E-02
f(141)=f( I210001 )=-0.53213602761745
f(146)=f( R101110 )=  2.1987473049335
f(147)=f( I101110 )= 0.58466970112729
f(148)=f( R101101 )= 0.75301564838345
f(149)=f( I101101 )=-6.06776606071731E-02
f(154)=f( R300010 )= 0.60605907986004
f(155)=f( I300010 )=-0.15606552400955
f(156)=f( R300001 )=-0.51984763345173
f(157)=f( I300001 )= 0.25846125448343
f(166)=f( R102010 )=  4.3592129309305
f(167)=f( I102010 )= 4.49920405965092E-02
f(168)=f( R102001 )= -3.0154536293288
f(169)=f( I102001 )=-0.57521031235161
f(174)=f( R100201 )=  4.9432735779290
```

```
f(175)=f( I100201 )= -9.7934519238561
f(176)=f( R100210 )= -5.4141208069259
f(177)=f( I100210 )=  10.314359775952
f(178)=f( R310000 )= -20.993401879515
f(179)=f( I310000 )=  40.949904788012
f(180)=f( R003100 )= -5.5744235907537
f(181)=f( I003100 )=  1.5597461909648
f(182)=f( R201100 )=  64.055773001312
f(183)=f( I201100 )=  44.662007307349
f(184)=f( R112000 )=  54.240703863451
f(185)=f( I112000 )= -45.972009756496
f(194)=f( R400000 )=  3.7275947891296
f(195)=f( I400000 )=  4.3600742354041
f(196)=f( R004000 )=  4.6943169683552
f(197)=f( I004000 )=  16.886776932036
f(206)=f( R202000 )=  26.288774532577
f(207)=f( I202000 )= -28.698163430731
f(208)=f( R200200 )=  17.482522799167
f(209)=f( I200200 )=  33.488531318382

resonance analysis of dynamic map

third order terms removed

requested resonance driving terms written as a map

nonzero elements in generating polynomial in
the dynamic resonance basis are :

f( 84)=f( R220000 )= -41.347036163370
f( 85)=f( R002200 )= -107.86511520173
f( 86)=f( R000022 )= -4.0276493572980
f( 87)=f( R111100 )=  505.92317355684
f( 88)=f( R110011 )= -2.5965048913001
f( 89)=f( R001111 )= -1.8680642544284
f( 90)=f( R110020 )=  1.9717279584754
f( 91)=f( I110020 )= 0.20843755891087
f( 92)=f( R001120 )=  1.8825233723552
f( 93)=f( I001120 )= 0.21221350378881
f( 94)=f( R000040 )= -1.2994748723816
f( 95)=f( I000040 )=-0.28732107608359
f( 96)=f( R000031 )= -5.2764937982982
f( 97)=f( I000031 )=-0.57650548887506
f( 98)=f( R100030 )=-0.27418097390425
f( 99)=f( I100030 )=-0.66670373378827
f(100)=f( R100003 )=-0.45675726095399
f(101)=f( I100003 )=-0.52057779889164
f(106)=f( R100021 )= -1.0441894977468
f(107)=f( I100021 )= -1.8686795010752
f(108)=f( R100012 )= -1.1787678797921
f(109)=f( I100012 )= -1.7242169893257
f(114)=f( R200011 )= -1.0520542561297
f(115)=f( I200011 )=  1.4100978467500
```

```
f(116)=f( R002011 )=-0.21813600089879
f(117)=f( I002011 )= 0.52777313213787
f(118)=f( R200020 )= 0.72858464469919
f(119)=f( I200020 )= -1.0126520789744
f(120)=f( R200002 )= 0.44611726645641
f(121)=f( I200002 )= -1.0403264576910
f(122)=f( R002020 )= 0.14207488449183
f(123)=f( I002020 )=-0.19292981820486
f(124)=f( R002002 )= 5.77755170179003E-02
f(125)=f( I002002 )=-0.37194309326294
f(138)=f( R210010 )=-0.17061005578537
f(139)=f( I210010 )=-0.13192162434174
f(140)=f( R210001 )= 2.54378083227571E-02
f(141)=f( I210001 )= 0.18151619596665
f(146)=f( R101110 )=  3.5689749423663
f(147)=f( I101110 )=  2.1687996210944
f(148)=f( R101101 )=  9.0679519047096
f(149)=f( I101101 )= -3.8551102004413
f(154)=f( R300010 )= 0.25905423680897
f(155)=f( I300010 )=-0.10572066160840
f(156)=f( R300001 )=-0.14922183803385
f(157)=f( I300001 )= 0.18218666060567
f(166)=f( R102010 )=  16.160573864816
f(167)=f( I102010 )=  4.4096337905211
f(168)=f( R102001 )= -15.500487666323
f(169)=f( I102001 )= -2.9935340668969
f(174)=f( R100201 )=  6.6886701471754
f(175)=f( I100201 )= -12.541033704758
f(176)=f( R100210 )= -7.2916232841087
f(177)=f( I100210 )=  11.659395190066
f(178)=f( R310000 )= -20.823644637169
f(179)=f( I310000 )=  39.687758221295
f(180)=f( R003100 )= -25.753235691739
f(181)=f( I003100 )=  26.921365547934
f(182)=f( R201100 )=  485.93713175751
f(183)=f( I201100 )=  30.267798554441
f(184)=f( R112000 )=  68.378545607683
f(185)=f( I112000 )= -46.870817166996
f(194)=f( R400000 )=  3.0179559837713
f(195)=f( I400000 )=  4.2348753104267
f(196)=f( R004000 )= -22.988250687191
f(197)=f( I004000 )= -74.095787091903
f(206)=f( R202000 )=  43.212497947728
f(207)=f( I202000 )= -14.650199008233
f(208)=f( R200200 )= -16.248121183108
f(209)=f( I200200 )= -34.750146505280
```

8.6 Translate Basis

Type Code: tbas

Required Parameters:

1. IOPT
   = 1 to translate from cartesian to static resonance basis.
   = 2 to translate from cartesian to dynamic resonance basis.
   = 3 to translate from static resonance to cartesian basis.
   = 4 to translate from dynamic resonance to cartesian basis.

Example:

```
dresdat    tbas
  2
```

This specifies a command with the user given name *dresdat*. It assumes the current transfer map is expressed in the cartesian basis. Since IOPT = 2, afer this command is executed the current transfer map is re-empressed in terms of the dynamic resonance basis.

Description:

MaryLie employs three different bases to describe polynomials: cartesian, static resonance, and dynamic resonance. See chapter 14 and the tables in sections 14.1 through 14.3. If a polynomial is to be printed out using a command with type code *ptm*, the parameter IBASIS should be set to value appropriate for the basis in which the polynomial is expressed. See section 7.7. Resonance bases are useful for examining the content of a polynomial, for filtering out various offensive terms with the use of *ftm* command (see section 7.22), for manufacturing special maps, and for fitting on various combinations of monomials (see section 10.6.1). All map analysis or application routines assume that the map to be analyzed or applied is in a cartesian basis. The basis in which a polynomial is expressed can be changed from cartesian to static resonance or dynamic resonance, or the reverse, by invoking a command with type code *tbas*. If (for some peculiar reason) the user wants to pass from a static resonance basis to a dynamic resonance basis, or the reverse, this can be done with two successive *tbas* commands. For example, to go from a static resonance basis to a dynamic resonance basis, the first command would have IOPT = 3 to go from a static resonance basis to a cartesian basis, and the second would have IOPT = 2 to go from a cartesian basis to a dynamic resonance basis.

8.7 Compute Exponential

Type Code: exp

Required Parameters:

1. POWER
2. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 to 9) to use map in storage location NMAP.
3. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting map in storage location KMAP.
   = 0 to make the resulting map the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting map in buffer number KBUF.

Example:

```
expf    exp
 2.5 , 1 , 0
```

This specifies a command with the user given name *expf*. Let f be the polynomial part of the map stored in location 1. When executed, this command computes the map $\mathcal{M} = \exp[(2.5) : f :]$ and makes it the current transfer map.

Description:

Given a polynomial f and a parameter λ it is often convenient to compute maps $\mathcal{M}$ of the form $\mathcal{M} = \exp(\lambda : f :)$. Here it is presumed that f has no terms of degree 1, or if present, they should be ignored. With regard to a possible constant term f_0, for convenience the map $\mathcal{M}$ associated with f is assigned a path length λf_0.

8.8 Static Normal Form Analysis

Type Code: snor

Required Parameters:

1. KEEP
   = 0 to keep only "exactly resonant" terms and remove all "nonresonant" terms.
   = IPSET (with IPSET an integer J from 1 to 9) to read a "filter" from an external file based on the parameter values in the parameter set IPSET.
2. IDATA
   = 0 to not put out analysis data.
   = 1 to put out normal form exponent.
   = 2 to put out complete pseudo hamiltonian.
   = 3 to put out normal form and pseudo hamiltonian.
3. IPMAPS
   = 0 to not put out maps.
   = 1 to put out normalizing map $\mathcal{A}$.
   = 2 to put out normal form $\mathcal{N}$.
   = 3 to put out normalizing map and normal form.
4. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.

   Note: When ISEND = -1, -2, or -3, the result is the same as when ISEND = $+1$, $+2$, or $+3$, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
5. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps and pseudo hamiltonian on file IFILE.

When KEEP $\neq 0$, the parameters P1...P6 from the parameter set associated with *psj* are used as follows (see section 7.22):

P1 = IFILE
P2 = NOPT
P3 = NSKIP

P4 = KIND
P5 = 0 (not used, but must be present to satisfy format requirements)
P6 = 0 (not used, but must be present to satisfy format requirements)

Example:

```
norm    snor
0 , 3 , 3 , 1 , 0
```

This specifies a command with the user given name *norm*. When invoked it puts out the normal form exponent, the pseudo hamiltonian, and the maps $\mathcal{A}$ and $\mathcal{N}$ on the assumption that the current map $\mathcal{M}$ is static and the transverse eigenvalues of its linear part lie on the unit circle and are not resonant.

Description:

Let $\mathcal{M}$ be a static map, and assume the transverse eigenvalues of its linear part lie on the unit circle and are not resonant. Then there is a normalizing map $\mathcal{A}$ such that the map $\mathcal{N}$ given by

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1}$$

has static normal form. See section 8.2. The normal form exponent g is defined by writing

$$\mathcal{N} = \exp : g : .$$

It follows that $\mathcal{M}$ can also be written in single exponent form,

$$\mathcal{M} = \mathcal{A}^{-1}\mathcal{N}\mathcal{A} = \mathcal{A}^{-1}(\exp : g :)\mathcal{A} = \exp[\mathcal{A}^{-1} : g : \mathcal{A}] = \exp(: \mathcal{A}^{-1}g :) = \exp(: h :)$$

where

$$h = \mathcal{A}^{-1}g.$$

The generator h is called the pseudo hamiltonian.

When a command with type code *snor* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}$, the normalizing map.
2. Buffer 2 contains the normal form map $\mathcal{N}$.
3. Buffer 3 contains the identity matrix in its matrix part and the normal form exponent in its array part.
4. Buffer 4 contains the identity matrix in its matrix part, and the pseudo hamiltonian in its array part.
5. Buffer 5 is empty.

6. Buffer 6 is empty.

The MaryLie run shown in the Exhibit below illustrates the use of the *snor* command.

```
#comment
  Exhibit 8.8.
  This MaryLie run illustrates the use of the type code
  snor for the small static storage ring of Exhibit 2.5.1.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000       1.00000000000000
    1.00000000000000
  hdq      quad
   0.500000000000000      -1.92000000000000       1.00000000000000
    1.00000000000000
  hcs      sext
   0.500000000000000      -1.62000000000000
  vcs      sext
   0.500000000000000       3.34000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  raysin   rt
    13.0000000000000       14.0000000000000      -1.00000000000000
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  track    rt
    13.0000000000000       14.0000000000000       5.00000000000000
    500.000000000000       1.00000000000000      0.000000000000000E+00
  chrom    tasm
    2.00000000000000      1.000000000000000E-03   1.00000000000000
   0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  snor     snor
   0.000000000000000E+00   3.00000000000000       3.00000000000000
    1.00000000000000      0.000000000000000E+00
```

```
  iden     iden
  zer      zer
   0.000000000000000E+00  5.000000000000000E-10  5.000000000000000E-10
   0.000000000000000E+00
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
#lumps
#loops
#labor
     1*fileout
     1*zer
     1*ring
     1*snor
     1*fin

static normal form analysis

exponent for normal form

nonzero elements in generating polynomial are :

  f(  7)=f( 20 00 00 )=-0.79828752664276
  f( 13)=f( 02 00 00 )=-0.79828752664276
  f( 18)=f( 00 20 00 )=-0.80248122015372
  f( 22)=f( 00 02 00 )=-0.80248122015372
  f( 27)=f( 00 00 02 )= -5.3023683040375
  f( 33)=f( 20 00 01 )= -6.9316102268115
  f( 53)=f( 02 00 01 )= -6.9316102268115
  f( 67)=f( 00 20 01 )= -15.779550156294
  f( 76)=f( 00 02 01 )= -15.779550156294
  f( 83)=f( 00 00 03 )= -28.010134513496
  f( 84)=f( 40 00 00 )= -6.5129435356557
  f( 90)=f( 22 00 00 )= -13.025887071311
  f( 95)=f( 20 20 00 )=-0.33642018654044
  f( 99)=f( 20 02 00 )=-0.33642018643167
  f(104)=f( 20 00 02 )= -121.63288412710
  f(140)=f( 04 00 00 )= -6.5129435356557
  f(145)=f( 02 20 00 )=-0.33642018643147
  f(149)=f( 02 02 00 )=-0.33642018654072
  f(154)=f( 02 00 02 )= -121.63288412710
```

```
  f(175)=f( 00 40 00 )= -1.4878351551883
  f(179)=f( 00 22 00 )= -2.9756703103767
  f(184)=f( 00 20 02 )= -18.880660741857
  f(195)=f( 00 04 00 )= -1.4878351551883
  f(200)=f( 00 02 02 )= -18.880660741858
  f(209)=f( 00 00 04 )= -100.13405799825

pseudo hamiltonian

nonzero elements in generating polynomial are :

  f(  7)=f( 20 00 00 )=-0.22578558226230
  f(  8)=f( 11 00 00 )= -1.4464587784470
  f( 12)=f( 10 00 01 )= -1.1619142004053
  f( 13)=f( 02 00 00 )= -5.1390514531704
  f( 17)=f( 01 00 01 )= -1.4127549494810
  f( 18)=f( 00 20 00 )=-0.22647807485868
  f( 19)=f( 00 11 00 )=  1.3980547332411
  f( 22)=f( 00 02 00 )= -5.0009934457920
  f( 27)=f( 00 00 02 )= -7.2694630964317
  f( 28)=f( 30 00 00 )=-6.50494239645407E-02
  f( 29)=f( 21 00 00 )= -1.4296836677323
  f( 33)=f( 20 00 01 )=-0.40869395506843
  f( 34)=f( 12 00 00 )= 0.55334461750424
  f( 38)=f( 11 00 01 )= -24.790343787318
  f( 39)=f( 10 20 00 )= -1.3275084402559
  f( 40)=f( 10 11 00 )=  6.9605355820113
  f( 43)=f( 10 02 00 )= -7.0840306681762
  f( 48)=f( 10 00 02 )=  9.7007786035018
  f( 49)=f( 03 00 00 )=  31.530925904754
  f( 53)=f( 02 00 01 )= -122.58526614797
  f( 54)=f( 01 20 00 )= 0.31290624872016
  f( 55)=f( 01 11 00 )=  14.741547167237
  f( 58)=f( 01 02 00 )= -43.575495252707
  f( 63)=f( 01 00 02 )=  44.015884671685
  f( 67)=f( 00 20 01 )= -12.115779219235
  f( 70)=f( 00 11 01 )=  47.519273230499
  f( 76)=f( 00 02 01 )= -52.485203074127
  f( 83)=f( 00 00 03 )= -17.836981057434
  f( 84)=f( 40 00 00 )=-0.73356773085699
  f( 85)=f( 31 00 00 )= -7.0544763343000
  f( 89)=f( 30 00 01 )= -6.2888672586137
  f( 90)=f( 22 00 00 )= -40.445426365101
  f( 94)=f( 21 00 01 )= -35.334465638042
  f( 95)=f( 20 20 00 )= -3.2120116456568
  f( 96)=f( 20 11 00 )=  12.876489268814
  f( 99)=f( 20 02 00 )= -26.780924888355
  f(104)=f( 20 00 02 )= -56.106294110038
  f(105)=f( 13 00 00 )= -171.23717153902
  f(109)=f( 12 00 01 )= -70.422880456703
  f(110)=f( 11 20 00 )= -11.859188988233
  f(111)=f( 11 11 00 )=  24.700180612381
  f(114)=f( 11 02 00 )= -91.363485403920
```

```
  f(119)=f( 11 00 02 )= -285.33759331255
  f(123)=f( 10 20 01 )= -31.703099130306
  f(126)=f( 10 11 01 )=  184.84623217230
  f(132)=f( 10 02 01 )= -292.07476551152
  f(139)=f( 10 00 03 )= -210.28624498273
  f(140)=f( 04 00 00 )= -481.69090842904
  f(144)=f( 03 00 01 )=  602.97081413949
  f(145)=f( 02 20 00 )= -57.844615084524
  f(146)=f( 02 11 00 )=  343.36463280866
  f(149)=f( 02 02 00 )= -702.54558355600
  f(154)=f( 02 00 02 )= -1955.6403607942
  f(158)=f( 01 20 01 )=  26.571107936173
  f(161)=f( 01 11 01 )= -202.99535807321
  f(167)=f( 01 02 01 )= -65.923662720770
  f(174)=f( 01 00 03 )=  649.54982847782
  f(175)=f( 00 40 00 )= -1.8585664452616
  f(176)=f( 00 31 00 )=  16.529026974099
  f(179)=f( 00 22 00 )= -66.378256434928
  f(184)=f( 00 20 02 )= -144.70816604931
  f(185)=f( 00 13 00 )=  166.63165037163
  f(190)=f( 00 11 02 )=  820.23250588993
  f(195)=f( 00 04 00 )= -250.40637641402
  f(200)=f( 00 02 02 )= -1098.0830036608
  f(209)=f( 00 00 04 )= -683.31338471679

normalizing map script A

matrix for map is :

  2.53724E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -3.88332E+00
 -3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  4.09055E-01
  0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
  3.48751E-01 -1.53053E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

  f( 28)=f( 30 00 00 )= 0.11524876175379
  f( 29)=f( 21 00 00 )=  1.2515351240181
  f( 33)=f( 20 00 01 )= 0.31146686998862
  f( 34)=f( 12 00 00 )=  7.5221737651877
  f( 38)=f( 11 00 01 )= -3.5797202887284
  f( 39)=f( 10 20 00 )= 0.19269224170971
  f( 40)=f( 10 11 00 )=-0.23874765194920
  f( 43)=f( 10 02 00 )=  6.5096974489741
  f( 48)=f( 10 00 02 )=  4.6338870765190
  f( 49)=f( 03 00 00 )=  10.550536896364
  f( 53)=f( 02 00 01 )=  5.1920485906365
  f( 54)=f( 01 20 00 )=  2.6033182213306
  f( 55)=f( 01 11 00 )= -20.702116048735
  f( 58)=f( 01 02 00 )=  71.558854955048
  f( 63)=f( 01 00 02 )= -17.821528349993
```

```
  f( 67)=f( 00 20 01 )= -1.0049839772826
  f( 70)=f( 00 11 01 )=  17.043323986621
  f( 76)=f( 00 02 01 )= -18.120490178280
  f( 83)=f( 00 00 03 )=  17.227231880558
  f( 84)=f( 40 00 00 )= 0.10703145055074
  f( 85)=f( 31 00 00 )=  2.1189451568260
  f( 89)=f( 30 00 01 )=-0.86409718199666
  f( 90)=f( 22 00 00 )=  10.241869643943
  f( 94)=f( 21 00 01 )=-0.87274305923961
  f( 95)=f( 20 20 00 )= -3.7961794506825
  f( 96)=f( 20 11 00 )=  266.23481653064
  f( 99)=f( 20 02 00 )= -741.12277022947
  f(104)=f( 20 00 02 )= -6.0589497741781
  f(105)=f( 13 00 00 )=  17.767501639493
  f(109)=f( 12 00 01 )= -1.6863417077846
  f(110)=f( 11 20 00 )= -265.29687606263
  f(111)=f( 11 11 00 )=  2971.9380326237
  f(114)=f( 11 02 00 )= -3283.6718164596
  f(119)=f( 11 00 02 )= -4.2385943045816
  f(123)=f( 10 20 01 )=  79.954563138599
  f(126)=f( 10 11 01 )=  872.17340524861
  f(132)=f( 10 02 01 )= -4746.8107726619
  f(139)=f( 10 00 03 )= -29.364772368317
  f(140)=f( 04 00 00 )=  8.1888122864298
  f(144)=f( 03 00 01 )=  13.179779492016
  f(145)=f( 02 20 00 )= -751.22804594571
  f(146)=f( 02 11 00 )=  3375.1778386752
  f(149)=f( 02 02 00 )=  6343.4134156609
  f(154)=f( 02 00 02 )= -99.703521616923
  f(158)=f( 01 20 01 )= -443.55059822109
  f(161)=f( 01 11 01 )=  9256.7920194779
  f(167)=f( 01 02 01 )= -19885.482739221
  f(174)=f( 01 00 03 )=  113.88619621872
  f(175)=f( 00 40 00 )=-3.64162370942615E-02
  f(176)=f( 00 31 00 )= 0.40974840871235
  f(179)=f( 00 22 00 )=  4.1392826612535
  f(184)=f( 00 20 02 )=  261.84971253562
  f(185)=f( 00 13 00 )= -40.404183016144
  f(190)=f( 00 11 02 )= -362.41946463192
  f(195)=f( 00 04 00 )=  57.781951346611
  f(200)=f( 00 02 02 )= -5367.3271317682
  f(209)=f( 00 00 04 )= -101.44675860192

normal form script N for transfer map

matrix for map is :

 -2.57759E-02  9.99668E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 -9.99668E-01 -2.57759E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -3.41595E-02  9.99416E-01  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00 -9.99416E-01 -3.41595E-02  0.00000E+00  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.06047E+01
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :

  f( 33)=f( 20 00 01 )= -6.9316102268115
  f( 53)=f( 02 00 01 )= -6.9316102268115
  f( 67)=f( 00 20 01 )= -15.779550156294
  f( 76)=f( 00 02 01 )= -15.779550156294
  f( 83)=f( 00 00 03 )= -28.010134513496
  f( 84)=f( 40 00 00 )= -6.5129435356557
  f( 90)=f( 22 00 00 )= -13.025887071311
  f( 95)=f( 20 20 00 )=-0.33642018654044
  f( 99)=f( 20 02 00 )=-0.33642018643167
  f(104)=f( 20 00 02 )= -121.63288412710
  f(140)=f( 04 00 00 )= -6.5129435356557
  f(145)=f( 02 20 00 )=-0.33642018643147
  f(149)=f( 02 02 00 )=-0.33642018654072
  f(154)=f( 02 00 02 )= -121.63288412710
  f(175)=f( 00 40 00 )= -1.4878351551883
  f(179)=f( 00 22 00 )= -2.9756703103767
  f(184)=f( 00 20 02 )= -18.880660741857
  f(195)=f( 00 04 00 )= -1.4878351551883
  f(200)=f( 00 02 02 )= -18.880660741858
  f(209)=f( 00 00 04 )= -100.13405799825
```

8.9 Dynamic Normal Form Analysis

Type Code: dnor

Required Parameters:

1. KEEP
   = 0 to keep only "exactly resonant" terms and remove all "nonresonant" terms.
   = IPSET (with IPSET and integer J from 1 to 9) to read a "filter" from an external file based on the parameter values in the parameter set IPSET.
2. IDATA
   = 0 to not put out analysis data.
   = 1 to put out normal form exponent.
   = 2 to put out complete pseudo hamiltonian.
   = 3 to put out normal form and pseudo hamiltonian.
3. IPMAPS
   = 0 to not put out maps.
   = 1 to put out normalizing map $\mathcal{A}$.
   = 2 to put out normal form $\mathcal{N}$.
   = 3 to put out normalizing map and normal form.
4. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.

   Note: When ISEND = -1, -2, or -3, the result is the same as when ISEND = $+1$, $+2$, or $+3$, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
5. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps and pseudo hamiltonian on file IFILE.

When KEEP $\neq 0$, the parameters P1$\cdots$P6 from the parameter set associated with *psj* are used as follows (see section 7.22):

P1 = IFILE
P2 = NOPT
P3 = NSKIP

P4 = KIND
P5 = 0 (not used, but must be present to satisfy format requirements)
P6 = 0 (not used, but must be present to satisfy format requirements)

Example:

```
dynnorm    dnor
0 , 3 , 3 , 1 , 0
```

This specifies a command with the user given name *dynnorm*. When invoked it puts out the normal form exponent, the pseudo hamiltonian, and the maps $\mathcal{A}$ and $\mathcal{N}$ on the assumption that the current map $\mathcal{M}$ is dynamic and the eigenvalues of its linear part lie on the unit circle and are not resonant.

Description:

Let $\mathcal{M}$ be a dynamic map, and assume the eigenvalues of its linear part lie on the unit circle and are not resonant. Then there is a normalizing map $\mathcal{A}$ such that the map $\mathcal{N}$ given by

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1}$$

has dynamic normal form. See section 8.3. The normal form exponent g is defined by writing

$$\mathcal{N} = \exp : g : .$$

It follows that $\mathcal{M}$ can also be written in single exponent form,

$$\mathcal{M} = \mathcal{A}^{-1}\mathcal{N}\mathcal{A} = \mathcal{A}^{-1}(\exp : g :)\mathcal{A} = \exp[\mathcal{A}^{-1} : g : \mathcal{A}] = \exp(: \mathcal{A}^{-1}g :) = \exp(: h :)$$

where

$$h = \mathcal{A}^{-1}g.$$

The generator h is called the pseudo hamiltonian.

When a command with type code *dnor* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}$, the normalizing map.
2. Buffer 2 contains the normal form map $\mathcal{N}$.
3. Buffer 3 contains the identity matrix in its matrix part and the normal form exponent in its array part.
4. Buffer 4 contains the identity matrix in its matrix part, and the pseudo hamiltonian in its array part.
5. Buffer 5 is empty.

6. Buffer 6 is empty.

The MaryLie run shown in the Exhibit below illustrates the use of the *dnor* command. See also Exhibit 2.6.2.

```
#comment
 Exhibit 8.9.
 This MaryLie run illustrates the use of the type code dnor
 for the small dynamic storage ring of Exhibit 2.6.1.
#beam
  4.78740000023600
  2807.64374771200
  1.00000000000000
  1.00000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.6094450000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.0110200000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       8.91000000000000
 vcs      sext
  0.200000000000000      -31.2000000000000
 cvty     srfc
  -350000.000000000       502000000.000000
 tadm     tadm
   1.00000000000000       123.000000000000       3.00000000000000
   1.00000000000000      0.000000000000000E+00
 radm0    radm
  0.000000000000000E+00   1.00000000000000       1.00000000000000
  0.000000000000000E+00
 radm1    radm
   1.00000000000000       1.00000000000000       1.00000000000000
  0.000000000000000E+00
 dnor     dnor
  0.000000000000000E+00   3.00000000000000       3.00000000000000
   1.00000000000000      0.000000000000000E+00
 ps1      ps1
  1.000000000000000E-02  0.000000000000000E+00  7.000000000000000E-03
```

```
  0.000000000000000E+00  2.000000000000000E-02  0.000000000000000E+00
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 iden     iden
 twsx     twsm
   1.00000000000000       20.0000000000000       1.00000000000000
   2.00000000000000
 twsy     twsm
   2.00000000000000       30.0000000000000       3.00000000000000
   4.00000000000000
 twst     twsm
   3.00000000000000       40.0000000000000       5.00000000000000
   6.00000000000000
 zer      zer
  0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
  0.000000000000000E+00
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 half
     1*arc         1*cvty        1*arc
 statring
     4*arc
 dynring
     2*half
 test
     1*twsx        1*twsy        1*twst
#lumps
#loops
#labor
    1*fileout
    1*zer
    1*dynring
    1*dnor
    1*fin

dynamic normal form analysis

exponent for normal form

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )=-0.55484710390720
 f( 13)=f( 02 00 00 )=-0.55484710390720
 f( 18)=f( 00 20 00 )=-0.37303541151836
 f( 22)=f( 00 02 00 )=-0.37303541151836
 f( 25)=f( 00 00 20 )= 5.43965713082494E-02
```

```
 f( 27)=f( 00 00 02 )= 5.43965713082494E-02
 f( 84)=f( 40 00 00 )= -41.347036163370
 f( 90)=f( 22 00 00 )= -82.694072326741
 f( 95)=f( 20 20 00 )=  505.92317355684
 f( 99)=f( 20 02 00 )=  505.92317355684
 f(102)=f( 20 00 20 )= -2.5965048913001
 f(104)=f( 20 00 02 )= -2.5965048913000
 f(140)=f( 04 00 00 )= -41.347036163370
 f(145)=f( 02 20 00 )=  505.92317355684
 f(149)=f( 02 02 00 )=  505.92317355684
 f(152)=f( 02 00 20 )= -2.5965048913001
 f(154)=f( 02 00 02 )= -2.5965048913001
 f(175)=f( 00 40 00 )= -107.86511520173
 f(179)=f( 00 22 00 )= -215.73023040346
 f(182)=f( 00 20 20 )= -1.8680642544285
 f(184)=f( 00 20 02 )= -1.8680642544283
 f(195)=f( 00 04 00 )= -107.86511520173
 f(198)=f( 00 02 20 )= -1.8680642544281
 f(200)=f( 00 02 02 )= -1.8680642544284
 f(205)=f( 00 00 40 )= -4.0276493572981
 f(207)=f( 00 00 22 )= -8.0552987145964
 f(209)=f( 00 00 04 )= -4.0276493572981

pseudo hamiltonian

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )=-6.51847654198605E-02
 f( 12)=f( 10 00 01 )=-0.23857534571501
 f( 13)=f( 02 00 00 )= -4.6968945777365
 f( 16)=f( 01 00 10 )= 1.89056507263843E-02
 f( 18)=f( 00 20 00 )=-0.17931488579704
 f( 22)=f( 00 02 00 )=-0.77603941038211
 f( 25)=f( 00 00 20 )= 2.56674875817710E-03
 f( 27)=f( 00 00 02 )= 0.93235200605533
 f( 28)=f( 30 00 00 )= 3.05660888347550E-02
 f( 29)=f( 21 00 00 )=  1.3243095646355
 f( 32)=f( 20 00 10 )=-2.38325666567029E-03
 f( 33)=f( 20 00 01 )= 0.51738244135089
 f( 34)=f( 12 00 00 )= -4.6721648167129
 f( 37)=f( 11 00 10 )= 9.82520227127529E-03
 f( 38)=f( 11 00 01 )= -3.4258000702678
 f( 39)=f( 10 20 00 )=-0.67507958513373
 f( 40)=f( 10 11 00 )= -12.026391795995
 f( 43)=f( 10 02 00 )= 0.56883323882708
 f( 46)=f( 10 00 20 )=-1.65154120975919E-06
 f( 47)=f( 10 00 11 )= 3.76823656606002E-04
 f( 48)=f( 10 00 02 )=  1.4499394933109
 f( 49)=f( 03 00 00 )= -4.5744476841350
 f( 52)=f( 02 00 10 )= 7.32548275325167E-03
 f( 53)=f( 02 00 01 )= -33.799889823831
 f( 54)=f( 01 20 00 )= -1.5952253770015
 f( 55)=f( 01 11 00 )= -59.233494944063
```

```
f( 58)=f( 01 02 00 )=  34.149622431586
f( 61)=f( 01 00 20 )= 7.46179561712994E-06
f( 62)=f( 01 00 11 )= 3.95865925100851E-02
f( 63)=f( 01 00 02 )= -6.5670331456014
f( 66)=f( 00 20 10 )=-8.51901922285355E-03
f( 67)=f( 00 20 01 )=-0.97630739442548
f( 69)=f( 00 11 10 )=-6.49961910273015E-03
f( 70)=f( 00 11 01 )= -23.092152381539
f( 75)=f( 00 02 10 )=-1.79654294466696E-02
f( 76)=f( 00 02 01 )=-8.69627522899012E-02
f( 80)=f( 00 00 30 )=-2.27608950105556E-10
f( 81)=f( 00 00 21 )=-1.14358441552529E-05
f( 82)=f( 00 00 12 )= 3.44602263275787E-04
f( 83)=f( 00 00 03 )= -1.6691021501565
f( 84)=f( 40 00 00 )= -1.6319887782438
f( 85)=f( 31 00 00 )= 6.42884448291736E-02
f( 88)=f( 30 00 10 )=-2.97866891859795E-03
f( 89)=f( 30 00 01 )= -10.962400349340
f( 90)=f( 22 00 00 )=  2.5244347957560
f( 93)=f( 21 00 10 )= 1.68701629757525E-02
f( 94)=f( 21 00 01 )= -3.8741288737795
f( 95)=f( 20 20 00 )= 0.68217225014840
f( 96)=f( 20 11 00 )=  7.6836123426788
f( 99)=f( 20 02 00 )= -66.121981629793
f(102)=f( 20 00 20 )= 1.09364252937803E-04
f(103)=f( 20 00 11 )=-5.21607452496559E-03
f(104)=f( 20 00 02 )= -55.820824526417
f(105)=f( 13 00 00 )= -7.4140935756961
f(108)=f( 12 00 10 )= 2.13868882107822E-02
f(109)=f( 12 00 01 )= -24.482013851041
f(110)=f( 11 20 00 )=  98.174762501672
f(111)=f( 11 11 00 )= -55.135937405321
f(114)=f( 11 02 00 )= -404.01822198314
f(117)=f( 11 00 20 )=-3.70706513756391E-05
f(118)=f( 11 00 11 )= 6.26752980506865E-02
f(119)=f( 11 00 02 )= -39.489183823312
f(122)=f( 10 20 10 )=-1.21263351409728E-02
f(123)=f( 10 20 01 )=  24.391874260967
f(125)=f( 10 11 10 )=-8.35499281588867E-02
f(126)=f( 10 11 01 )=  6.8182020316210
f(131)=f( 10 02 10 )=-1.70891066153328E-02
f(132)=f( 10 02 01 )= -331.09088070773
f(136)=f( 10 00 30 )= 2.73011671433880E-09
f(137)=f( 10 00 21 )= 3.92476336414212E-03
f(138)=f( 10 00 12 )= 1.97008369171889E-02
f(139)=f( 10 00 03 )= -130.13121845361
f(140)=f( 04 00 00 )= -33.891556870440
f(143)=f( 03 00 10 )= -4.6793060846351
f(144)=f( 03 00 01 )= -39.407390383280
f(145)=f( 02 20 00 )=  273.59805097959
f(146)=f( 02 11 00 )= -491.79881761011
f(149)=f( 02 02 00 )= -1014.4297220519
f(152)=f( 02 00 20 )= -1.9293126813240
```

```
 f(153)=f( 02 00 11 )= 2.59832559364861E-02
 f(154)=f( 02 00 02 )= -77.583791995084
 f(157)=f( 01 20 10 )= 5.79767940054812E-02
 f(158)=f( 01 20 01 )=  181.74299791586
 f(160)=f( 01 11 10 )= 1.33153297250469E-02
 f(161)=f( 01 11 01 )= -61.954309133107
 f(166)=f( 01 02 10 )=-9.11981107177066E-02
 f(167)=f( 01 02 01 )= -826.01391838102
 f(171)=f( 01 00 30 )=-0.34921099352855
 f(172)=f( 01 00 21 )=-1.39188082071072E-05
 f(173)=f( 01 00 12 )= 0.30109686957793
 f(174)=f( 01 00 03 )= -39.802397347070
 f(175)=f( 00 40 00 )= -4.3213491275024
 f(176)=f( 00 31 00 )= -4.8493855397512
 f(179)=f( 00 22 00 )= -24.607100198005
 f(182)=f( 00 20 20 )=-1.55996081826755E-04
 f(183)=f( 00 20 11 )=-1.06721025516105E-02
 f(184)=f( 00 20 02 )= -1.4137901700869
 f(185)=f( 00 13 00 )= -21.614944393373
 f(188)=f( 00 11 20 )=-3.10824531565144E-06
 f(189)=f( 00 11 11 )= 8.09607796236020E-02
 f(190)=f( 00 11 02 )= 0.49835864464763
 f(195)=f( 00 04 00 )= -89.096838535216
 f(198)=f( 00 02 20 )=-4.84459334867349E-05
 f(199)=f( 00 02 11 )=-6.18299023650477E-02
 f(200)=f( 00 02 02 )= -544.30960871861
 f(205)=f( 00 00 40 )=-2.37053577589241E-02
 f(206)=f( 00 00 31 )= 2.86151999046139E-09
 f(207)=f( 00 00 22 )= 2.23049977478240E-02
 f(208)=f( 00 00 13 )= 2.41714801520909E-02
 f(209)=f( 00 00 04 )= -118.72340862395

normalizing map script A

matrix for map is :

 2.91212E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -3.93992E-01
 0.00000E+00  3.42760E-01  0.00000E+00  0.00000E+00  4.67616E-03  0.00000E+00
 0.00000E+00  0.00000E+00  1.44234E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  6.93319E-01  0.00000E+00  0.00000E+00
 0.00000E+00 -6.21097E-01  0.00000E+00  0.00000E+00  4.59072E+00  0.00000E+00
 2.96632E-03  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  2.17430E-01

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= 4.24563094309178E-02
 f( 29)=f( 21 00 00 )=-0.23362179700584
 f( 32)=f( 20 00 10 )= 4.56563375582720E-04
 f( 33)=f( 20 00 01 )=-9.75890742675257E-02
 f( 34)=f( 12 00 00 )=-0.50107227049797
 f( 37)=f( 11 00 10 )=-4.07175935420772E-03
 f( 38)=f( 11 00 01 )= -3.5521666819433
 f( 39)=f( 10 20 00 )= -3.3786897291313
```

```
f( 40)=f( 10 11 00 )=  10.274152915083
f( 43)=f( 10 02 00 )=  17.522707805072
f( 46)=f( 10 00 20 )=-1.07393332756529E-05
f( 47)=f( 10 00 11 )=-2.50196794546159E-03
f( 48)=f( 10 00 02 )=-0.33993980234456
f( 49)=f( 03 00 00 )= 0.53759204610951
f( 52)=f( 02 00 10 )=-2.33222724286306E-02
f( 53)=f( 02 00 01 )=  7.0095220997548
f( 54)=f( 01 20 00 )= -23.055922845211
f( 55)=f( 01 11 00 )= -84.417256948179
f( 58)=f( 01 02 00 )=  117.83376922949
f( 61)=f( 01 00 20 )= 9.67252218481395E-03
f( 62)=f( 01 00 11 )= 6.25580127135236E-03
f( 63)=f( 01 00 02 )= -2.3883197693790
f( 66)=f( 00 20 10 )= 1.57085959872146E-02
f( 67)=f( 00 20 01 )= -6.9327429984476
f( 69)=f( 00 11 10 )= 0.10262987377485
f( 70)=f( 00 11 01 )=  17.978402203975
f( 75)=f( 00 02 10 )=-6.50892505078353E-02
f( 76)=f( 00 02 01 )=  30.003542446900
f( 80)=f( 00 00 30 )= 1.80160797959117E-03
f( 81)=f( 00 00 21 )=-3.95951672783498E-05
f( 82)=f( 00 00 12 )=  1.2006711790575
f( 83)=f( 00 00 03 )=-0.30761142712027
f( 84)=f( 40 00 00 )=-8.93719151360201E-03
f( 85)=f( 31 00 00 )=  7.3934647667393
f( 88)=f( 30 00 10 )=-9.85506393337277E-03
f( 89)=f( 30 00 01 )=-8.37787968503297E-02
f( 90)=f( 22 00 00 )= 0.60218717588989
f( 93)=f( 21 00 10 )= 1.66506667801550E-02
f( 94)=f( 21 00 01 )=  43.327148574623
f( 95)=f( 20 20 00 )=  12.965277177406
f( 96)=f( 20 11 00 )= -15.068546134915
f( 99)=f( 20 02 00 )=  66.289167380011
f(102)=f( 20 00 20 )=-4.23488291946144E-04
f(103)=f( 20 00 11 )=  2.0565827442989
f(104)=f( 20 00 02 )=-0.97702215928762
f(105)=f( 13 00 00 )=  312.30554494092
f(108)=f( 12 00 10 )= -1.5325588535848
f(109)=f( 12 00 01 )=  7.4091602058811
f(110)=f( 11 20 00 )= -97.009628172142
f(111)=f( 11 11 00 )= -311.45470328356
f(114)=f( 11 02 00 )= -454.00490893284
f(117)=f( 11 00 20 )=-0.12467262428162
f(118)=f( 11 00 11 )= 2.57366306237254E-02
f(119)=f( 11 00 02 )=  186.96854318158
f(122)=f( 10 20 10 )=  1.1047750369854
f(123)=f( 10 20 01 )=  30.916609264541
f(125)=f( 10 11 10 )= -1.1184727940998
f(126)=f( 10 11 01 )= -346.90281956655
f(131)=f( 10 02 10 )= -2.2443985860602
f(132)=f( 10 02 01 )=  350.75460910199
f(136)=f( 10 00 30 )=-1.64906156413144E-02
```

```
 f(137)=f( 10 00 21 )= 7.52459446224435E-04
 f(138)=f( 10 00 12 )=  10.371250485781
 f(139)=f( 10 00 03 )= -1.5817798716743
 f(140)=f( 04 00 00 )=  34.309229666723
 f(142)=f( 03 01 00 )= 2.55242497880502E-10
 f(143)=f( 03 00 10 )= 0.75992561291716
 f(144)=f( 03 00 01 )=  901.55277833062
 f(145)=f( 02 20 00 )= -863.56566663938
 f(146)=f( 02 11 00 )= -891.84993361085
 f(149)=f( 02 02 00 )= -5151.5659566156
 f(152)=f( 02 00 20 )= 5.25654967164942E-02
 f(153)=f( 02 00 11 )=  128.93327171225
 f(154)=f( 02 00 02 )=  50.229620445406
 f(157)=f( 01 20 10 )=  1.2644045823879
 f(158)=f( 01 20 01 )=  205.72062820517
 f(160)=f( 01 11 10 )=  28.758219913498
 f(161)=f( 01 11 01 )= -1261.9432336499
 f(166)=f( 01 02 10 )= -15.371600405125
 f(167)=f( 01 02 01 )= -2453.4322901979
 f(171)=f( 01 00 30 )=-1.72934044107995E-05
 f(172)=f( 01 00 21 )= -15.230903930439
 f(173)=f( 01 00 12 )=-3.88627021873839E-02
 f(174)=f( 01 00 03 )= -1164.0826768022
 f(175)=f( 00 40 00 )= -2.9408447025029
 f(176)=f( 00 31 00 )= -43.158179318091
 f(179)=f( 00 22 00 )=  86.484838081922
 f(182)=f( 00 20 20 )=-1.43911542812581E-03
 f(183)=f( 00 20 11 )=  10.033672598524
 f(184)=f( 00 20 02 )=  17.505456809913
 f(185)=f( 00 13 00 )=  414.71957260645
 f(188)=f( 00 11 20 )=-3.04209386618690E-02
 f(189)=f( 00 11 11 )= -1.6112088367381
 f(190)=f( 00 11 02 )= -545.89925987384
 f(195)=f( 00 04 00 )= -69.681447335178
 f(198)=f( 00 02 20 )= 1.95736175913004E-02
 f(199)=f( 00 02 11 )=  33.880139296519
 f(200)=f( 00 02 02 )=  403.03147658488
 f(205)=f( 00 00 40 )= 1.98111795926177E-06
 f(206)=f( 00 00 31 )= -2.9247770467390
 f(207)=f( 00 00 22 )= 2.40613662410394E-03
 f(208)=f( 00 00 13 )= -761.25171020629
 f(209)=f( 00 00 04 )=-0.41113091998924

normal form script N for transfer map

matrix for map is :

 4.44935E-01  8.95563E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-8.95563E-01  4.44935E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  7.34361E-01  6.78759E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -6.78759E-01  7.34361E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  9.94088E-01 -1.08579E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.08579E-01  9.94088E-01
```

```
nonzero elements in generating polynomial are :

 f( 84)=f( 40 00 00 )= -41.347036163370
 f( 90)=f( 22 00 00 )= -82.694072326741
 f( 95)=f( 20 20 00 )=  505.92317355684
 f( 99)=f( 20 02 00 )=  505.92317355684
 f(102)=f( 20 00 20 )= -2.5965048913001
 f(104)=f( 20 00 02 )= -2.5965048913000
 f(140)=f( 04 00 00 )= -41.347036163370
 f(145)=f( 02 20 00 )=  505.92317355684
 f(149)=f( 02 02 00 )=  505.92317355684
 f(152)=f( 02 00 20 )= -2.5965048913001
 f(154)=f( 02 00 02 )= -2.5965048913001
 f(175)=f( 00 40 00 )= -107.86511520173
 f(179)=f( 00 22 00 )= -215.73023040346
 f(182)=f( 00 20 20 )= -1.8680642544285
 f(184)=f( 00 20 02 )= -1.8680642544283
 f(195)=f( 00 04 00 )= -107.86511520173
 f(198)=f( 00 02 20 )= -1.8680642544281
 f(200)=f( 00 02 02 )= -1.8680642544284
 f(205)=f( 00 00 40 )= -4.0276493572981
 f(207)=f( 00 00 22 )= -8.0552987145964
 f(209)=f( 00 00 04 )= -4.0276493572981
```

8.10 Static Invariant Analysis

Type Code: sia

Required Parameters:

1. IOPT
   = 0 to compute regular invariants.
   = INFILE ≥ 1 to read parameters for mixed invariant from an external file with number INFILE.
   = $-$J (with J an integer from 1 to 9) to read parameters for mixed invariant from the parameter set array associated with the type code *psj*.
2. IPINV
   = 0 to not put out invariants.
   = 1 to put out invariants.
3. IPMAPS
   = 0 to not put out maps.
   = 1 to put out normalizing map $\mathcal{A}$.
   = 2 to put out normal form $\mathcal{N}$.
   = 3 to put out normalizing map and normal form.
4. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
   Note: When ISEND = -1, -2, or -3, the result is the same as when ISEND = $+1$, $+2$, or $+3$, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
5. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps and invariant polynomials on file IFILE.

When IOPT = INFILE ≥ 1, the file INFILE should contain the values ANX and ANY written on one line in free format. When IOPT = $-J < 0$, the contents of the parameter set with index J are as follows:

1. ANX
2. ANY
3. 0 (not used, but must be present)
4. 0 (not used, but must be present)

5. 0 (not used, but must be present)
6. 0 (not used, but must be present)

Example:

```
invar    sia
 0 , 1 , 0 , 3, 0
```

This specifies a command with the user given name *invar*. When invoked it puts out the regular invariants at the terminal and on file 12 on the assumption that the current map $\mathcal{M}$ is static and the transverse eigenvalues of its linear part lie on the unit circle and are not resonant.

Description:

Let $\mathcal{M}$ be a static map and assume the transverse eigenvalues of its linear part lie on the unit circle and are not resonant. Then there is a normalizing map $\mathcal{A}$ such that the map given by

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1}$$

has static normal form. See section 8.8. The "regular" invariants I_x and I_y are given by the relations

$$I_x = \mathcal{A}^{-1}(P_x^2 + X^2),$$
$$I_y = \mathcal{A}^{-1}(P_y^2 + Y^2).$$

These quantities are nonlinear generalizations of the Courant-Snyder invariants, and satisfy the relations

$$\mathcal{M}I_{x,y} = I_{x,y}.$$

The *mixed* invariant I_m is defined by the relation

$$I_m = (ANX)I_x + (ANY)I_y.$$

It is also invariant in the nonresonant case, and remains invariant in the resonant case providing the integers (ANX) and (ANY) are those *complementary* to the resonance. For example, for the $(2\nu_x - \nu_y)$ resonance, they should have the values $ANX = 1$ and $ANY = 2$.

When a command with type code *sia* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}$, the normalizing map.
2. Buffer 2 contains the normal form map $\mathcal{N}$.
3. If IOPT $= 0$, buffer 3 contains the identity matrix in its matrix part, and the x invariant polynomial in its array part. Otherwise buffer 3 contains the mixed invariant.
4. If IOPT $= 0$, buffer 4 contains the identity matrix in its matrix part, and the y invariant polynomial in its array part. Otherwise buffer 4 is empty.

5. Buffer 5 is empty.

6. Buffer 6 is empty.

The MaryLie run shown in the Exhibit below illustrates the use of the *sia* command. See also Exhibits 8.11 and 8.17.

```
#comment
 Exhibit 8.10.
 This MaryLie run illustrates the use of the type code
 sia for the small static storage ring of Exhibit 2.5.1.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       2.72000000000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.500000000000000      -1.92000000000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.500000000000000      -1.62000000000000
 vcs      sext
  0.500000000000000       3.34000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
   13.0000000000000       14.0000000000000       5.00000000000000
   500.000000000000       1.00000000000000      0.000000000000000E+00
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
```

```
 snor     snor
  0.000000000000000E+00   3.00000000000000       3.00000000000000
   1.00000000000000      0.000000000000000E+00
 sia      sia
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
   3.00000000000000      0.000000000000000E+00
 iden     iden
 zer      zer
  0.000000000000000E+00  5.000000000000000E-10  5.000000000000000E-10
  0.000000000000000E+00
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
#lumps
#loops
#labor
    1*fileout
    1*zer
    1*ring
    1*sia
    1*fin

static invariant analysis

x invariant polynomial

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.28283741725472
 f(  8)=f( 11 00 00 )=  1.8119521227272
 f( 12)=f( 10 00 01 )=  1.4555083997013
 f( 13)=f( 02 00 00 )=  6.4375945779623
 f( 17)=f( 01 00 01 )=  1.7697319603908
 f( 27)=f( 00 00 02 )=  2.4641432149979
 f( 28)=f( 30 00 00 )= 8.14862086573107E-02
 f( 29)=f( 21 00 00 )=  1.7909382522170
 f( 33)=f( 20 00 01 )= -1.9439421600706
 f( 34)=f( 12 00 00 )=-0.69316455416935
 f( 38)=f( 11 00 01 )=  15.321043502166
 f( 39)=f( 10 20 00 )=  1.1234824876277
 f( 40)=f( 10 11 00 )= -11.278066755118
```

```
 f( 43)=f( 10 02 00 )=  28.683783323401
 f( 48)=f( 10 00 02 )= -24.790310322533
 f( 49)=f( 03 00 00 )= -39.498206914693
 f( 53)=f( 02 00 01 )=  97.662016671074
 f( 54)=f( 01 20 00 )=  2.2361389163827
 f( 55)=f( 01 11 00 )= -34.437321940070
 f( 58)=f( 01 02 00 )=  45.847633132341
 f( 63)=f( 01 00 02 )= -70.504642686509
 f( 67)=f( 00 20 01 )=  3.4481379195691
 f( 70)=f( 00 11 01 )= -29.709584450401
 f( 76)=f( 00 02 01 )=  92.634094832153
 f( 83)=f( 00 00 03 )= -34.140122269086
 f( 84)=f( 40 00 00 )= 0.26625964929444
 f( 85)=f( 31 00 00 )= 0.47460012884628
 f( 89)=f( 30 00 01 )= 0.45302042859664
 f( 90)=f( 22 00 00 )= -5.8313785812592
 f( 94)=f( 21 00 01 )= -22.489395096546
 f( 95)=f( 20 20 00 )= -133.94768609570
 f( 96)=f( 20 11 00 )=  716.49307547164
 f( 99)=f( 20 02 00 )=  774.63973894177
 f(104)=f( 20 00 02 )=  15.411106871683
 f(105)=f( 13 00 00 )=  24.170787005918
 f(109)=f( 12 00 01 )= -110.98043731593
 f(110)=f( 11 20 00 )= -729.49630673371
 f(111)=f( 11 11 00 )= -3093.2302592337
 f(114)=f( 11 02 00 )=  26125.403574480
 f(119)=f( 11 00 02 )= -166.56546917394
 f(123)=f( 10 20 01 )= -765.01365459675
 f(126)=f( 10 11 01 )=  7138.4652828102
 f(132)=f( 10 02 01 )= -5514.3389878372
 f(139)=f( 10 00 03 )=  198.38331770005
 f(140)=f( 04 00 00 )=  265.28967521583
 f(144)=f( 03 00 01 )= -598.26336904547
 f(145)=f( 02 20 00 )=  756.61139263247
 f(146)=f( 02 11 00 )= -26007.556806726
 f(149)=f( 02 02 00 )=  64393.054340637
 f(154)=f( 02 00 02 )=  336.51063798049
 f(158)=f( 01 20 01 )= -3530.0340246697
 f(161)=f( 01 11 01 )=  9875.9553103357
 f(167)=f( 01 02 01 )=  49139.325361611
 f(174)=f( 01 00 03 )= -542.28702501823
 f(175)=f( 00 40 00 )=  1.2469479054074
 f(176)=f( 00 31 00 )= -22.724591921870
 f(179)=f( 00 22 00 )=  178.46754108576
 f(184)=f( 00 20 02 )= -798.84747137678
 f(185)=f( 00 13 00 )= -582.86899632005
 f(190)=f( 00 11 02 )=  12419.528107262
 f(195)=f( 00 04 00 )=  877.06223497910
 f(200)=f( 00 02 02 )= -22602.667052125
 f(209)=f( 00 00 04 )=  601.98597677505

y invariant polynomial
```

```
nonzero elements in generating polynomial are :

 f( 18)=f( 00 20 00 )= 0.28222227408050
 f( 19)=f( 00 11 00 )= -1.7421650477668
 f( 22)=f( 00 02 00 )=  6.2319133709247
 f( 39)=f( 10 20 00 )= 0.53664356643599
 f( 40)=f( 10 11 00 )=  2.5453610401654
 f( 43)=f( 10 02 00 )= -19.706225365361
 f( 54)=f( 01 20 00 )= -2.6143765122715
 f( 55)=f( 01 11 00 )=  15.887396574912
 f( 58)=f( 01 02 00 )=  8.6929156992971
 f( 67)=f( 00 20 01 )=  6.1183153892148
 f( 70)=f( 00 11 01 )=  4.5958685600997
 f( 76)=f( 00 02 01 )= -149.28726800044
 f( 95)=f( 20 20 00 )=  137.21682431703
 f( 96)=f( 20 11 00 )= -728.58802024760
 f( 99)=f( 20 02 00 )= -737.95782611400
 f(110)=f( 11 20 00 )=  740.24779668774
 f(111)=f( 11 11 00 )=  3047.6089348420
 f(114)=f( 11 02 00 )= -25879.757217720
 f(123)=f( 10 20 01 )=  780.09330458197
 f(126)=f( 10 11 01 )= -7283.0744155013
 f(132)=f( 10 02 01 )=  5985.4133761219
 f(145)=f( 02 20 00 )= -681.33686819601
 f(146)=f( 02 11 00 )=  25448.466733518
 f(149)=f( 02 02 00 )= -63197.894117966
 f(158)=f( 01 20 01 )=  3510.3583312332
 f(161)=f( 01 11 01 )= -9585.0333015133
 f(167)=f( 01 02 01 )= -49371.953367217
 f(175)=f( 00 40 00 )= 0.92791989651083
 f(176)=f( 00 31 00 )=  3.8316189898871
 f(179)=f( 00 22 00 )= -106.96761937402
 f(184)=f( 00 20 02 )=  817.97580713121
 f(185)=f( 00 13 00 )=  412.43621376403
 f(190)=f( 00 11 02 )= -13167.703659685
 f(195)=f( 00 04 00 )= -632.44363424370
 f(200)=f( 00 02 02 )=  25835.201289497
```

8.11 Dynamic Invariant Analysis

Type Code: dia

Required Parameters:

1. IOPT
   = 0 to compute regular invariants.
   = INFILE ≥ 1 to read parameters for mixed invariant from an external file with number INFILE.
   = $-$J (with J an integer from 1 to 9) to read parameters for mixed invariant from the parameter set array associated with the type code *psj*.
2. IPINV
   = 0 to not put out invariants.
   = 1 to put out invariants.
3. IPMAPS
   = 0 to not put out maps.
   = 1 to put out normalizing map $\mathcal{A}$.
   = 2 to put out normal form $\mathcal{N}$.
   = 3 to put out normalizing map and normal form.
4. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
   Note: When ISEND = -1, -2, or -3, the result is the same as when ISEND = $+1$, $+2$, or $+3$, respectively, except that the printing or writing of determinants associated with various "purifying" routines is inhibited. The sizes of these determinants are a measure of proximity to various resonances. See section 7.3.6.
5. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps and invariant polynomials on file IFILE.

When IOPT = INFILE ≥ 1, the file INFILE should contain the values ANX, ANY, and ANT written on one line in free format. When IOPT = $-J < 0$, the contents of the parameter set with index J are as follows:

1. ANX
2. ANY
3. ANT
4. 0 (not used, but must be present)

5. 0 (not used, but must be present)
6. 0 (not used, but must be present)

Example:

```
dyninv    dia
 0 , 1 , 0 , 3 , 0
```

This specifies a command with the user given name *dyninv*. When invoked it puts out the regular invariants at the terminal and on file 12 on the assumption that the current map $\mathcal{M}$ is dynamic and the eigenvalues of its linear part lie on the unit circle and are not resonant.

Description:

Let $\mathcal{M}$ be a dynamic map and assume the eigenvalues of its linear part lie on the unit circle and are not resonant. Then there is a normalizing map $\mathcal{A}$ such that the map given by

$$\mathcal{N} = \mathcal{A}\mathcal{M}\mathcal{A}^{-1}$$

has dynamic normal form. See section 8.9. The "regular" invariants I_x, I_y, and I_τ are given by the relations

$$I_x = \mathcal{A}^{-1}(P_x^2 + X^2),$$
$$I_y = \mathcal{A}^{-1}(P_x^2 + Y^2),$$
$$I_\tau = \mathcal{A}^{-1}(P_\tau^2 + \tau^2).$$

These quantities are nonlinear generalizations of the Courant-Snyder invariants, and satisfy the relations

$$\mathcal{M}I_{x,y,\tau} = I_{x,y,\tau}.$$

The *mixed* invariant I_m is defined by the relation

$$I_m = (ANX)I_x + (ANY)I_y + (ANT)I_\tau.$$

It is also invariant in the nonresonant case, and remains invariant in the resonant case providing the integers (ANX), (ANY), and (ANT) are those *complementary* to the resonance. For example, for the $(2\nu_x - \nu_y)$ resonance, they should have the values $ANX = 1$ and $ANY = 2$.

When a command with type code *dia* is executed, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{A}$, the normalizing map.
2. Buffer 2 contains the normal form map $\mathcal{N}$.
3. If IOPT = 0, buffer 3 contains the identity matrix in its matrix part, and the x invariant polynomial in its array part. Otherwise buffer 3 contains the mixed invariant.

4. If IOPT = 0, buffer 4 contains the identity matrix in its matrix part, and the y invariant polynomial in its array part. Otherwise buffer 4 is empty.

5. If IOPT = 0, buffer 5 contains the identity matrix in its matrix part, and the τ invariant polynomial in its array part. Otherwise buffer 5 is empty.

6. Buffer 6 is empty.

The MaryLie run shown in the Exhibit below illustrates the use of the *dia* command.

```
#comment
 Exhibit 8.11.
 This MaryLie run illustrates the use of the type code dia
 for the small dynamic storage ring of Exhibit 2.6.1.  It does
 three things:

 1. It generates tracking data by tracking a particle for 1000 turns.
 2. It computes the regular invariants.
 3. The value of the "x" invariant is computed for this turn by turn data,
    and written on an external file using a pval command.

#beam
  4.78740000023600
  2807.64374771200
  1.00000000000000
  1.00000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.6094450000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.0110200000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       8.91000000000000
 vcs      sext
  0.200000000000000      -31.2000000000000
 cvty     srfc
  -350000.000000000       502000000.000000
 tadm     tadm
   1.00000000000000       123.000000000000       3.00000000000000
```

```
  1.00000000000000      0.000000000000000E+00
 radm0    radm
  0.000000000000000E+00   1.00000000000000       1.00000000000000
  0.000000000000000E+00
 radm1    radm
  1.00000000000000       1.00000000000000       1.00000000000000
  0.000000000000000E+00
 dnor     dnor
  0.000000000000000E+00   3.00000000000000       3.00000000000000
  1.00000000000000      0.000000000000000E+00
 dia      dia
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
  3.00000000000000      0.000000000000000E+00
 ps1      ps1
  2.000000000000000E-03  0.000000000000000E+00  2.000000000000000E-03
  0.000000000000000E+00  0.000000000000000E+00  4.000000000000000E-04
 fin      end
 mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
 iden     iden
 twsx     twsm
  1.00000000000000       20.0000000000000       1.00000000000000
  2.00000000000000
 twsy     twsm
  2.00000000000000       30.0000000000000       3.00000000000000
  4.00000000000000
 twst     twsm
  3.00000000000000       40.0000000000000       5.00000000000000
  6.00000000000000
 zer      zer
  0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
  0.000000000000000E+00
 rayin    rt
  -1.00000000000000       14.0000000000000      0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
  1000.00000000000       1.00000000000000      0.000000000000000E+00
 gbuf3    gbuf
  2.00000000000000       3.00000000000000
 pval     pval
  0.000000000000000E+00   14.0000000000000       -1.0000000000000
  16.0000000000000
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 half
     1*arc         1*cvty        1*arc
```

```
 statring
     4*arc
 dynring
     2*half
 test
     1*twsx        1*twsy        1*twst
 makedata
     1*ps1         1*rayin       1*track
#lumps
#loops
#labor
    1*fileout
    1*zer
    1*dynring
    1*makedata
    1*dia
    1*gbuf3
    1*mapout
    1*pval
    1*fin

dynamic invariant analysis

x invariant polynomial

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.11748453658825
 f( 12)=f( 10 00 01 )= 0.42577475913485
 f( 13)=f( 02 00 00 )=  8.4804237450446
 f( 16)=f( 01 00 10 )=-1.72765145618715E-02
 f( 25)=f( 00 00 20 )= 8.79902833808772E-06
 f( 27)=f( 00 00 02 )= 0.38576171550066
 f( 28)=f( 30 00 00 )=-5.50882902775623E-02
 f( 29)=f( 21 00 00 )= -2.3923382903153
 f( 32)=f( 20 00 10 )= 1.25118131874185E-03
 f( 33)=f( 20 00 01 )=-0.93340655191617
 f( 34)=f( 12 00 00 )=  8.3750989475502
 f( 37)=f( 11 00 10 )=-4.27756873472307E-02
 f( 38)=f( 11 00 01 )=  6.1899646278779
 f( 39)=f( 10 20 00 )= -5.4241171458410
 f( 40)=f( 10 11 00 )= -19.879141835003
 f( 43)=f( 10 02 00 )=  27.715004904705
 f( 46)=f( 10 00 20 )=-1.27624198129861E-05
 f( 47)=f( 10 00 11 )=-3.59740667659051E-03
 f( 48)=f( 10 00 02 )= -2.5828893512412
 f( 49)=f( 03 00 00 )=  8.3775102508003
 f( 52)=f( 02 00 10 )= 6.04189825706330E-02
 f( 53)=f( 02 00 01 )=  61.034954085844
 f( 54)=f( 01 20 00 )=  57.425214847493
 f( 55)=f( 01 11 00 )= -174.56894481006
 f( 58)=f( 01 02 00 )= -297.71833133322
 f( 61)=f( 01 00 20 )= 1.12596540487675E-04
```

```
f( 62)=f( 01 00 11 )=-9.52896733065691E-02
f( 63)=f( 01 00 02 )=  11.745739173364
f( 66)=f( 00 20 10 )=-5.84939851095901E-02
f( 67)=f( 00 20 01 )= -9.8287495459235
f( 69)=f( 00 11 10 )= 0.17781793738926
f( 70)=f( 00 11 01 )= -36.021918596274
f( 75)=f( 00 02 10 )= 0.30325932059822
f( 76)=f( 00 02 01 )=  50.220862593510
f( 80)=f( 00 00 30 )=-1.86235045719531E-07
f( 81)=f( 00 00 21 )=-5.25988324609915E-05
f( 82)=f( 00 00 12 )=-3.16455555721983E-03
f( 83)=f( 00 00 03 )= -1.9432322221152
f( 84)=f( 40 00 00 )=  1.9132431502147
f( 85)=f( 31 00 00 )=-0.10368800849886
f( 88)=f( 30 00 10 )= 7.50495883010115E-03
f( 89)=f( 30 00 01 )=  12.344132438274
f( 90)=f( 22 00 00 )= -153.08348376971
f( 93)=f( 21 00 10 )= 0.23217193074577
f( 94)=f( 21 00 01 )=  6.8945551533720
f( 95)=f( 20 20 00 )= -151.31003235747
f( 96)=f( 20 11 00 )= -42.752290560201
f( 99)=f( 20 02 00 )= -711.89811709482
f(102)=f( 20 00 20 )=-6.98583672404596E-03
f(103)=f( 20 00 11 )= 5.53255475826037E-03
f(104)=f( 20 00 02 )=  60.197043046966
f(105)=f( 13 00 00 )=  13.320075294500
f(108)=f( 12 00 10 )= 4.45681259236022E-02
f(109)=f( 12 00 01 )= -485.94417845835
f(110)=f( 11 20 00 )= -747.18921802317
f(111)=f( 11 11 00 )=  170.17559872156
f(114)=f( 11 02 00 )= -5184.9697144935
f(117)=f( 11 00 20 )=-1.70678552714363E-04
f(118)=f( 11 00 11 )=  3.8323215550997
f(119)=f( 11 00 02 )=  70.945160683992
f(122)=f( 10 20 10 )= 0.38606649050144
f(123)=f( 10 20 01 )= -408.83768889409
f(125)=f( 10 11 10 )=  6.3247364435349
f(126)=f( 10 11 01 )= -308.47735274358
f(131)=f( 10 02 10 )= -2.2808371710206
f(132)=f( 10 02 01 )= -3267.1326492585
f(136)=f( 10 00 30 )=-4.87739332784781E-07
f(137)=f( 10 00 21 )= 0.21858294596590
f(138)=f( 10 00 12 )=-4.89779715877232E-02
f(139)=f( 10 00 03 )=  134.31486673222
f(140)=f( 04 00 00 )= -5298.5687297520
f(141)=f( 03 10 00 )=-1.35363746789461E-10
f(142)=f( 03 01 00 )= 3.81334639038963E-10
f(143)=f( 03 00 10 )=  28.438522352190
f(144)=f( 03 00 01 )=  68.966599419430
f(145)=f( 02 20 00 )= -7275.8014676874
f(146)=f( 02 11 00 )=  4564.4939409789
f(149)=f( 02 02 00 )= -37734.446026528
f(152)=f( 02 00 20 )=  2.2313597119217
```

```
 f(153)=f( 02 00 11 )= 0.48071759591037
 f(154)=f( 02 00 02 )= -1839.9796519194
 f(157)=f( 01 20 10 )= -1.8746810810045
 f(158)=f( 01 20 01 )= -1240.8034835664
 f(160)=f( 01 11 10 )=  17.320549897559
 f(161)=f( 01 11 01 )=  3098.2374084437
 f(166)=f( 01 02 10 )=  112.74946095488
 f(167)=f( 01 02 01 )= -11269.107393673
 f(171)=f( 01 00 30 )= 0.32785514328880
 f(172)=f( 01 00 21 )=-1.43283339847053E-04
 f(173)=f( 01 00 12 )= 0.14755456095827
 f(174)=f( 01 00 03 )=  71.739716317321
 f(175)=f( 00 40 00 )=  159.81996481884
 f(176)=f( 00 31 00 )= -132.14925277272
 f(179)=f( 00 22 00 )=  91.508031217761
 f(182)=f( 00 20 20 )= 1.14448195880048E-02
 f(183)=f( 00 20 11 )= 0.67874638447693
 f(184)=f( 00 20 02 )= -274.74630889213
 f(185)=f( 00 13 00 )=  719.47692862941
 f(188)=f( 00 11 20 )=-2.10421722967605E-02
 f(189)=f( 00 11 11 )=  7.2477003996116
 f(190)=f( 00 11 02 )= -364.95567290012
 f(195)=f( 00 04 00 )=  4247.4812562524
 f(198)=f( 00 02 20 )=-8.59102881217631E-02
 f(199)=f( 00 02 11 )= -2.7260247497464
 f(200)=f( 00 02 02 )= -3423.7350675560
 f(205)=f( 00 00 40 )=-3.36289716508154E-04
 f(206)=f( 00 00 31 )=-1.76813792248603E-06
 f(207)=f( 00 00 22 )= 0.42679206614881
 f(208)=f( 00 00 13 )=-5.30841978498887E-02
 f(209)=f( 00 00 04 )=  100.26854952693

y invariant polynomial

nonzero elements in generating polynomial are :

 f( 18)=f( 00 20 00 )= 0.48069132382682
 f( 22)=f( 00 02 00 )=  2.0803371112233
 f( 39)=f( 10 20 00 )=  9.8773923319011
 f( 40)=f( 10 11 00 )=  61.807308970686
 f( 43)=f( 10 02 00 )= -42.747403191262
 f( 54)=f( 01 20 00 )= -81.157285992498
 f( 55)=f( 01 11 00 )=  418.42304980328
 f( 58)=f( 01 02 00 )=  351.23270491394
 f( 66)=f( 00 20 10 )= 9.86665797779823E-02
 f( 67)=f( 00 20 01 )=  17.284123911440
 f( 69)=f( 00 11 10 )=-0.25586805276409
 f( 70)=f( 00 11 01 )=  115.37954032215
 f( 75)=f( 00 02 10 )=-0.42700947026779
 f( 76)=f( 00 02 01 )= -74.802274610858
 f( 95)=f( 20 20 00 )=  299.82883160303
 f( 96)=f( 20 11 00 )=  42.986147531379
 f( 99)=f( 20 02 00 )=  1567.5832915854
```

```
 f(107)=f( 12 01 00 )=-1.12593118049752E-10
 f(110)=f( 11 20 00 )=  847.62487744520
 f(111)=f( 11 11 00 )= -112.38088076553
 f(114)=f( 11 02 00 )=  8797.5988098807
 f(122)=f( 10 20 10 )=-0.86624763539945
 f(123)=f( 10 20 01 )=  814.50391442449
 f(125)=f( 10 11 10 )= -13.194453927152
 f(126)=f( 10 11 01 )=  448.70276697049
 f(131)=f( 10 02 10 )=  4.8478290037874
 f(132)=f( 10 02 01 )=  6966.8608308110
 f(142)=f( 03 01 00 )=-8.85478610781911E-10
 f(145)=f( 02 20 00 )=  15628.975501143
 f(146)=f( 02 11 00 )= -5494.2514776672
 f(149)=f( 02 02 00 )=  82716.320262076
 f(157)=f( 01 20 10 )= -2.0452453133466
 f(158)=f( 01 20 01 )=  1354.4382484524
 f(160)=f( 01 11 10 )= -38.691466603194
 f(161)=f( 01 11 01 )= -4628.5673923803
 f(166)=f( 01 02 10 )= -246.69803528274
 f(167)=f( 01 02 01 )=  19096.542371698
 f(175)=f( 00 40 00 )= -292.93813645217
 f(176)=f( 00 31 00 )=  209.56289760343
 f(179)=f( 00 22 00 )= -648.43286671981
 f(182)=f( 00 20 20 )=-1.44776120998400E-02
 f(183)=f( 00 20 11 )= -1.5390518179861
 f(184)=f( 00 20 02 )=  549.80827320942
 f(185)=f( 00 13 00 )= -1012.1744296260
 f(188)=f( 00 11 20 )= 3.95471152571372E-02
 f(189)=f( 00 11 11 )= -17.835951031237
 f(190)=f( 00 11 02 )=  557.42906302379
 f(195)=f( 00 04 00 )= -7330.1711969619
 f(198)=f( 00 02 20 )= 0.18435321881443
 f(199)=f( 00 02 11 )=  6.6607123735332
 f(200)=f( 00 02 02 )=  7233.7790122193

t invariant polynomial

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 2.18664519385818E-05
 f( 12)=f( 10 00 01 )=-4.29338400580191E-02
 f( 13)=f( 02 00 00 )= 0.15522995115863
 f( 16)=f( 01 00 10 )= 0.17133114150351
 f( 25)=f( 00 00 20 )= 4.72756060119122E-02
 f( 27)=f( 00 00 02 )=  21.074688149054
 f( 28)=f( 30 00 00 )= 9.38505935152254E-06
 f( 29)=f( 21 00 00 )=-5.64816354842978E-02
 f( 32)=f( 20 00 10 )=-3.10505293602481E-02
 f( 33)=f( 20 00 01 )=-9.47634631028945E-03
 f( 34)=f( 12 00 00 )=-0.46446715520325
 f( 37)=f( 11 00 10 )=-0.25569192389275
 f( 38)=f( 11 00 01 )= 0.15964014990980
 f( 39)=f( 10 20 00 )=-3.33872109937026E-04
```

```
f( 40)=f( 10 11 00 )= 7.14482802133079E-04
f( 43)=f( 10 02 00 )= 2.35868143239339E-03
f( 46)=f( 10 00 20 )=-1.60538296290295E-04
f( 47)=f( 10 00 11 )=-2.97663433656634E-02
f( 48)=f( 10 00 02 )= 0.30940952080601
f( 49)=f( 03 00 00 )=  1.3565122708977
f( 52)=f( 02 00 10 )= 0.75094402590398
f( 53)=f( 02 00 01 )=  1.1981947803135
f( 54)=f( 01 20 00 )=-0.13884708840807
f( 55)=f( 01 11 00 )=-0.10945233839303
f( 58)=f( 01 02 00 )=-0.29955697692021
f( 61)=f( 01 00 20 )= 1.28566301762307E-03
f( 62)=f( 01 00 11 )=-0.24421772249247
f( 63)=f( 01 00 02 )=-0.91814209627494
f( 66)=f( 00 20 10 )=-7.66244850746732E-02
f( 67)=f( 00 20 01 )= 0.32777178044548
f( 69)=f( 00 11 10 )=-6.04026282851268E-02
f( 70)=f( 00 11 01 )=-0.70142816121125
f( 75)=f( 00 02 10 )=-0.16531422711277
f( 76)=f( 00 02 01 )= -2.3155848889119
f( 80)=f( 00 00 30 )=-1.90378882742967E-06
f( 81)=f( 00 00 21 )=-7.46741072804825E-04
f( 82)=f( 00 00 12 )=-2.59435877824300E-02
f( 83)=f( 00 00 03 )= -50.505001597403
f( 84)=f( 40 00 00 )= 5.01119528778989E-03
f( 85)=f( 31 00 00 )= 0.12422572675419
f( 88)=f( 30 00 10 )= 2.17924719283171E-02
f( 89)=f( 30 00 01 )= 0.18648669416910
f( 90)=f( 22 00 00 )= 0.44149074325185
f( 93)=f( 21 00 10 )= 0.55403530691170
f( 94)=f( 21 00 01 )=-0.89536735553777
f( 95)=f( 20 20 00 )= 6.66844863032831E-02
f( 96)=f( 20 11 00 )=-3.70800225599605E-02
f( 99)=f( 20 02 00 )=-7.01396697442052E-02
f(102)=f( 20 00 20 )= 0.19759499395224
f(103)=f( 20 00 11 )=-3.94574968328468E-02
f(104)=f( 20 00 02 )= -87.961727539884
f(105)=f( 13 00 00 )=-0.43179880915992
f(108)=f( 12 00 10 )= 0.84776269344040
f(109)=f( 12 00 01 )=  67.163501206438
f(110)=f( 11 20 00 )= -3.8038466989529
f(111)=f( 11 11 00 )= -48.468999768741
f(114)=f( 11 02 00 )=  17.137854067554
f(117)=f( 11 00 20 )=-2.42241650324144E-03
f(118)=f( 11 00 11 )=  31.487668261831
f(119)=f( 11 00 02 )= -2.3065218735741
f(122)=f( 10 20 10 )= -2.2254987990427
f(123)=f( 10 20 01 )= -40.364187168156
f(125)=f( 10 11 10 )= -27.507005386115
f(126)=f( 10 11 01 )=  55.931053998004
f(131)=f( 10 02 10 )=  9.6661769172904
f(132)=f( 10 02 01 )=  123.99643901773
f(136)=f( 10 00 30 )=-4.92477071091652E-06
```

```
 f(137)=f( 10 00 21 )=  2.9676153141086
 f(138)=f( 10 00 12 )=-0.13740661579554
 f(139)=f( 10 00 03 )= -479.03124396619
 f(140)=f( 04 00 00 )=  60.796594599549
 f(143)=f( 03 00 10 )=  54.487456242220
 f(144)=f( 03 00 01 )= -20.984271648506
 f(145)=f( 02 20 00 )=  83.878480269184
 f(146)=f( 02 11 00 )= -160.90967504799
 f(149)=f( 02 02 00 )= -369.97865687468
 f(151)=f( 02 01 01 )=-5.77863075056257E-10
 f(152)=f( 02 00 20 )=  9.8886276957637
 f(153)=f( 02 00 11 )=  5.3810013165502
 f(154)=f( 02 00 02 )= -6202.6489871747
 f(157)=f( 01 20 10 )=  47.985390834227
 f(158)=f( 01 20 01 )= -26.836104485562
 f(160)=f( 01 11 10 )= -88.419449469516
 f(161)=f( 01 11 01 )= -1278.1283107756
 f(166)=f( 01 02 10 )= -196.89457620152
 f(167)=f( 01 02 01 )=  828.01216586161
 f(171)=f( 01 00 30 )= -1.9152808746455
 f(172)=f( 01 00 21 )=-1.71737210817384E-03
 f(173)=f( 01 00 12 )=  517.38088687994
 f(174)=f( 01 00 03 )= 4.00114679571856E-02
 f(175)=f( 00 40 00 )= 3.23227630094046E-02
 f(176)=f( 00 31 00 )= 4.34957980119417E-02
 f(179)=f( 00 22 00 )= 0.14109404043753
 f(182)=f( 00 20 20 )= 0.75566006899188
 f(183)=f( 00 20 11 )= -3.8273085986910
 f(184)=f( 00 20 02 )= -434.73799071575
 f(185)=f( 00 13 00 )= 0.14414329091242
 f(188)=f( 00 11 20 )= 5.65141834546327E-02
 f(189)=f( 00 11 11 )= -46.898451778664
 f(190)=f( 00 11 02 )=  109.28152741820
 f(195)=f( 00 04 00 )= 0.20812479298665
 f(198)=f( 00 02 20 )=  3.5942795336489
 f(199)=f( 00 02 11 )=  16.734965447889
 f(200)=f( 00 02 02 )= -1279.7401952595
 f(204)=f( 00 01 03 )=-1.55074789772656E-10
 f(205)=f( 00 00 40 )=-0.27371461005316
 f(206)=f( 00 00 31 )=-1.79824696191326E-05
 f(207)=f( 00 00 22 )=  153.18738767644
 f(208)=f( 00 00 13 )=-9.71041585914343E-02
 f(209)=f( 00 00 04 )=  32226.709610105

matrix for map is :

 1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00
 0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00  0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00 0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 1.00000E+00
```

```
nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.11748453658825
 f( 12)=f( 10 00 01 )= 0.42577475913485
 f( 13)=f( 02 00 00 )=  8.4804237450446
 f( 16)=f( 01 00 10 )=-1.72765145618715E-02
 f( 25)=f( 00 00 20 )= 8.79902833808772E-06
 f( 27)=f( 00 00 02 )= 0.38576171550066
 f( 28)=f( 30 00 00 )=-5.50882902775623E-02
 f( 29)=f( 21 00 00 )= -2.3923382903153
 f( 32)=f( 20 00 10 )= 1.25118131874185E-03
 f( 33)=f( 20 00 01 )=-0.93340655191617
 f( 34)=f( 12 00 00 )=  8.3750989475502
 f( 37)=f( 11 00 10 )=-4.27756873472307E-02
 f( 38)=f( 11 00 01 )=  6.1899646278779
 f( 39)=f( 10 20 00 )= -5.4241171458410
 f( 40)=f( 10 11 00 )= -19.879141835003
 f( 43)=f( 10 02 00 )=  27.715004904705
 f( 46)=f( 10 00 20 )=-1.27624198129861E-05
 f( 47)=f( 10 00 11 )=-3.59740667659051E-03
 f( 48)=f( 10 00 02 )= -2.5828893512412
 f( 49)=f( 03 00 00 )=  8.3775102508003
 f( 52)=f( 02 00 10 )= 6.04189825706330E-02
 f( 53)=f( 02 00 01 )=  61.034954085844
 f( 54)=f( 01 20 00 )=  57.425214847493
 f( 55)=f( 01 11 00 )= -174.56894481006
 f( 58)=f( 01 02 00 )= -297.71833133322
 f( 61)=f( 01 00 20 )= 1.12596540487675E-04
 f( 62)=f( 01 00 11 )=-9.52896733065691E-02
 f( 63)=f( 01 00 02 )=  11.745739173364
 f( 66)=f( 00 20 10 )=-5.84939851095901E-02
 f( 67)=f( 00 20 01 )= -9.8287495459235
 f( 69)=f( 00 11 10 )= 0.17781793738926
 f( 70)=f( 00 11 01 )= -36.021918596274
 f( 75)=f( 00 02 10 )= 0.30325932059822
 f( 76)=f( 00 02 01 )=  50.220862593510
 f( 80)=f( 00 00 30 )=-1.86235045719531E-07
 f( 81)=f( 00 00 21 )=-5.25988324609915E-05
 f( 82)=f( 00 00 12 )=-3.16455555721983E-03
 f( 83)=f( 00 00 03 )= -1.9432322221152
 f( 84)=f( 40 00 00 )=  1.9132431502147
 f( 85)=f( 31 00 00 )=-0.10368800849886
 f( 88)=f( 30 00 10 )= 7.50495883010115E-03
 f( 89)=f( 30 00 01 )=  12.344132438274
 f( 90)=f( 22 00 00 )= -153.08348376971
 f( 93)=f( 21 00 10 )= 0.23217193074577
 f( 94)=f( 21 00 01 )=  6.8945551533720
 f( 95)=f( 20 20 00 )= -151.31003235747
 f( 96)=f( 20 11 00 )= -42.752290560201
 f( 99)=f( 20 02 00 )= -711.89811709482
 f(102)=f( 20 00 20 )=-6.98583672404596E-03
 f(103)=f( 20 00 11 )= 5.53255475826037E-03
 f(104)=f( 20 00 02 )=  60.197043046966
```

```
f(105)=f( 13 00 00 )=  13.320075294500
f(108)=f( 12 00 10 )= 4.45681259236022E-02
f(109)=f( 12 00 01 )= -485.94417845835
f(110)=f( 11 20 00 )= -747.18921802317
f(111)=f( 11 11 00 )=  170.17559872156
f(114)=f( 11 02 00 )= -5184.9697144935
f(117)=f( 11 00 20 )=-1.70678552714363E-04
f(118)=f( 11 00 11 )=  3.8323215550997
f(119)=f( 11 00 02 )=  70.945160683992
f(122)=f( 10 20 10 )= 0.38606649050144
f(123)=f( 10 20 01 )= -408.83768889409
f(125)=f( 10 11 10 )=  6.3247364435349
f(126)=f( 10 11 01 )= -308.47735274358
f(131)=f( 10 02 10 )= -2.2808371710206
f(132)=f( 10 02 01 )= -3267.1326492585
f(136)=f( 10 00 30 )=-4.87739332784781E-07
f(137)=f( 10 00 21 )= 0.21858294596590
f(138)=f( 10 00 12 )=-4.89779715877232E-02
f(139)=f( 10 00 03 )=  134.31486673222
f(140)=f( 04 00 00 )= -5298.5687297520
f(141)=f( 03 10 00 )=-1.35363746789461E-10
f(142)=f( 03 01 00 )= 3.81334639038963E-10
f(143)=f( 03 00 10 )=  28.438522352190
f(144)=f( 03 00 01 )=  68.966599419430
f(145)=f( 02 20 00 )= -7275.8014676874
f(146)=f( 02 11 00 )=  4564.4939409789
f(149)=f( 02 02 00 )= -37734.446026528
f(152)=f( 02 00 20 )=  2.2313597119217
f(153)=f( 02 00 11 )= 0.48071759591037
f(154)=f( 02 00 02 )= -1839.9796519194
f(157)=f( 01 20 10 )= -1.8746810810045
f(158)=f( 01 20 01 )= -1240.8034835664
f(160)=f( 01 11 10 )=  17.320549897559
f(161)=f( 01 11 01 )=  3098.2374084437
f(166)=f( 01 02 10 )=  112.74946095488
f(167)=f( 01 02 01 )= -11269.107393673
f(171)=f( 01 00 30 )= 0.32785514328880
f(172)=f( 01 00 21 )=-1.43283339847053E-04
f(173)=f( 01 00 12 )= 0.14755456095827
f(174)=f( 01 00 03 )=  71.739716317321
f(175)=f( 00 40 00 )=  159.81996481884
f(176)=f( 00 31 00 )= -132.14925277272
f(179)=f( 00 22 00 )=  91.508031217761
f(182)=f( 00 20 20 )= 1.14448195880048E-02
f(183)=f( 00 20 11 )= 0.67874638447693
f(184)=f( 00 20 02 )= -274.74630889213
f(185)=f( 00 13 00 )=  719.47692862941
f(188)=f( 00 11 20 )=-2.10421722967605E-02
f(189)=f( 00 11 11 )=  7.2477003996116
f(190)=f( 00 11 02 )= -364.95567290012
f(195)=f( 00 04 00 )=  4247.4812562524
f(198)=f( 00 02 20 )=-8.59102881217631E-02
f(199)=f( 00 02 11 )= -2.7260247497464
```

```
 f(200)=f( 00 02 02 )= -3423.7350675560
 f(205)=f( 00 00 40 )=-3.36289716508154E-04
 f(206)=f( 00 00 31 )=-1.76813792248603E-06
 f(207)=f( 00 00 22 )= 0.42679206614881
 f(208)=f( 00 00 13 )=-5.30841978498887E-02
 f(209)=f( 00 00 04 )=  100.26854952693

 5001 ray(s) read in from file  14

value(s) of polymomial written on file          16
```

Figures 8.11.1 through 8.11.3 below show turn-by-turn values of I_x computed through second, third, and fourth orders. Evidently I_x through second order (the usual Courant-Snyder invariant but with synchro-betatron coupling) shows considerable variation, and including higher order terms provides significant improvements in making the invariant more nearly constant. For example, in this case, I_x through second order varies by $\pm 15\%$. By contrast, I_x through third and fourth orders vary by $\pm .5\%$ and $\pm .05\%$ respectively. (Note the different scales.)

Figure 8.11.1: Contents of file 16, turn by turn values of I_x through second order.

Figure 8.11.2: Contents of file 16, turn by turn values of I_x through third order.

Figure 8.11.3: Contents of file 16, turn by turn values of I_x through fourth order.

8.12 Compute Power of Static Normal Form

Type Code: psnf

Required Parameters:

1. JINOPT (controls meaning of the second parameter, POW)
   = 1 if POW is to be interpreted as a power.
   = 2 if POW is to be interpreted as a file number or the number of a parameter set array.
2. POW
   = power when JINOPT = 1.
   = NPOWFILE [Number of file from which power(s) is (are) to be read when JINOPT = 2].
   Note: When NPOWFILE = −J (with J an integer from 1 to 9), powers are taken from the parameter set array associated with the type code *psj*. In this case, zero parameter values are ignored.
3. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 to 9) to use map in storage location NMAP.
4. JOUTOPT
   = 0 to place resulting map in a location specified by the parameter MAPOUT. If several maps are computed, only the last is placed and the rest are lost.
   = NOUTFILE to write resulting map(s) on file NOUTFILE. In this case the parameter MAPOUT is ignored.
5. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting map in storage location KMAP.
   = 0 to make the resulting map the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting map in buffer number KBUF.

Example:

```
powrs    psnf
  1 , 2.5 , 0, 16 , 0
```

This specifies a command with the user given name *powrs*. When invoked it assumes that the current transfer map is in static normal form and raises it to a power (which can be any real number) that in this case is 2.5.

Description:

Suppose $\mathcal{N}$ is a map that is known to be in static normal form. This command makes it possible to compute $\mathcal{N}^s$ where s is any real number. This command can be useful for producing a filamented phase-space distribution since the normal form can have anharmonic terms. See also section 8.29.

NOTE WELL:

The file NOUTFILE is brought to its end before writing, so that transfer maps (if any) already on file NOUTFILE are not overwritten.

8.13 Compute Power of Dynamic Normal Form

Type Code: pdnf

Required Parameters:

1. JINOPT (controls meaning of the second parameter, POW)
   = 1 if POW is to be interpreted as a power.
   = 2 if POW is to be interpreted as a file number or the number of a parameter set array.
2. POW = NPOWFILE [Number of file from which power(s) is (are) to be read when JINOPT = 2].
   Note: If NPOWFILE = −J (with J an integer from 1 to 9), powers are taken from the parameter set array associated with the type code *psj*. In this case, zero parameter values are ignored.
3. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 to 9) to use map in storage location NMAP.
4. JOUTOPT
   = 0 to place resulting map in a location specified by the parameter MAPOUT. If several maps are computed, only the last is placed and the rest are lost.
   = NOUTFILE to write resulting map(s) on file NOUTFILE. In this case the parameter MAPOUT is ignored.
5. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting map in storage location KMAP.
   = 0 to make the resulting map the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting map in buffer number KBUF.

Example:

```
powrd    pdnf
  1, 2.5 , 0 , 16 , 0
```

This specifies a command with the user given name *powrd*. When invoked it assumes that the current transfer map is in dynamic normal form and raises it to a power (which can be any real number) that in this case is 2.5.

Description:

Suppose $\mathcal{N}$ is a map that is known to be in dynamic normal form. This command makes it possible to compute $\mathcal{N}^s$ where s is any real number. This command can be useful for producing a filamented phase-space distribution since the normal form can have anharmonic terms. See also section 8.29.

NOTE WELL:

The file NOUTFILE is brought to its end before writing, so that transfer maps (if any) already on file NOUTFILE are not overwritten.

8.14 Get Buffer Contents

Type Code: gbuf

Required Parameters:

1. IOPT
   = 1 to concatenate existing map with the map in NBUF.
   = 2 to replace the existing map with the map in NBUF.
2. NBUF (an integer from 1 to 6).

Example:

```
getbuf2    gbuf
1, 2
```

This specifies a command with the user given name *getbuf2*. When invoked it concatenates the current map with the map in buffer 2 and makes the result the current map.

Description:

Several analysis commands fill various buffers with various maps. These maps are then available for further use by employing a command with type code *gbuf*.

8.15 Fourier Analyze Static Map

Type Code: fasm

Not yet available.

8.16 Fourier Analyze Dynamic Map

Type Code: fadm

Not yet available.

8.17 Apply Map to a Function or Moments

Type Code: amap

Required Parameters:

1. JOB
   = 0 to act on a function.
   = 1 to act on initial moments through order 4 and to compute transformed moments through 2nd moments.
   = 2 to act on initial moments through order 4 and to compute transformed moments through 4th moments. (This requires more execution time.)
2. ISEND
   = 0 to do nothing.
   = 1 to write transformed function or moments at the terminal.
   = 2 to write transformed function or moments on file 12.
   = 3 to write transformed function or moments at the terminal and on file 12.
3. NINFILE
   = −NMAP to get function or moments from storage location NMAP (see section 7.17).
   = INFILE (file number from which function or moments are to be read).
4. NREWIND
   = 0 to read (from file INFILE) in its current position.
   = 1 to first rewind file unit INFILE, then read.
5. NSKIP
   Number of data sets (functions or moment sets) to be skipped in file INFILE (starting from the current position) before a read actually occurs.
6. NOUTFILE
   = 0 to not write out transformed function or transformed moments.
   = NOUTFILE to write transformed function or transformed moment set on file NOUTFILE.

Example 1:

```
amapf    amap
0 , 1 , -3 , 0 , 0 , 0
```

This specifies a command with the user given name *amapf*. When it is invoked the current transfer map $\mathcal{M}$ acts on the function in storage location 3 and writes the transformed function at the terminal.

Example 2:

```
amapm    amap
2 , 1 , -3 , 0 , 0 , 0
```

This specifies a command with the user given name *amapm*. When it is invoked the current transfer map $\mathcal{M}$ acts on the set of moments in storage location 3 and writes the transformed moments at the terminal.

Note: When *amap* takes in moments from a "map", it uses the array part of the map and ignores its matrix part. Note that the zeroth entry $P(0)$ must have the value 1. See section 4.4.3.

Description:

Suppose JOB = 0. Let $f(z)$ be a polynomial function gotten from a storage location or read in from an external file, and let $\mathcal{M}$ be the current transfer map. Then this command computes the transformed function $g(z)$ given by

$$g = \mathcal{M}f. \tag{8.17.1}$$

The case of a map acting on moments (JOB = 1 or 2) requires more explanation. Consider the following gedanken exercise: Suppose one is given a collection of particles distributed in phase space. Their coordinates could be stored, for instance, in the initial conditions buffer. For this distribution one could find the moments $\langle 1\rangle$, $\langle z_a\rangle$, $\langle z_a z_b\rangle$, $\langle z_a z_b z_c\rangle$, $\langle z_a z_b z_c z_d\rangle$. This could be done, for example, by use of a *bgen* command with JOB = 0. See section 8.34. Next, supose $\mathcal{M}$ is the transfer map describing the system under study. This map could be applied to all the particles in the phase-space distribution, and this transformed distribution could be used to compute transformed moments. In this way a set of original moments is sent, under the action of $\mathcal{M}$, to a set of transformed moments.

With this background in mind, one may wonder if there is a way to compute the action of $\mathcal{M}$ on moments without tracking (computing the effect of $\mathcal{M}$ on) thousands or millions of particles, which can be computationally intensive. The answer is yes. Let $P_\alpha(z)$ with $\alpha = 0, 1, 2, \cdots 209$ denote a basis monomial as in section 14.1. Given a phase-space distribution function $f(z)$, one can define associated moments m_α by the rule

$$m_\alpha = \int d^6 z P_\alpha(z) f(z). \tag{8.17.2}$$

For example, we have the definitions

$$m_1 = \int d^6 z P_1(z) f(z) = \int d^6 z (z_1) f(z) = \langle z_1\rangle, \tag{8.17.3}$$

$$m_8 = \int d^6 z P_8(z) f(z) = \int d^6 z (z_1 z_2) f(z) = \langle z_1 z_2\rangle, \text{ etc.} \tag{8.17.4}$$

Also, for convenience, the phase-space distribution is normalized to have an integral of one,

$$m_0 = \int d^6 z P_0(z) f(z) dz = \int d^6 z (1) f(z) = \langle 1\rangle = 1. \tag{8.17.5}$$

Under the action of $\mathcal{M}$ the phase-space distribution function is sent to the *transformed* distribution $f^{\rm tr}$ with

$$f^{\rm tr}(z) = f(\mathcal{M}^{-1}z). \tag{8.17.6}$$

This result is just Liouville's theorem. Correspondingly, the moments m_α are sent to trasformed moments $m_\alpha^{\rm tr}$ with

$$m_\alpha^{\rm tr} = \int d^6 z P_\alpha(z) f^{\rm tr}(z). \tag{8.17.7}$$

Combining (17.6) and (17.7) gives the result

$$m_\alpha^{\rm tr} = \int d^6 z P_\alpha(z) f(\mathcal{M}^{-1}z) = \int d^6 z' P_\alpha(\mathcal{M}z') f(z'). \tag{8.17.8}$$

Here we have made the change of variables

$$z' = \mathcal{M}^{-1}z, \tag{8.17.9}$$

which has unit Jacobian because $\mathcal{M}$ is a symplectic map. Also, since the basis monomials form a complete set, there are relations of the form

$$P_\alpha(\mathcal{M}z') = \sum_{\alpha'} \mathcal{D}_{\alpha\alpha'}(\mathcal{M}) P_{\alpha'}(z'). \tag{8.17.10}$$

Finally, insert (17.10) into (17.8) to obtain the result

$$m_\alpha^{\rm tr} = \int d^6 z' \sum_{\alpha'} \mathcal{D}_{\alpha\alpha'}(\mathcal{M}) P_{\alpha'}(z') f(z') \tag{8.17.11}$$

or

$$m_\alpha^{\rm tr} = \sum_{\alpha'} \mathcal{D}_{\alpha\alpha'}(\mathcal{M}) m_{\alpha'}. \tag{8.17.12}$$

We see that moments transform like the components of a vector, and their transformation can be found without tracking provided one knows the matrix $\mathcal{D}$. This is what is done in MaryLie when a map $\mathcal{M}$ is made to act on moments by invoking an *amap* command with JOB = 1 or 2.

Since MaryLie 3.0 is a third order code, its internal computational machinery has been restricted to work with polynomials of degree 4 and lower. Correspondingly, there is some truncation in the entries of $\mathcal{D}$ as listed below.

JOB = 1 or 2

Required Initial Moments: $\langle 1\rangle, \langle z_a\rangle, \langle z_a z_b\rangle, \langle z_a z_b z_c\rangle, \langle z_a z_b z_c z_d\rangle$.

Final Moment		Map Employed
$\langle z_a\rangle$		$e^{:f_2:}[1+ :f_3: +(\frac{:f_3:^2}{2!}+ :f_4:) + (\frac{:f_3:^3}{3!}+ :f_3::f_4:)]$
$\langle z_a z_b\rangle$	(1)	$e^{:f_2:}[1+ :f_3: +(\frac{:f_3:^2}{2!}+ :f_4:)]$
$\langle z_a z_b z_c\rangle$	(2)	$e^{:f_2:}[1+ :f_3:]$
$\langle z_a z_b z_c z_d\rangle$	(3)	$e^{:f_2:}$

Remarks:

(1) Effects of $:f_3:^3$ and $:f_3::f_4:$ are lost

(2) Effects of $:f_3:^2$ and $:f_4:$ are lost.

(3) All effects of $:f_3:$ and $:f_4:$ are lost.

When *amap* is invoked, the contents of buffers 1 through 6 are as follows: Buffers 3 through 6 are all empty. If JOB = 0, buffer 1 contains the identity as its matrix part, and the transformed function as its array part. Buffer 2 is empty. If JOB > 0, buffer 1 contains the transformed second moments as its matrix part, and all the transformed moments as its array part. The content of NOUTFILE, if written, is the same as buffer 1. Buffer 2 contains in its matrix part a matrix whose diagonal entries are the transformed rms moments $\sqrt{\langle z_a z_a \rangle}$ and whose off-diagonal entries are zero. The array part of buffer 2 is empty.

The Exhibit below illustrates the use of *amap* to cause the current transfer map $\mathcal{M}$ to act on a function. For examples of the use of *amap* to cause $\mathcal{M}$ to act on moments, see sections 8.37 and 10.12.

```
#comment
 Exhibit 8.17.
 This MaryLie run illustrates the use of the type code
 amap for the small static storage ring of Exhibit 2.5.1.
 It uses sia to compute I_x, and then uses amap to verify
 the relation
   script M I_x = I_x.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       2.72000000000000       1.00000000000000
   1.00000000000000
 hdq      quad
  0.500000000000000      -1.92000000000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.500000000000000      -1.62000000000000
 vcs      sext
```

```
  0.500000000000000       3.34000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 track    rt
   13.0000000000000       14.0000000000000       5.00000000000000
   500.000000000000       1.00000000000000      0.000000000000000E+00
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 snor     snor
  0.000000000000000E+00   3.00000000000000       3.00000000000000
   1.00000000000000      0.000000000000000E+00
 sia      sia
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
   3.00000000000000      0.000000000000000E+00
 amap     amap
  0.000000000000000E+00   1.00000000000000      -3.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 gbuf3    gbuf
   2.00000000000000       3.00000000000000
 stm1     stm
   1.00000000000000
 gtm1     gtm
   2.00000000000000       1.00000000000000
 stm3     stm
   3.00000000000000
 iden     iden
 zer      zer
  0.000000000000000E+00  5.000000000000000E-10  5.000000000000000E-10
  0.000000000000000E+00
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
#lumps
#loops
```

```
#labor
    1*fileout
    1*zer
    1*ring
    1*stm1
    1*sia
    1*gbuf3
    1*stm3
    1*gtm1
    1*amap
    1*fin
map stored in location   1

static invariant analysis

x invariant polynomial

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.28283741725472
 f(  8)=f( 11 00 00 )=  1.8119521227272
 f( 12)=f( 10 00 01 )=  1.4555083997013
 f( 13)=f( 02 00 00 )=  6.4375945779623
 f( 17)=f( 01 00 01 )=  1.7697319603908
 f( 27)=f( 00 00 02 )=  2.4641432149979
 f( 28)=f( 30 00 00 )= 8.14862086573107E-02
 f( 29)=f( 21 00 00 )=  1.7909382522170
 f( 33)=f( 20 00 01 )= -1.9439421600706
 f( 34)=f( 12 00 00 )=-0.69316455416935
 f( 38)=f( 11 00 01 )=  15.321043502166
 f( 39)=f( 10 20 00 )=  1.1234824876277
 f( 40)=f( 10 11 00 )= -11.278066755118
 f( 43)=f( 10 02 00 )=  28.683783323401
 f( 48)=f( 10 00 02 )= -24.790310322533
 f( 49)=f( 03 00 00 )= -39.498206914693
 f( 53)=f( 02 00 01 )=  97.662016671074
 f( 54)=f( 01 20 00 )=  2.2361389163827
 f( 55)=f( 01 11 00 )= -34.437321940070
 f( 58)=f( 01 02 00 )=  45.847633132341
 f( 63)=f( 01 00 02 )= -70.504642686509
 f( 67)=f( 00 20 01 )=  3.4481379195691
 f( 70)=f( 00 11 01 )= -29.709584450401
 f( 76)=f( 00 02 01 )=  92.634094832153
 f( 83)=f( 00 00 03 )= -34.140122269086
 f( 84)=f( 40 00 00 )= 0.26625964929444
 f( 85)=f( 31 00 00 )= 0.47460012884628
 f( 89)=f( 30 00 01 )= 0.45302042859664
 f( 90)=f( 22 00 00 )= -5.8313785812592
 f( 94)=f( 21 00 01 )= -22.489395096546
 f( 95)=f( 20 20 00 )= -133.94768609570
 f( 96)=f( 20 11 00 )=  716.49307547164
 f( 99)=f( 20 02 00 )=  774.63973894177
 f(104)=f( 20 00 02 )=  15.411106871683
```

```
 f(105)=f( 13 00 00 )=  24.170787005918
 f(109)=f( 12 00 01 )= -110.98043731593
 f(110)=f( 11 20 00 )= -729.49630673371
 f(111)=f( 11 11 00 )= -3093.2302592337
 f(114)=f( 11 02 00 )=  26125.403574480
 f(119)=f( 11 00 02 )= -166.56546917394
 f(123)=f( 10 20 01 )= -765.01365459675
 f(126)=f( 10 11 01 )=  7138.4652828102
 f(132)=f( 10 02 01 )= -5514.3389878372
 f(139)=f( 10 00 03 )=  198.38331770005
 f(140)=f( 04 00 00 )=  265.28967521583
 f(144)=f( 03 00 01 )= -598.26336904547
 f(145)=f( 02 20 00 )=  756.61139263247
 f(146)=f( 02 11 00 )= -26007.556806726
 f(149)=f( 02 02 00 )=  64393.054340637
 f(154)=f( 02 00 02 )=  336.51063798049
 f(158)=f( 01 20 01 )= -3530.0340246697
 f(161)=f( 01 11 01 )=  9875.9553103357
 f(167)=f( 01 02 01 )=  49139.325361611
 f(174)=f( 01 00 03 )= -542.28702501823
 f(175)=f( 00 40 00 )=  1.2469479054074
 f(176)=f( 00 31 00 )= -22.724591921870
 f(179)=f( 00 22 00 )=  178.46754108576
 f(184)=f( 00 20 02 )= -798.84747137678
 f(185)=f( 00 13 00 )= -582.86899632005
 f(190)=f( 00 11 02 )=  12419.528107262
 f(195)=f( 00 04 00 )=  877.06223497910
 f(200)=f( 00 02 02 )= -22602.667052125
 f(209)=f( 00 00 04 )=  601.98597677505

y invariant polynomial

nonzero elements in generating polynomial are :

 f( 18)=f( 00 20 00 )= 0.28222227408050
 f( 19)=f( 00 11 00 )= -1.7421650477668
 f( 22)=f( 00 02 00 )=  6.2319133709247
 f( 39)=f( 10 20 00 )= 0.53664356643599
 f( 40)=f( 10 11 00 )=  2.5453610401654
 f( 43)=f( 10 02 00 )= -19.706225365361
 f( 54)=f( 01 20 00 )= -2.6143765122715
 f( 55)=f( 01 11 00 )=  15.887396574912
 f( 58)=f( 01 02 00 )=  8.6929156992971
 f( 67)=f( 00 20 01 )=  6.1183153892148
 f( 70)=f( 00 11 01 )=  4.5958685600997
 f( 76)=f( 00 02 01 )= -149.28726800044
 f( 95)=f( 20 20 00 )=  137.21682431703
 f( 96)=f( 20 11 00 )= -728.58802024760
 f( 99)=f( 20 02 00 )= -737.95782611400
 f(110)=f( 11 20 00 )=  740.24779668774
 f(111)=f( 11 11 00 )=  3047.6089348420
 f(114)=f( 11 02 00 )= -25879.757217720
 f(123)=f( 10 20 01 )=  780.09330458197
```

```
 f(126)=f( 10 11 01 )= -7283.0744155013
 f(132)=f( 10 02 01 )=  5985.4133761219
 f(145)=f( 02 20 00 )= -681.33686819601
 f(146)=f( 02 11 00 )=  25448.466733518
 f(149)=f( 02 02 00 )= -63197.894117966
 f(158)=f( 01 20 01 )=  3510.3583312332
 f(161)=f( 01 11 01 )= -9585.0333015133
 f(167)=f( 01 02 01 )= -49371.953367217
 f(175)=f( 00 40 00 )= 0.92791989651083
 f(176)=f( 00 31 00 )=  3.8316189898871
 f(179)=f( 00 22 00 )= -106.96761937402
 f(184)=f( 00 20 02 )=  817.97580713121
 f(185)=f( 00 13 00 )=  412.43621376403
 f(190)=f( 00 11 02 )= -13167.703659685
 f(195)=f( 00 04 00 )= -632.44363424370
 f(200)=f( 00 02 02 )=  25835.201289497
map stored in location   3
map gotten from location   1
map gotten from location   3

 transformed function

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 0.28283741725472
 f(  8)=f( 11 00 00 )=  1.8119521227272
 f( 12)=f( 10 00 01 )=  1.4555083997013
 f( 13)=f( 02 00 00 )=  6.4375945779624
 f( 17)=f( 01 00 01 )=  1.7697319603908
 f( 27)=f( 00 00 02 )=  2.4641432149979
 f( 28)=f( 30 00 00 )= 8.14862086573105E-02
 f( 29)=f( 21 00 00 )=  1.7909382522170
 f( 33)=f( 20 00 01 )= -1.9439421600706
 f( 34)=f( 12 00 00 )=-0.69316455416941
 f( 38)=f( 11 00 01 )=  15.321043502166
 f( 39)=f( 10 20 00 )=  1.1234824876277
 f( 40)=f( 10 11 00 )= -11.278066755118
 f( 43)=f( 10 02 00 )=  28.683783323402
 f( 48)=f( 10 00 02 )= -24.790310322533
 f( 49)=f( 03 00 00 )= -39.498206914693
 f( 53)=f( 02 00 01 )=  97.662016671074
 f( 54)=f( 01 20 00 )=  2.2361389163827
 f( 55)=f( 01 11 00 )= -34.437321940070
 f( 58)=f( 01 02 00 )=  45.847633132341
 f( 63)=f( 01 00 02 )= -70.504642686509
 f( 67)=f( 00 20 01 )=  3.4481379195691
 f( 70)=f( 00 11 01 )= -29.709584450401
 f( 76)=f( 00 02 01 )=  92.634094832153
 f( 83)=f( 00 00 03 )= -34.140122269086
 f( 84)=f( 40 00 00 )= 0.26625964929447
 f( 85)=f( 31 00 00 )= 0.47460012884679
 f( 89)=f( 30 00 01 )= 0.45302042859695
 f( 90)=f( 22 00 00 )= -5.8313785812583
```

```
 f( 94)=f( 21 00 01 )= -22.489395096541
 f( 95)=f( 20 20 00 )= -133.94768609570
 f( 96)=f( 20 11 00 )=  716.49307547149
 f( 99)=f( 20 02 00 )=  774.63973894216
 f(104)=f( 20 00 02 )=  15.411106871684
 f(105)=f( 13 00 00 )=  24.170787005911
 f(109)=f( 12 00 01 )= -110.98043731592
 f(110)=f( 11 20 00 )= -729.49630673357
 f(111)=f( 11 11 00 )= -3093.2302592353
 f(114)=f( 11 02 00 )=  26125.403574482
 f(119)=f( 11 00 02 )= -166.56546917392
 f(123)=f( 10 20 01 )= -765.01365459680
 f(126)=f( 10 11 01 )=  7138.4652828097
 f(132)=f( 10 02 01 )= -5514.3389878349
 f(139)=f( 10 00 03 )=  198.38331770005
 f(140)=f( 04 00 00 )=  265.28967521582
 f(144)=f( 03 00 01 )= -598.26336904547
 f(145)=f( 02 20 00 )=  756.61139263289
 f(146)=f( 02 11 00 )= -26007.556806728
 f(149)=f( 02 02 00 )=  64393.054340634
 f(154)=f( 02 00 02 )=  336.51063798053
 f(158)=f( 01 20 01 )= -3530.0340246695
 f(161)=f( 01 11 01 )=  9875.9553103310
 f(167)=f( 01 02 01 )=  49139.325361620
 f(174)=f( 01 00 03 )= -542.28702501822
 f(175)=f( 00 40 00 )=  1.2469479054074
 f(176)=f( 00 31 00 )= -22.724591921871
 f(179)=f( 00 22 00 )=  178.46754108576
 f(184)=f( 00 20 02 )= -798.84747137692
 f(185)=f( 00 13 00 )= -582.86899632005
 f(190)=f( 00 11 02 )=  12419.528107262
 f(195)=f( 00 04 00 )=  877.06223497910
 f(200)=f( 00 02 02 )= -22602.667052123
 f(209)=f( 00 00 04 )=  601.98597677505

end of MARYLIE run
```

8.18 Multiply a Polynomial by a Scalar

Type Code: smul

Required Parameters:

1. SCALAR
2. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 to 9) to use polynomial in storage location NMAP.
3. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting polynomial in storage location KMAP.
   = 0 to make the resulting polynomial the polynomial for the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting polynomial in buffer number KBUF.

Example:

```
stimesp    smul
  3.2 , 1 , 2
```

This specifies a command with the user given name *stimesp*. When invoked, it multiplies the polynomial associated with the map in storage location 1 by the number (3.2), and puts the result in storage location 2.

Description:

This command makes it possible to multiply a polynomial f by a scalar λ.

8.19 Add Two Polynomials

Type Code: padd

Required Parameters:

1. SCAL1
2. SCAL2
3. MAP1IN
   = 0 to use current transfer map.
   = NMAP1 (with NMAP1 an integer from 1 to 9) to use polynomial in storage location NMAP1.
4. MAP2IN
   = 0 to use current transfer map.
   = NMAP2 (with NMAP2 an integer from 1 to 9) to use polynomial in storage location NMAP2.
5. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting polynomial in storage location KMAP.
   = 0 to make the resulting polynomial the polynomial for the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting polynomial in buffer number KBUF.

Example:

```
 p1+p2    padd
1 , 1 , 1 , 2 , 0
```

This specifies a command with the user given name *p1+p2*. When invoked, it takes the polynomials stored in location 1 and 2, adds them, and makes the result the polynomial entry in the current map.

Description:

Given two scalars λ, μ and two polymonials f, g this command makes it possible to compute and store the polynomial $(\lambda f + \mu g)$.

8.20 Multiply Two Polynomials

Type Code: pmul

Required Parameters:

1. MAP1IN
   = 0 to use current transfer map.
   = NMAP1 (with NMAP1 an integer from 1 to 9) to use polynomial in storage location NMAP1.
2. MAP2IN
   = 0 to use current transfer map.
   = NMAP2 (with NMAP2 an integer from 1 to 9) to use polynomial in storage location NMAP2.
3. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting polynomial in storage location KMAP.
   = 0 to make the resulting polynomial the polynomial for the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting polynomial in buffer number KBUF.

Example:

```
p1xp2     pmul
   1 ,  2 ,  0
```

This specifies a command with user given name *p1xp2*. When invoked, it takes the polynomials stored in locations 1 and 2, multiplies them, and makes the result the polynomial entry in the current map.

Description:

Given two polynomials f, g this command makes it possible to compute and store the polynomial fg.

8.21 Poisson Bracket Two Polynomials

Type Code: pb

Required Parameters:

1. MAP1IN
   = 0 to use current transfer map.
   = NMAP1 (with NMAP1 an integer from 1 to 9) to use polynomial in storage location NMAP1.
2. MAP2IN
   = 0 to use current transfer map.
   = NMAP2 (with NMAP2 an integer from 1 to 9) to use polynomial in storage location NMAP2.
3. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting polynomial in storage location KMAP.
   = 0 to make the resulting polynomial the polynomial for the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting polynomial in buffer number KBUF.

Example:

```
[p1p2]      pb
   1 ,  2 ,  0
```

This specifies a command with user given name *[p1p2]*. When invoked it takes the polynomials stored in locations 1 and 2, forms their Poisson bracket, and makes the result the polynomial in the current map.

Description:

Given two polynomials f, g this command makes it possible to compute and store the polynomial $[f, g]$.

8.22 Polar Decomposition of a Map

Type Code: pold

Required Parameters:

1. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 to 9) to use map in storage location NMAP.
2. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
3. IDATA
   = 0 to not put out analysis data.
   = 1 to put out eigenvalues of $\mathcal{P}$ as a matrix.
   = 2 to put out eigenvectors of $\mathcal{P}$ as a matrix.
   = 3 to put out eigenvalues and eigenvectors of $\mathcal{P}$.
4. IPMAPS
   = 0 to not put out maps.
   = 1 to put out $\mathcal{O}$.
   = 2 to put out $\mathcal{P}$.
   = 3 to put out $\mathcal{O}$ and $\mathcal{P}$.
5. IWMAPS
   = 0 to not write out maps.
   = IFILE to write maps on file IFILE.

Example:

```
polar     pold
 1 , 3 , 1 , 2 , 0
```

This specifies a command with the user given name *polar*. When invoked it takes the map from storage location 1, polar decomposes it, writes the results at the terminal and on file 12, and puts out the map $\mathcal{P}$.

Description:

The incoming map, call it $\mathcal{M}$, is decomposed by a command with the type code *pold* into the product $\mathcal{M} = e^{:f_2^c:}\, e^{:f_2^a:}\, e^{:f_3:}\, e^{:f_4:}$. When this is done, the following maps are put in the following buffers:

1. Buffer 1 contains $\mathcal{O} = e^{:f_2^c:}$, the "orthogonal" part of the map. Specifically, it contains the matrix associated with $\mathcal{O}$ in its matrix part, and f_2^c in its array part.

2. Buffer 2 contains $\mathcal{P} = e^{:f_2^a:}$, the positive definite symmetric part of the map. Specifically, it contains the matrix associated with $\mathcal{P}$ in its matrix part, and f_2^a in its array part.

3. Buffer 3 contains $\mathcal{C} = e^{:f_3:}\, e^{:f_4:}$, the nonlinear part of the map.

4. Buffer 4 contains $\mathcal{D}$, the eigenvalues of $\mathcal{P}$ written as a map. That is, the matrix part of $\mathcal{D}$ is a diagonal matrix of eigenvalues, and the polynomial array part of $\mathcal{D}$ is empty.

5. Buffer 5 contains $\mathcal{E}$, the eigenvectors of $\mathcal{P}$ written as a map. That is, the matrix part of $\mathcal{E}$ is the matrix of eigenvectors of the matrix part of $\mathcal{P}$, the polynomial array part of $\mathcal{E}$ is empty.

6. Buffer 6 is empty.

Theory:

Any symplectic matrix M can be written uniquely in the form

$$M = PO \tag{8.22.1}$$

where P is symplectic and positive-definite symmetric, and O is symplectic and orthogonal. This is called orthogonal polar decomposition. The matrices P and O, in turn, can be written in form

$$P = \exp(JS^a), \tag{8.22.2}$$

$$O = \exp(JS^c). \tag{8.22.3}$$

Here S^a is a symmetric matrix that *anticommutes* with J,

$$JS^a + S^aJ = 0, \tag{8.22.4}$$

and S^c is a symmetric matrix that *commutes* with J,

$$JS^c - S^cJ = 0. \tag{8.22.5}$$

Finally, the polynomial f_2^a and f_2^c are defined by the relations

$$f_2^a = -(1/2)\sum_{de} S^a_{de} z_d z_e, \tag{8.22.6}$$

$$f_2^c = -(1/2)\sum_{de} S^c_{de} z_d z_e. \tag{8.22.7}$$

8.23 Evaluate a Polynomial

Type Code: pval

Required Parameters:

1. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 through 9) to use map in storage location NMAP.
   = −NMAP (with NMAP an integer from 1 through 6) to use the map in buffer number NMAP.
2. IDATAIN
   = 0 to use data in the initial conditions buffer.
   = ICFILE $\geq$ 1 to read phase space coordinates from an external file with number ICFILE.
   = −K (with K an integer from 1 to 9) to read phase space coordinates from the parameter set array associated with the type code *psk*.
3. IANS
   = -1 to not write results.
   = 0 to write results into array part of current map.
   = NBUF (an integer from 1 through 6) to write results into array part of buffer number NBUF.
4. IWNUM
   = 0 to not write out numerical data.
   = JFILE to write numerical data on file JFILE.

Example:

```
valofp     pval
 0 , 0 , 1 , 20
```

This specifies a command with the user given name *valofp*. When invoked it finds the value(s) of the polynomial associated with the current transfer map for the phase-space point(s) in the initial conditions buffer. The values are written into buffer 1 and on file 20.

Description:

A command with type code *pval* can be used to evaluate a polynomial function of the phase-space variables at any point in phase space. Such a command can be used, for example, to evaluate invariants. (See sections 8.10 and 8.11.) For convenience in plotting, the lines in file JFILE are numbered. Thus, if the initial conditions buffer contains turn-by-turn phase-space data, the file JFILE will contain the turn number in its first column and the values

val(0) through val(4) in the remaining 5 columns. Here val(0) is the value of the constant term in the polynomial, val(1) is the value of the linear part of the polynomial plus that of the constant part, etc. Thus val(4) is the cumulative value of all terms through fourth order.

The quantities val(0) through val(4) are also available to be used by fitting and optimization routines. See sections 9.7 through 9.9. In this case it is assumed there is only one phase-space point (the last) at which the polynomial is being evaluated. The quantities val(0) through val(4) are put in array locations 0 through 4 of the current transfer map or of buffer number NBUF.

NOTE WELL:

The initial conditions buffer cannot be empty if a command with the type code *pval* and IDATAIN = 0 is to be executed. Attempting to do so results in MaryLie program termination accompanied by appropriate messages at the terminal and file 12.

If desired, the contents of an external file (file 13 for example) can be placed in the initial conditions buffer for use by a *pval* type command by executing a command of the kind shown below:

```
   putinbuf     rt
13 , 14 , -1 , 0 , 0 , 0
```

See section 7.2. This command should precede the command with the type code *pval*.

8.24 Compute Chromatic Expansion

Type Code: cxp

Required Parameters:

1. IOPT (controls meaning of the second parameter, EPSILON)
   = 1 if EPSILON = P_τ.
   = 2 if EPSILON = δ. See section 4.1.2.
2. EPSILON (value of expansion parameter).
3. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP a positive integer in the range 1 through 9) to use the map in storage location NMAP.
   = −NMAP (with NMAP a positive integer in the range 1 through 6) to use the map in buffer number NMAP.

Example:

```
cxppt    cxp
 1  .001  -3
```

This specifies a command with the user given name *cxppt*. When invoked it makes a chromatic expansion of the map in buffer 3 for the option EPSILON = P_τ with EPSILON having the value .001.

Description:

Consider a map $\mathcal{M}$ of the form

$$\mathcal{M} = e^{:f_2:} e^{:f_3:} e^{:f_4:} \tag{8.24.1}$$

where f_2 contains only terms that are quadratic in the "geometric" coordinates or terms that are quadratic in P_τ; f_3 contains only terms that are linear in P_τ (and quadratic in the geometric coordinates) or terms that are cubic in P_τ; f_4 contains only terms that are quadratic in P_τ (and quadratic in the geometric coordinates) or terms that are quartic in P_τ. The maps $\mathcal{B}$ and $\mathcal{A}_b$ produced by *cod* and *tasm* are of this form. See sections 8.1 and 8.2. Such maps, when acting only on transverse variables, are described by a matrix that has a power series expansion in ϵ (where $\epsilon = P_\tau$ or $\epsilon = \delta$). See (8.1.30) and (8.2.26). A command with type code *cxp* produces these expansions. When executed, the following maps are put in the following buffers:

1. Buffer 1 contains the ϵ independent matrix in its matrix part, and zeroes in its polynomial part.

2. Buffer 2 contains the matrix that multiples ϵ in its matrix part, and zeroes in its polynomial part.

3. Buffer 3 contains the matrix that multiples ϵ^2 in its matrix part, and zeroes in its polynomial part.

4. Buffer 4 contains in its matrix part the sum of the terms described above multiplied by their appropriate powers of ϵ and evaluated for the specified value of ϵ. Its polynomial part contains zeroes.

5. Buffer 5 is left unchanged.

6. Buffer 6 is left unchanged.

As described in sections 8.1 and 8.2, such expansions are produced by *cod* and *tasm* for visual inspection, but they are not then available for fitting or optimizing or scanning, etc. Since *cxp* computes and puts these expansions in buffers, they then become so available. The Exhibit below illustrates, for example, how a *tasm* command followed by a *cxp* command produces the desired expansions in buffers.

```
#comment
 Exhibit 8.24.
 This is a test of the type code cxp.  A command with type
 code tasm is applied to the one-turn map for the small static
 storage ring of Exhibit 2.5.1 to compute an eigenvector expansion.
 Next, the chromatic expansion for script A_b, the normalizing map for
 the betatron factor, is found using a cxp command, and the result
 is seen to agree with the eigenvector expansion.  Both options
 of tasm and cxp are illustrated.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 cxppt    cxp
   1.00000000000000      1.000000000000000E-03  -3.00000000000000
 cxpd     cxp
   2.00000000000000     -1.188970448993188E-03  -3.00000000000000
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
```

```
 hfq      quad
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hdq      quad
  0.500000000000000      -1.92000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hcs      sext
  0.500000000000000      0.000000000000000E+00
 vcs      sext
  0.500000000000000      0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 tasmpt   tasm
   1.00000000000000       1.00000000000000       3.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 tasmd    tasm
   2.00000000000000       1.00000000000000       3.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 ptdata   ps1
  1.000000000000000E-02  0.000000000000000E+00  1.000000000000000E-02
  0.000000000000000E+00  0.000000000000000E+00  1.000000000000000E-03
 ddata    ps1
  1.000000000000000E-02  0.000000000000000E+00  1.000000000000000E-02
  0.000000000000000E+00  0.000000000000000E+00 -1.188970448993188E-03
 clear    iden
 zer      zer
  0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
  0.000000000000000E+00
 gbuf1    gbuf
   2.00000000000000       1.00000000000000
 gbuf2    gbuf
   2.00000000000000       2.00000000000000
 gbuf3    gbuf
   2.00000000000000       3.00000000000000
 gbuf4    gbuf
   2.00000000000000       4.00000000000000
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
```

```
     2*half
 pttest
     1*ring        1*ptdata      1*tasmpt      1*cxppt       1*writeout
 dtest
     1*ring        1*ddata       1*tasmd       1*cxpd        1*writeout
 writeout
     1*gbuf1       1*mapout      1*gbuf2       1*mapout      1*gbuf3    &
     1*mapout      1*gbuf4       1*mapout
#lumps
#loops
#labor
    1*fileout
    1*pttest
    1*clear
    1*dtest
    1*fin

twiss analysis of static map
 P sub tau =   1.000000000000000E-003
 delta =  -1.188970448993188E-003


eigenvector expansion for epsilon defined in terms of P sub tau:

on energy matrix of eigenvectors

matrix for map is :

 2.53724E+00  6.93889E-18  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00


epsilon correction

matrix for map is :

 4.73207E-01  4.56623E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 4.33537E-03 -1.37768E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.51450E-01  9.07617E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.45609E-02  6.90817E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


epsilon**2 correction

matrix for map is :

 8.13029E-01  5.87798E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
-1.05289E-01 -1.08092E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -5.00009E-01  9.47381E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  8.21307E-02  2.21126E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


matrix of eigenvectors when epsilon=  0.10000000D-02

matrix for map is :

 2.53772E+00  4.56682E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57067E-01  3.93991E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49603E+00  9.17091E-05  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48904E-01  4.00649E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

matrix for map is :

 2.53724E+00  6.93889E-18  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 4.73207E-01  4.56623E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 4.33537E-03 -1.37768E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.51450E-01  9.07617E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.45609E-02  6.90817E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 8.13029E-01  5.87798E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-1.05289E-01 -1.08092E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -5.00009E-01  9.47381E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  8.21307E-02  2.21126E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :
```

```
matrix for map is :

 2.53772E+00  4.56682E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57067E-01  3.93991E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49603E+00  9.17091E-05  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48904E-01  4.00649E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


twiss analysis of static map
 P sub tau =   1.000000000000000E-003
 delta =  -1.188970448993188E-003


eigenvector expansion for epsilon defined in terms of momentum deviation:

on momentum matrix of eigenvectors

matrix for map is :

 2.53724E+00  6.93889E-18  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00


epsilon correction

matrix for map is :

-3.98066E-01 -3.84116E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.64696E-03  1.15892E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.95644E-01 -7.63497E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.90730E-02 -5.81122E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


epsilon**2 correction

matrix for map is :

 5.17137E-01 -1.45565E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-7.50390E-02 -5.95479E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.10606E-01  6.59239E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  6.23685E-02  1.47981E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00


matrix of eigenvectors when epsilon= -0.11889704D-02

matrix for map is :

 2.53772E+00  4.56682E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57067E-01  3.93991E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49603E+00  9.17094E-05  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48904E-01  4.00649E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

matrix for map is :

 2.53724E+00  6.93889E-18  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.57071E-01  3.94129E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49638E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48938E-01  4.00580E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

-3.98066E-01 -3.84116E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.64696E-03  1.15892E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.95644E-01 -7.63497E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.90730E-02 -5.81122E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 5.17137E-01 -1.45565E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-7.50390E-02 -5.95479E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.10606E-01  6.59239E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  6.23685E-02  1.47981E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :


matrix for map is :

 2.53772E+00  4.56682E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
-3.57067E-01  3.93991E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  2.49603E+00  9.17094E-05  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.48904E-01  4.00649E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


end of MARYLIE run
```

8.25 Transfer Quadratic Moments

Type Code: tqm

Required Parameters:

1. JOB
   = 1 to use array part of map as source of 2nd moments, and to set matrix part accordingly.
   = 2 to use matrix part of map as source of 2nd moments, and to set array part accordingly.
2. IMAP
   = 0 to carry out this "transfer" on the current map.
   = NBUF (an integer from 1 through 6) to carry out this "transfer" on the map in buffer number NBUF.

Example:

```
setmat     tqm
  1         2
```

This specifies a command with the user name *setmat*. When invoked it uses the array part of buffer 2, locations 7 through 27 (see section 14.1), as a source of 2nd moments and sets the matrix part accordingly so that

$$r1(i,j) = \langle z_i z_j \rangle$$

where r1 is the matrix part of buffer 1.

Description:

When beam moments are stored in map format, it is usually the case that the array part of the map contains all moments from zero through fourth order, and the matrix part contains 2nd order moments. However, it may happen that the matrix part contains something else, and it is desired that it contain 2nd moments. This can be achieved by executing a command with type code *tqm* and JOB = 1. See the example above and section 8.37. One could also imagine a circumstance in which 2nd moments are specified in matrix form, and it would be desirable to have them in the array part of a map so they could be acted on by *amap* or analyzed by *moma*. See sections 8.17 and 8.37. This can be achieved by executing a *tqm* command with JOB = 2.

8.26 Select Quantities

Type Code: sq

Required Parameters:

1. INFILE (File number from which selections are to be read).
   = 0 or 5 to read interactively from the terminal.
   = IFILE (any positive integer) to read from that file, after rewinding that file.
   = −IFILE (any negative integer) to read from file IFILE at its current position without rewinding.
2. LOGFILE (File number to which selections are to be written. Ignored unless INFILE = 0 or 5, but still required.)
   = 0 to not write out selections.
   = JFILE (any positive integer) to write selections to that file.
3. ISTORE
   = 1 to put selections in storage associated with inner procedure.
   = 2 to put selections in storage associated with outer procedure.
   = 3 to put selections in auxiliary storage.
4. ISEND (Ignored, but still required, when INFILE = 0 or 5).
   = 0 to write out nothing.
   = 1 to write out selections at the terminal.
   = 2 to write out selections on file 12.
   = 3 to write out selections at the terminal and on file 12.

Example:

```
choices     sq
   25, 0, 3, 1
```

This specifies a command with the user name *choices*. When invoked it reads selections from file 25, puts them in auxiliary storage, and writes them at the terminal.

Description:

This command is used in conjunction with the *wsq* command. See section 8.27. It is used to select various parameters used by MaryLie and various quantities computed by MaryLie . See, for example, section 10.9. If desired, it may be employed interactively by setting INFILE = 5. The results of an interactive use may be written on LOGFILE for subsequent noninteractive use. The Exhibit below shows part of the response to *sq* when invoked interactively and help is requested. The remainder of the response is similar to that of *aim*. See section 9.5.

```
#comment
  Exhibit 8.26.
  This is a fragmentary MaryLie run to illustrate the initial response
  to a command with type code sq when invoked interactively and help
  is requested.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  sq       sq
   0.000000000000000E+00  0.000000000000000E+00   3.00000000000000
    1.00000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*sq
     1*end

  In subroutine sq

  Select quantities by entering their symbols
  (Enter a # sign when finished, or type help to review the defined
symbols)
help

  Symbols are defined for the following categories:
   1. Path length and related quantities, the current map,
      various stored maps, and work buffers in map format.
   2. Standard quantities derived from the current map, including tunes,
      chromaticities, anharmonicities, twiss parameters,
      dispersions and phase slip factors.
   3. Beam parameters.
   4. User calculated quantities and merit functions.
   -----Select help catagory-----
```

8.27 Write Selected Quantities

Type Code: wsq

Required Parameters:

1. JOB
   = 1 to write selected quantities.
   = 2 to write selected variables (parameters).
   = 3 to write selected quantities and variables.
   = 4
2. ISTORE
   = 1 to use selections in storage associated with inner procedure.
   = 2 to use selections in storage associated with outer procedure.
   = 3 to use selections in auxiliary storage.
3. IFILE (File number into which selected quantities are to be written).
   = 0 to not write selected quantities on a file.
   = NFILE to write selected quantities on file NFILE.
   = −K to write selected quantities into the array *ucalc*. See note below.
4. LFORM (Disk file format selection).
   = 0 to write a line of names only for use as a column headers in table format (up to 10 columns).
   = 1 to write the values in 6 digit table format (up to 10 items/line).
   = 2 to write 3 items/line in the form 'name=value' to 8 digits.
   = 3 to write 1 item/line in the form 'name=value' to 16 digits.
   = 4 for 1-D scan table format: I, X(I) = the first loop variable, followed by loop aims, including any selected RMS errors.
   = 5 for 2-D scan table format: I, J, X(I), Y(J) = the first two loop variables, which would be scanned if a 2D scan were defined, followed by loop aims, including any selected RMS errors.
5. JFORM
   Format flag for terminal output and/or file 12 output as specified by ISEND. Parameter values have the same meaning as for LFORM above.
6. ISEND
   = 0 to do nothing.
   = 1 to write at the terminal.
   = 2 to write on file 12.
   = 3 to write at the terminal and on file 12.

NOTE: When JOB=1 and IFILE=−K, *wsq* writes the values of the selected quantities into successive locations of the array *ucalc* starting at location K. See, for example, section 10.5. When JOB=2 and IFILE=−K, *wsq* writes the values of the selected variables into successive locations of the array *ucalc* starting at location K. See, for example, section 10.3.2. To view the contents of *ucalc* at any time, say to check that it indeed contains what the user has in mind, the command *wuca* may be used. See section 7.41.

Example:

```
results     wsq
 1 , 1 , 30 , 1 , 2 , 0
```

This specifies a command with the user name *results*. When invoked it writes the selected quantities (from the selection list in storage associated with an inner procedure) to file 30 using the format specified by LFORM = 1. If it were to write at the terminal and/or file file 12 (it doesn't because ISEND = 0), it would use the format specified by JFORM = 2.

Description:

Commands with type code *wsq* are used in conjunction with *sq* commands (see section 8.26) or *aim* and *vary* commands (see sections 9.5 and 9.6) to write various output files that can be used for subsequent plotting and/or other purposes. See, for example, section 10.9. They may also be used to put various quantities in the *ucalc* array for further use. See note above and sections 10.3.2 and 10.5.

8.28 Change Tune Range

Type Code: ctr

Required Parameters:

1. JOB
   = 0 to change tune ranges.
   = 1 to write tune ranges at the terminal.
   = 2 to write tune ranges on file 12.
   = 3 to write tune ranges at the terminal and on file 12.
2. ITX
   = 0 to place horizontal tune jump at 0.
   = 1 to place horizontal tune jump at 1/2.
3. ITY
   = 0 to place vertical tune jump at 0.
   = 1 to place vertical tune jump at 1/2.
4. ITT
   = 0 to place temporal tune jump at 0.
   = 1 to place temporal tune jump at 1/2.

Example:

```
mytr     ctr
 0, 0, 0, 0
```

This specifies a command with the user given name *mytr*. When invoked it places all tune jumps at zero.

Description:

A command with type code *tasm* usually reports tunes in the range (0,1), and a command with type code *tadm* reports transverse tunes in the range (0,1) and the temporal tune in the range (-1/2,1/2). See sections 8.2 and 8.3. Thus, transverse tunes usually (have a discontinuity) jump at 0, and the temporal tune jumps at 1/2. The locations of these jumps can be changed using a command with type code *ctr*.

8.29 Apply Script N Inverse

Type Code: asni

Required Parameters:

1. IOPT
   = 1 for a static script N.
   = 2 for a dynamic script N.
2. IMAP
   = J (with J an integer from 1 to 9) to use map in storage location J.
3. NFCFILE
4. ISTART
5. IGROUP
6. NWRITE

Example:

```
filament     asni
 1, 1, 16, 1, 10, 2
```

This specifies a command with user given name *filament*. When evoked it applies the inverse of the map (assumed to be in static normal form) stored in map location 1. It acts on the phase-space data in the initial conditions buffer, and writes results to file 16. It presumes that the initial conditions buffer contains groups of data as described below.

Description:

Commands with type code *asni* are primarily meant to act on files produced by a *rt* (ray trace, see section 7.2) or a *circ* (see section 7.3) command. (However there is sufficient flexibility to handle the general case.)

Let $\mathcal{M}$ be the current transfer map. Suppose a *rt* or *circ* command is applied to a set of initial conditions and the results are written on some external final conditions file for some value of NTRACE (or NTIMES) and some value of NWRITE. At the end of this operation the external final conditions file will contain groups of rays and (assuming no rays are lost) the number of rays in each group will be the same, namely the number of initial conditions. The first group of rays will be the result of $\mathcal{M}$ acting on the initial conditions 1*NWRITE times, the second group will be the result of $\mathcal{M}$ acting on the initial conditions 2*NWRITE times, the third group will be the result of $\mathcal{M}$ acting on the initial conditions 3*NWRITE times, etc. Alternatively, if the initial conditions are first copied onto the final conditions file before starting the operation, the first group of rays will be the result of $\mathcal{M}$ acting on the initial conditions 0*NWRITE times, the second group will be the result of $\mathcal{M}$ acting on

the initial conditions 1*NWRITE times, the third group will be the result of $\mathcal{M}$ acting on the initial conditions 2*NWRITE times, etc.

Next suppose a file having this structure is read into the initial conditions buffer. Set IGROUP to equal the number of initial conditions and let $\mathcal{N}$ be the map stored in storage location IMAP. Then, after a command with type code *asni* is executed, the file NFCFILE will contain as many groups of rays as were in the initial conditions buffer. The first group will be the result of applying the map $\mathcal{N}^{-1}$ ISTART times to the first group in the initial conditions buffer, the second group will be the result of applying the map $\mathcal{N}^{-1}$ (ISTART + 1*NWRITE) times to the second group in the initial conditions buffer, the third group will be the result of applying the map $\mathcal{N}^{-1}$ (ISTART + 2*NWRITE) times to the third group in the initial conditions buffer, etc. It should be noted that the application of $\mathcal{N}^{-1}$ is done analytically. Thus, unlike the case of an ordinary map application routine, no truncations or approximations occur. This is possible because of the simple structure of a map in normal form.

8.30 Compute Power of Nonlinear Part

Type Code: pnlp

Required Parameters:

1. IOPT
   = 0 to replace matrix portion of the map by the identity matrix.
   = 1 to leave the matrix portion unaffected.
2. POWER
3. MAPIN
   = 0 to use current transfer map.
   = NMAP (with NMAP an integer from 1 to 9) to use map in storage location NMAP.
4. MAPOUT
   = KMAP (with KMAP an integer from 1 to 9) to place resulting map in storage location KMAP.
   = 0 to make the resulting map the current transfer map.
   = −KBUF (with KBUF an integer from 1 to 6) to place resulting map in buffer number KBUF.

Example:

```
sqroot     pnlp
0 , .5 , 0 , 0
```

This specifies a command with the user given name *sqroot*. When invoked, it sets the matrix portion of the map to the identity matrix, and computes the square root of the remainder of the map.

Description:

Given a map in the form $\mathcal{M} = \mathcal{R}\mathcal{N}$ where in this context $\mathcal{N}$ is the *nonlinear* part of the map,

$$\mathcal{N} = \exp : f_3 : \exp : f_4 :,$$

a command with the type code *pnlp* computes the map $\mathcal{N}^s$ where s is any real number. Such maps are sometimes of interest for ray tracing. For example, symplectic tracking $\mathcal{R}\mathcal{N}^{1/2}\mathcal{N}^{1/2}$ through a loop using *circ* may have better Newton convergence properties than simple symplectic tracking using $\mathcal{M}$.

8.31 Check Symplectic Condition

Type Code: csym

Required Parameters:

1. ISEND
   =0 to do nothing.
   =1 to write results at the terminal.
   =2 to write results on file 12.
   =3 to write results at the terminal and on file 12.

Example:

```
check     csym
  1
```

This specifies a command with the user given name *check*. When invoked it computes the "symplectic violation" of the matrix part of the current transfer map and writes the result at the terminal. This result is also placed in location 4 of the array EX where it is available for fitting or optimization, etc. See section 7.44.

Description:

It is sometimes useful to ascertain how nearly symplectic a matrix is. Let R be the matrix under scrutiny. A command with the type code *csym* computes the quantity

$$\| RJR^T J^T - I \| .$$

This quantity vanishes for a symplectic matrix. Here the matrix norm employed is the maximum column sum norm. See section 8.33.

8.32 Polynomial Scalar Product

Type Code: psp

Required Parameters:

1. JOB
   =1 to use $USp(6)$ invariant scalar product.
   =2 to use integration over S^5 invariant scalar product.
2. MAP1IN
   =0 to use current transfer map.
   =NMAP1 (with NMAP1 an integer from 1 to 9) to use polynomial in storage location NMAP1.
3. MAP2IN
   =0 to use current transfer map.
   =NMAP2 (with NMAP2 an integer from 1 to 9) to use polynomial in storage location NMAP2.
4. ISEND
   =0 to do nothing.
   =1 to write results at the terminal.
   =2 to write results on file 12.
   =3 to write results at the terminal and on file 12.

Example:

```
fdotg     psp
 1 , 1 , 2 , 1
```

This specifies a command with the user given name *fdotg*. When invoked it takes the polynomial f from storage location 1 and the polynomial g from storage location 2. It computes the scalar product (f, g) using the $USp(6)$ invariant scalar product and writes the result at the terminal. Its value is also placed in location 3 of the array EX where it is available for fitting or optimization, etc. See section 7.44.

Description:

For some purposes it is useful to introduce a scalar product in the vector space of polynomials. This can be done in several ways. Two are available in MaryLie .

The first scalar product is invariant under the group $USp(6)$. For monomials introduce the notation

$$z^j = z_1^{j_1} z_2^{j_2} \cdots z_{2n}^{j_{2n}},$$

$$j! = j_1! j_2! \cdots j_{2n}!,$$

$$|j| = j_1 + j_2 \cdots + j_{2n}.$$

The $USp(6)$ invariant scalar product is defined by the rule

$$(z^j, z^{j'}) = \delta_{jj'} j!.$$

Note that for this definition different monomials are orthogonal.

The second scalar product, probably less natural, is invariant under the group $U(3)$. It is defined by the rule

$$(z^j, z^{j'}) = \delta_{|j||j'|} \int_{S^5} d\Omega z^j z^{j'}.$$

8.33 Compute Matrix Norm

Type Code: mn

Required Parameters:

1. IOPT
   = 0 to work directly with matrix.
   = 1 to first subtract the identity from the matrix.
   = 2 to first subtract J from the matrix.
2. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.

Example:

```
norm     mn
   0 , 1
```

This specifies a command with the user given name *norm*. When invoked, it computes the norm of the matrix part of the current transfer map and writes the result at the terminal. The result is also placed in location 5 of the array EX where it is available for fitting or optmization. See section 7.44

Description:

Let R be a matrix. Its norm $\| R \|$ is defined to be the maximum column sum norm:

$$\| R \| = \max_k (\sum_j |R_{jk}|).$$

If desired, the identity matrix or the J matrix can be subtracted off before the norm is computed. That is, the quantities $\| R - I \|$ and $\| R - J \|$ can also be computed.

8.34 Generate Beam

Type Code: bgen

Required Parameters:

1. JOB
   = 0 to compute moments of particle distribution stored in the initial conditions buffer.
   = 1 to generate a random 2-variable distribution corresponding to a uniformly filled 2D ellipse.
   = 2 to generate a random 4-variable distribution corresponding to a uniformly filled 4D ellipsoid.
   = 3 to generate a random 6-variable distribution corresponding to a uniformly filled 6D ellipsoid.
   = 4 to generate a random 2-variable gaussian distribution.
   = 5 to generate a random 4-variable gaussian distribution.
   = 6 to generate a random 6-variable gaussian distribution.
   = 7 to generate a 2-variable systematic uniform distribution on a 1-torus in 2D.
   = 8 to generate a 4-variable systematic uniform distribution on a 2-torus in 4D.
   = 9 to generate a 6-variable systematic uniform distribution on a 3-torus in 6D.
   = 10 to generate a 2-variable random uniform distribution on a 1-torus in 2D.
   = 11 to generate a 4-variable random uniform distribution on a 2-torus in 4D.
   = 12 to generate a 6-variable random uniform distribution on a 3-torus in 6D.
   = 13 to generate a 4-variable random uniform distribution on the 3D surface of an ellipsoid in 4D (a KV distribution).
2. IOPT (Used when JOB $\neq$ 0.)
   = 1 to generate particle distribution only.
   = 2 to compute analytic moments only.
   = 12 to generate particle distribution and compute analytic moments.
   = 13 to generate particle distribution and compute corresponding moments numerically.
   = 123 to generate particle distribution and compute corresponding moments both analytically and numerically.
3. NRAY (Number of rays to be generated.)

4. ISEED (Initial seed for random number generator.)
   = any *nonzero* integer. See description below.
5. ISEND
   = 0 to do nothing.
   = 1 to write at the terminal numerical and analytic values (if computed) of principal second-order moments.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
6. IPSET (Zero or an integer from 1 to 9.)
   This command also requires additional parameters whose values are specified by the parameter set IPSET. (Setting IPSET = 0 is equivalent to using a parameter set all of whose parameters are zero.) The contents of parameter set IPSET are used as follows:
   1. P1 = $\langle X^2 \rangle$ eigenmoment.
   2. P2 = $\langle Y^2 \rangle$ eigenmoment.
   3. P3 = $\langle \tau^2 \rangle$ eigenmoment.
   4. P4 = SIGMAX (Used for gaussian distribution.)
   5. P5 = NOUTFILE1 (File for numerical moments.)
      = 0 to not write out numerical moments.
      = NOUTFILE1 to write out numerically computed moments on the file NOUTFILE1.
   6. P6 = NOUTFILE2 (File for analytic moments.)
      = 0 to not write out analytic moments.
      = NOUTFILE2 to write out analytically computed moments on the file NOUTFILE2.

   Note: When P1 < 0, the values of P2 and P3 are ignored and the eigenmoments are taken from the "f" array of the "current" map according to the prescription

$$\langle X^2 \rangle \text{ eigenmoment } = f(7),$$

$$\langle Y^2 \rangle \text{ eigenmoment } = f(18),$$

$$\langle \tau^2 \rangle \text{ eigenmoment } = f(25).$$

   This feature is convenient when using *bgen* in tandem with *moma*. See sections 8.37, 10.7, and 14.1.

Example:

```
makeic     bgen
   8 , 123 , 2304 , 123 , 1 , 1
```

This specifies a command with the user given name *makeic*. When invoked it generates a 4-variable systematic uniform distribution on a 2-torus in 4-dimensional transverse phase space. The moments of this distribution are also calculated both numerically and analytically. The distribution has 2304 particles. The principal second-order moments (both numerical and analytic) are written at the terminal. Further instructions are taken from parameter set 1. See, for examples of *bgen* use, sections 8.37, 10.7.1, 10.8.1, and 10.12.

Description:

When requested, the type code *bgen* generates a particle distribution and places it in the initial conditions buffer. Some distributions require the use of a random number generator. Two different generators are available: ran1 and ran2. They are described in the *Numerical Recipes* book by Press et al. See section 11.12. When ISEED $\geq +1$, ran1 is used. When ISEED ≤ -1, ran2 is used. (Setting ISEED $= 0$ is not allowed, and produces an error message.) Both are better than most computer system-supplied random number generators. The generator ran1 is believed to be satisfactory for the production of up to 10^8 pseudo random numbers. The generator ran2 is believed to be among the best available for the production of as many as 10^{16} pseudo random numbers. However it is somewhat slower than ran1. If the user is concerned that his/her calculation may be sensitive to the choice of random number generator, the results of both should be compared for several different seeds.

Contents of buffers:

Buffer 1 contains numerically computed second moments in its matrix part, and all numerically computed moments in its array part. Buffer 2 contains analytically computed second moments in its matrix part, and all analytically computed moments in its array part. Buffer 3 contains in its matrix part the zero matrix and in its array part the (second order) eigen moments used by *bgen*. See section 8.17 for the definition of moments. Buffers 4 through 6 are empty.

8.35 Translate Initial Conditions

Type Code: tic

Required Parameters:

1. DX
2. DPx
3. DY
4. DPy
5. Dτ
6. DPτ

Example:

```
shiftx     tic
.001 , 0 , 0 , 0 , 0 , 0
```

This specifies a command with the user given name *shiftx*. When invoked, it augments the x coordinates of particles in the initial conditions buffer by .001, and leaves all other coordinates unchanged.

Description:

This routine translates all the "initial conditions" in the particle phase-space array *zblock* (that have not been marked as 'lost') by the parameter values listed above. That is, the X coordinate for each (nonlost) particle is replaced by X+DX, etc. Commands of this type are useful for simulating element misplacements and the action of kickers. They are also useful for generating beams. See section 10.7.

8.36 Principal Planes Analysis

Type Code: ppa

Required Parameters:

Example:

Description:

Available but not yet documented.

8.37 Moment Analysis

Type Code: moma

Required Parameters:

1. JOB
   = 0 to simply compute moments about the centroid.
   = 11 to analyze moments of a 2-dimensional phase-space distribution given through order 4. Analysis is made with respect to the origin.
   = 12 to analyze moments of a 2-dimensional phase-space distribution given through order 4. Analysis is made with respect to the distribution centroid.
   = 21 to analyze moments of a 4-dimensional phase-space distribution given through order 4. Analysis is made with respect to the origin.
   = 22 to analyze moments of a 4-dimensional phase-space distribution given through order 4. Analysis is made with respect to the distribution centroid.
   = 31 to analyze moments of a 6-dimensional phase-space distribution given through order 4. Analysis is made with respect to the origin.
   = 32 to analyze moments of a 6-dimensional phase-space distribution given through order 4. Analysis is made with respect to the distribution centroid.
2. ISEND
   = 0 to do nothing.
   = 1 to write results at the terminal.
   = 2 to write results on file 12.
   = 3 to write results at the terminal and on file 12.
3. NINFILE
   = 0 to use current transfer map as moment source.
   = −NMAP to get moments from storage location NMAP (see section 7.17).
   = INFILE (file number from which moments are to be read).
4. NREWIND
   = 0 to read (from file INFILE) in its current position.
   = 1 to first rewind file unit INFILE, then read.
5. NSKIP
   Number of data sets (moment sets) to be skipped in file INFILE (starting from the current position) before a read actually occurs.
6. NOUTFILE
   = 0 to not write out transformed moments.
   = NOUTFILE to write transformed moment set on file NOUTFILE.

Example:

```
analmom     moma
31 , 1 , -2 , 0 , 0 , 20
```

This specifies a command with the user given name *analmom*. When invoked it analyzes the moments in storage location 2, with respect to the origin, on the assumption that the moments are those for a 6-dimensional phase-space distribution. Transformed moments are written on file 20.

Note: When *moma* takes in moments from a "map", it uses only the array part of the map and ignores its matrix part. Note that the zeroth entry $P(0)$ must have the value 1. See section 4.4.3. It should also be observed that after execution of a command with type code *moma* buffers 2 and 3 contain moments in their array parts, but not in their matrix parts. (See the description below.) However, if desired, the corresponding quadratic moments can be placed in the matrix parts by subsequent invocation of a command with type code *tqm*. See section 8.25.

Description:

Consider first the case of two-dimensional phase space, which we take to have coordinates X, P_x. In this case the emittance ϵ_x is defined by the relations

$$\epsilon_x^2 = \langle X^2 \rangle \langle P_x^2 \rangle - \langle X P_x \rangle^2, \tag{8.37.1}$$

$$\epsilon_x = +\sqrt{\epsilon_x^2}. \tag{8.37.2}$$

It can be shown from the Schwarz inequality and the positivity of the phase-space distribution function that the right side of (37.1) is positive, and therefore (37.2) makes sense. Next define "beam" betatron functions α, β, γ by the rules

$$\alpha = -\langle X P_x \rangle / \epsilon_x, \tag{8.37.3}$$

$$\beta = \langle X^2 \rangle / \epsilon_x, \tag{8.37.4}$$

$$\gamma = \langle P_x^2 \rangle / \epsilon_x. \tag{8.37.5}$$

Then, from (37.4) and (37.5), there are the inequalities $\beta \geq 0$ and $\gamma \geq 0$. And from (37.1) through (37.5) there is the relation

$$1 = \beta\gamma - \alpha^2. \tag{8.37.6}$$

Next, define a matrix A by the rule

$$A = \begin{pmatrix} 1/\sqrt{\beta} & 0 \\ \alpha/\sqrt{\beta} & \sqrt{\beta} \end{pmatrix}. \tag{8.37.7}$$

Since A is 2×2 and has unit determinant, it is symplectic. Finally, define a moment matrix Z_{ab} by the rule

$$Z_{ab} = \langle z_a z_b \rangle. \tag{8.37.8}$$

In the two-dimensional case this matrix has the entries

$$Z = \begin{pmatrix} \langle X^2 \rangle & \langle XP_x \rangle \\ \langle XP_x \rangle & \langle P_x^2 \rangle \end{pmatrix}. \tag{8.37.9}$$

From these various definitions it is easily verified that there is the matrix relation

$$AZA^T = \epsilon_x I_x, \tag{8.37.10}$$

where I_x is the matrix

$$I_x = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{8.37.11}$$

The relation (37.10) can also be written in the form

$$Z = (A)^{-1} I_x (A^{-1})^T \epsilon_x = E_x \epsilon_x \tag{8.37.12}$$

where E_x is the "X" *envelope matrix*

$$E_x = (A)^{-1} I_x (A^{-1})^T. \tag{8.37.13}$$

With the aid of the symplectic condition (3.5.3) the "X" envelope matrix can also be written in the form

$$E_x = J^T A^T J I_x J^T A J = J^T A^T I_x A J. \tag{8.37.14}$$

Evaluation of (37.14) gives the result

$$E_x = \begin{pmatrix} \beta & -\alpha \\ -\alpha & \gamma \end{pmatrix}, \tag{8.37.15}$$

which is consistent with (37.3) through (37.5) and (37.9).

Consider next four-dimensional transverse phase space, which we take to have coordinates X, P_x, Y, P_y. In this case we again define a second-order moment matrix Z using (37.8) with a, b ranging from 1 to 4. See section 14.1. Evidently Z is symmetric. It can be shown from the positivity of the phase-space distribution function that Z is positive definite. It follows that there is a symplectic matrix A with the property

$$AZA^T = \epsilon_x I_x + \epsilon_y I_y \tag{8.37.16}$$

where I_x and I_y are the matrices

$$I_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{8.37.17}$$

$$I_y = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{8.37.18}$$

Evidently the right side of (37.16) is a diagonal matrix with eigenvalues ϵ_x,ϵ_y. These eigenvalues will be positive because Z is positive definite. We will refer to the quantities ϵ_x,ϵ_y as *eigen emittances*. The operation on the left side of (37.16) is called a *congruency* transformation, and we say that Z is *symplectically congruent* (congruent under a symplectic matrix) to a diagonal matrix. For a proof of these results, see the appropriate reference in section 11.5.

Suppose the particles in a phase-space distribution are acted upon by a linear symplectic map $\mathcal{M}_2$. Then, employing a notation similar to (17.5), we will have a result of the form

$$Z_{ab}^{\rm tr} = \int d^4z(\mathcal{M}_2 z_a)(\mathcal{M}_2 z_b)f(z). \tag{8.37.19}$$

But since $\mathcal{M}_2$ is linear, there are the relations

$$\mathcal{M}_2 z_a = \sum_c R_{ac} z_c, \tag{8.37.20}$$

$$\mathcal{M}_2 z_b = \sum_d R_{bd} z_d, \tag{8.37.21}$$

where R is a symplectic matrix. It follows that (37.19) can be rewritten in the form

$$Z_{ab}^{\rm tr} = \sum_{cd} R_{ac} R_{bd} Z_{cd}, \tag{8.37.22}$$

or, in matrix notation,

$$Z^{\rm tr} = RZR^T. \tag{8.37.23}$$

This result is the second-moment version of (17.12).

Next, observe that the relations (37.16) and (37.23) are of the same general form. Thus, if we view the right side of (37.16) as a set of second-order *eigen moments*, we may say that any given set of second-order moments Z is transformed to the corresponding set of eigen moments (whose off-diagonal entries are all zero and diagonal entries are ϵ_x, ϵ_y) by the action of a linear symplectic map $\mathcal{M}_2$ with $R = A$.

Now combine (37.16) and (37.23) to find the relation

$$AR^{-1}Z^{\rm tr}(R^T)^{-1}A^T = AZA^T = \epsilon_x I_x + \epsilon_y I_y. \tag{8.37.24}$$

Evidently this relation can be rewritten in the form

$$AR^{-1}Z^{\rm tr}(AR^{-1})^T = \epsilon_x I_x + \epsilon_y I_y. \tag{8.37.25}$$

It is easily verified that the combination (AR^{-1}) is a symplectic matrix if R and A are symplectic. It follows that both $Z^{\rm tr}$ and Z are symplectically congruent to the same diagonal form, and therefore the eigen emittances are *invariant* under linear transport. Put another way, two sets of quadratic moments are related by linear (and symplectic) transport if and only if they have the same eigen emittances.

Finally, observe that (37.16) can be rewritten in the form

$$Z = A^{-1}I_x(A^T)^{-1}\epsilon_x + A^{-1}I_y(A^T)^{-1}\epsilon_y = E_x\epsilon_x + E_y\epsilon_y. \tag{8.37.26}$$

Here E_x and E_y are the "X" and "Y" envelope matrices given by the relations

$$E_x = A^{-1} I_x (A^{-1})^T = J^T A^T I_x A J, \tag{8.37.27}$$

$$E_y = A^{-1} I_y (A^{-1})^T = J^T A^T I_y A J. \tag{8.37.28}$$

Also, we will refer to the matrix $E_x \epsilon_x$ as the second-order moments *excited* by the "X" eigen emittance ϵ_x, etc.

At this point it should be obvious that the full six-dimensional phase-space case is completely analogous. In this case we define a second-order moment matrix Z using (37.9) with a, b ranging from 1 to 6. Now there is a symplectic martrix A with the congruency property

$$AZA^T = \epsilon_x I_x + \epsilon_y I_y + \epsilon_\tau I_\tau \tag{8.37.29}$$

where I_x, I_y, and I_τ are the matrices

$$I_x = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{8.37.30}$$

$$I_y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{8.37.31}$$

$$I_\tau = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \tag{8.37.32}$$

and there are three eigen emittances, namely ϵ_x, ϵ_y, and ϵ_τ.

Also, (37.29) can be rewritten in the form

$$Z = A^{-1} I_x (A^T)^{-1} \epsilon_x + A^{-1} I_y (A^T)^{-1} \epsilon_y + A^{-1} I_\tau (A^T)^{-1} \epsilon_\tau = E_x \epsilon_x + E_y \epsilon_y + E_\tau \epsilon_\tau. \tag{8.37.33}$$

Here E_x, E_y, and E_τ are the "X", "Y", and "τ" envelope matrices given by the relations

$$E_x = A^{-1} I_x (A^{-1})^T = J^T A^T I_x A J, \tag{8.37.34}$$

$$E_y = A^{-1} I_y (A^{-1})^T = J^T A^T I_y A J, \tag{8.37.35}$$

$$E_\tau = A^{-1} I_\tau (A^{-1})^T = J^T A^T I_\tau A J. \tag{8.37.36}$$

Contents of buffers:

1. The matrix part of buffer 1 contains second-order moments about the centroid. The array part contains zeroth through fourth-order moments about the centroid.

2. Buffer 2 contains in its array part the "diagonalized" (transformed) moments through fourth order. In particular, the quadratic moments are zero except for the "eigenmoments"
$$\langle X^2 \rangle = \langle P_x^2 \rangle = \epsilon_x, \tag{8.37.37}$$
$$\langle Y^2 \rangle = \langle P_y^2 \rangle = \epsilon_y, \tag{8.37.38}$$
$$\langle \tau^2 \rangle = \langle P_\tau^2 \rangle = \epsilon_\tau. \tag{8.37.39}$$
In its matrix part buffer 2 contains a map (matrix) that sends the diagonalized moments back to the original moments. In the notation just employed, this is the matrix A^{-1}. Thus, buffer 2 contains the information needed to recreate the original moments.

3. Buffer 3 contains in its matrix part the matrix A. It specifies the "diagonalizing" map $\mathcal{A}$. This map sends the "original" moments to the diagonalized moments. Buffer 3 contains in its array part the original moments through fourth order with respect to the origin or (if JOB = 12, 22, or 32) the original moments through fourth order with respect to the distribution centroid.

4. Buffer 4 contains the "X" envelope matrix in its matrix part, and the second order moments "excited" by the "X" eigen-emittance in its array part.

5. Buffer 5 contains the "Y" envelope matrix in its matrix part, and the second order moments "excited" by the "Y" eigen-emittance in its array part.

6. Buffer 6 contains the "τ" envelope matrix in its matrix part, and the second order moments "excited" by the "τ" eigen-emittance in its array part.

The Exhibit below illustrates the use of *moma* to find eigen emittances.

```
#comment
Exhibit 8.37.
This MaryLie run illustrates the use of the type code
moma.  It does the folowing:
1. A set of initial conditions is read in.  For the sake of whimsy
    they are the initial conditions for the word MARYLIE.  See section
    2.3.
2. To mix them up, these initial conditions are acted upon (transformed)
    by two twsm maps.
3. A bgen command is used to find the moments of this transformed
    particle distribution.
4. These moments are analyzed by moma.
5. To add more whimsy, the linear part of the one-turn map for the PSR
    (see section 2.5) supplemented by an arot element to couple the
    transverse planes is made to act on these moments using an amap

    command.
```

```
6. The resulting moments are again analyzed by moma.  Note that the
    eigen moments are unchanged, as expected.
7. The same linear part of the PSR one-turn map is made to act on the
    transformed particle distribution.
8. The bgen command is used to find the moments of the resulting particle
    distribution.  Note that these moments agree with those found using

    amap in step 5 above, as expected.
#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hdq      quad
   0.500000000000000      -1.92000000000000      0.000000000000000E+00
   0.000000000000000E+00
  hcs      sext
   0.500000000000000      0.000000000000000E+00
  vcs      sext
   0.500000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  matout   ptm
    1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  iden     iden
  twsx     twsm
    1.00000000000000       30.0000000000000       1.00000000000000
    2.00000000000000
  twsy     twsm
    2.00000000000000       40.0000000000000       3.00000000000000
    4.00000000000000
  arot     arot
    45.0000000000000
  raysin   rt
```

```
    13.0000000000000       14.0000000000000      -1.00000000000000
    1.00000000000000       1.00000000000000      0.000000000000000E+00
  xform    rt
   0.000000000000000E+00   14.0000000000000       4.00000000000000
    1.00000000000000       10.0000000000000      0.000000000000000E+00
  bgen     bgen
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000       1.00000000000000
  gbuf1    gbuf
    2.00000000000000       1.00000000000000
  stm1     stm
    1.00000000000000
  amap     amap
    2.00000000000000       1.00000000000000      -1.00000000000000
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  moma     moma
    31.0000000000000       1.00000000000000      0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  mask     mask
    1.00000000000000       1.00000000000000       1.00000000000000
    1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  fin      end
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
  makeic
      1*raysin      1*iden        1*twsx        1*twsy        1*xform    &
      1*iden
  funnymap
      1*iden        1*ring        1*arot        1*mask
#lumps
#loops
#labor
     1*fileout
     1*makeic
     1*bgen
     1*gbuf1
     1*stm1
     1*moma
     1*funnymap
     1*amap
     1*gbuf1
```

```
     1*moma
     1*funnymap
     1*xform
     1*bgen
     1*gbuf1
     1*mapout
     1*fin

  2754 ray(s) read in from file  13

************************************************
response from bgen for initial distribution:   *
************************************************

  numerically computed values of selected moments
  values of <x*x>, <x*px>, <px*px>:
  0.911426796223987      -0.333603408972363       0.122108037004153
  values of <y*y>, <y*py>, <py*py>:
   2.22066693394091       -1.32442342937190       0.789897039102947
  values of <t*t>, <t*pt>, <pt*pt>:
  0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

map stored in location   1

************************************************
response from moma for initial distribution   *
************************************************
moment analysis

eigen emittances
ex=  1.150208302759803E-003
ey=  8.917327300879897E-004
et=  0.000000000000000E+000

map gotten from location   1

*************************
response from amap      *
*************************
  transformed moments

matrix for map is :

  2.56442E+01  4.02828E+00 -2.27002E+01 -4.14438E+00  8.95406E-01  0.00000E+00
  4.02828E+00  6.72238E-01 -3.94437E+00 -6.75547E-01  2.55200E-02  0.00000E+00
 -2.27002E+01 -3.94437E+00  2.37255E+01  3.90393E+00  3.11838E-01  0.00000E+00
 -4.14438E+00 -6.75547E-01  3.90393E+00  6.85030E-01 -7.31326E-02  0.00000E+00
  8.95406E-01  2.55200E-02  3.11838E-01 -7.31326E-02  3.67180E-01  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

  f(  1)=f( 10 00 00 )=  1.6427969606320
```

```
f(  2)=f( 01 00 00 )= 0.25337979304929
f(  3)=f( 00 10 00 )= -1.4093360626378
f(  4)=f( 00 01 00 )=-0.26258622059057
f(  5)=f( 00 00 10 )= 7.10042331749078E-02
f(  7)=f( 20 00 00 )=  25.644222528780
f(  8)=f( 11 00 00 )=  4.0282833804564
f(  9)=f( 10 10 00 )= -22.700172399453
f( 10)=f( 10 01 00 )= -4.1443849789773
f( 11)=f( 10 00 10 )= 0.89540566852674
f( 13)=f( 02 00 00 )= 0.67223818330547
f( 14)=f( 01 10 00 )= -3.9443679266310
f( 15)=f( 01 01 00 )=-0.67554652714702
f( 16)=f( 01 00 10 )= 2.55200353794006E-02
f( 18)=f( 00 20 00 )=  23.725460452435
f( 19)=f( 00 11 00 )=  3.9039318642944
f( 20)=f( 00 10 10 )= 0.31183800881827
f( 22)=f( 00 02 00 )= 0.68502951473739
f( 23)=f( 00 01 10 )=-7.31326369560792E-02
f( 25)=f( 00 00 20 )= 0.36718018081847
f( 28)=f( 30 00 00 )=  57.763775979984
f( 29)=f( 21 00 00 )=  7.8672760816525
f( 30)=f( 20 10 00 )= -39.558751646963
f( 31)=f( 20 01 00 )= -8.5851972495982
f( 32)=f( 20 00 10 )=  5.5368629088634
f( 34)=f( 12 00 00 )=  1.1619663150996
f( 35)=f( 11 10 00 )= -6.2556059328187
f( 36)=f( 11 01 00 )= -1.2255215456628
f( 37)=f( 11 00 10 )= 0.49016886958874
f( 39)=f( 10 20 00 )=  35.416088982804
f( 40)=f( 10 11 00 )=  6.4189690085460
f( 41)=f( 10 10 10 )= -1.2599372191073
f( 43)=f( 10 02 00 )=  1.3109482802407
f( 44)=f( 10 01 10 )=-0.65883685360157
f( 46)=f( 10 00 20 )=  1.3007942318064
f( 49)=f( 03 00 00 )= 0.18974694248181
f( 50)=f( 02 10 00 )= -1.0978383283479
f( 51)=f( 02 01 00 )=-0.19227538324372
f( 52)=f( 02 00 10 )= 1.95028147046973E-02
f( 54)=f( 01 20 00 )=  6.6424002684596
f( 55)=f( 01 11 00 )=  1.0825830170125
f( 56)=f( 01 10 10 )= 0.11764361858597
f( 58)=f( 01 02 00 )= 0.19791209348012
f( 59)=f( 01 01 10 )=-4.34729619365373E-02
f( 61)=f( 01 00 20 )= 0.18486153487156
f( 64)=f( 00 30 00 )= -41.702065675090
f( 65)=f( 00 21 00 )= -6.3945034856785
f( 66)=f( 00 20 10 )= -1.9118281919574
f( 68)=f( 00 12 00 )= -1.0835478429965
f( 69)=f( 00 11 10 )= 7.44097018050738E-03
f( 71)=f( 00 10 20 )=-0.96466932108752
f( 74)=f( 00 03 00 )=-0.20687996272775
f( 75)=f( 00 02 10 )= 6.91619687513366E-02
f( 77)=f( 00 01 20 )=-0.19811725062738
```

```
f( 80)=f( 00 00 30 )= 0.10222992293838
f( 84)=f( 40 00 00 )=  987.42305940068
f( 85)=f( 31 00 00 )=  149.17441518208
f( 86)=f( 30 10 00 )= -817.14443250242
f( 87)=f( 30 01 00 )= -155.88960701602
f( 88)=f( 30 00 10 )=  51.788851508784
f( 90)=f( 22 00 00 )=  23.446959955017
f( 91)=f( 21 10 00 )= -132.18403407589
f( 92)=f( 21 01 00 )= -24.116981465482
f( 93)=f( 21 00 10 )=  5.1674426378288
f( 95)=f( 20 20 00 )=  760.01771966059
f( 96)=f( 20 11 00 )=  134.43688536022
f( 97)=f( 20 10 10 )= -17.374565641116
f( 99)=f( 20 02 00 )=  24.963016717637
f(100)=f( 20 01 10 )= -6.5247030079177
f(102)=f( 20 00 20 )=  10.467002421343
f(105)=f( 13 00 00 )=  3.8368178570398
f(106)=f( 12 10 00 )= -22.229443822602
f(107)=f( 12 01 00 )= -3.8848262988042
f(108)=f( 12 00 10 )= 0.37028415274474
f(110)=f( 11 20 00 )=  131.06725911171
f(111)=f( 11 11 00 )=  22.273464262369
f(112)=f( 11 10 10 )=-0.33961141134684
f(114)=f( 11 02 00 )=  3.9575273822759
f(115)=f( 11 01 10 )=-0.56068899177234
f(117)=f( 11 00 20 )=  1.4683719073413
f(120)=f( 10 30 00 )= -784.81961275476
f(121)=f( 10 21 00 )= -130.08911462416
f(122)=f( 10 20 10 )= -7.5433910969757
f(124)=f( 10 12 00 )= -22.444934383721
f(125)=f( 10 11 10 )=  1.3223592663866
f(127)=f( 10 10 20 )= -7.5787483682171
f(130)=f( 10 03 00 )= -4.0568366193274
f(131)=f( 10 02 10 )= 0.76586488412865
f(133)=f( 10 01 20 )= -1.5822803164613
f(136)=f( 10 00 30 )= 0.87845155172029
f(140)=f( 04 00 00 )= 0.65476439654589
f(141)=f( 03 10 00 )= -3.8957712163317
f(142)=f( 03 01 00 )=-0.65243793289841
f(143)=f( 03 00 10 )=-1.79335782042371E-02
f(145)=f( 02 20 00 )=  23.551891656653
f(146)=f( 02 11 00 )=  3.8436116832359
f(147)=f( 02 10 10 )= 0.40222665433808
f(149)=f( 02 02 00 )= 0.65406265298300
f(150)=f( 02 01 10 )=-1.25336862486389E-02
f(152)=f( 02 00 20 )= 0.23495754879529
f(155)=f( 01 30 00 )= -144.44763007066
f(156)=f( 01 21 00 )= -23.024189289137
f(157)=f( 01 20 10 )= -4.0695905593703
f(159)=f( 01 12 00 )= -3.8140043148182
f(160)=f( 01 11 10 )=-0.22832995764225
f(162)=f( 01 10 20 )= -1.3410517136462
f(165)=f( 01 03 00 )=-0.65972297139373
```

```
  f(166)=f( 01 02 10 )= 4.36519629781583E-02
  f(168)=f( 01 01 20 )=-0.23997834123786
  f(171)=f( 01 00 30 )= 3.87228050188257E-02
  f(175)=f( 00 40 00 )=  897.75692378667
  f(176)=f( 00 31 00 )=  139.99369237471
  f(177)=f( 00 30 10 )=  34.349148324348
  f(179)=f( 00 22 00 )=  22.633592790097
  f(180)=f( 00 21 10 )=  3.0123846700303
  f(182)=f( 00 20 20 )=  8.1529289276729
  f(185)=f( 00 13 00 )=  3.8065681637423
  f(186)=f( 00 12 10 )= 5.73704205846520E-02
  f(188)=f( 00 11 20 )=  1.3184041675673
  f(191)=f( 00 10 30 )= 0.17464081928501
  f(195)=f( 00 04 00 )= 0.66959389314255
  f(196)=f( 00 03 10 )=-7.61197585044729E-02
  f(198)=f( 00 02 20 )= 0.25038528518346
  f(201)=f( 00 01 30 )=-8.02570740372245E-02
  f(205)=f( 00 00 40 )= 0.32029819392481

****************************************************************
response from moma for transformed moments produced by amap:   *
note that the eigen emittances are unchanged.                  *
****************************************************************

moment analysis

eigen emittances
ex=  1.150207882933154E-003
ey=  8.917312194344432E-004
et=  4.857225732735060E-017

****************************************************************
response from bgen for transfomed particle distribution:       *
note that the moments are the same as those produced by amap.  *
****************************************************************

  numerically computed values of selected moments
  values of <x*x>, <x*px>, <px*px>:
   25.6442225287797        4.02828338045640       0.672238183305473
  values of <y*y>, <y*py>, <py*py>:
   23.7254604524353        3.90393186429444       0.685029514737397
  values of <t*t>, <t*pt>, <pt*pt>:
  0.367180180818534       0.000000000000000E+000  0.000000000000000E+000

matrix for map is :

  2.56442E+01  4.02828E+00 -2.27002E+01 -4.14438E+00  8.95406E-01  0.00000E+00
  4.02828E+00  6.72238E-01 -3.94437E+00 -6.75547E-01  2.55200E-02  0.00000E+00
 -2.27002E+01 -3.94437E+00  2.37255E+01  3.90393E+00  3.11838E-01  0.00000E+00
 -4.14438E+00 -6.75547E-01  3.90393E+00  6.85030E-01 -7.31326E-02  0.00000E+00
  8.95406E-01  2.55200E-02  3.11838E-01 -7.31326E-02  3.67180E-01  0.00000E+00
  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
nonzero elements in generating polynomial are :

  f(  1)=f( 10 00 00 )=  1.6427969606320
  f(  2)=f( 01 00 00 )= 0.25337979304929
  f(  3)=f( 00 10 00 )= -1.4093360626378
  f(  4)=f( 00 01 00 )=-0.26258622059057
  f(  5)=f( 00 00 10 )= 7.10042331749088E-02
  f(  7)=f( 20 00 00 )=  25.644222528780
  f(  8)=f( 11 00 00 )=  4.0282833804564
  f(  9)=f( 10 10 00 )= -22.700172399453
  f( 10)=f( 10 01 00 )= -4.1443849789773
  f( 11)=f( 10 00 10 )= 0.89540566852679
  f( 13)=f( 02 00 00 )= 0.67223818330547
  f( 14)=f( 01 10 00 )= -3.9443679266310
  f( 15)=f( 01 01 00 )=-0.67554652714701
  f( 16)=f( 01 00 10 )= 2.55200353793911E-02
  f( 18)=f( 00 20 00 )=  23.725460452435
  f( 19)=f( 00 11 00 )=  3.9039318642944
  f( 20)=f( 00 10 10 )= 0.31183800881830
  f( 22)=f( 00 02 00 )= 0.68502951473740
  f( 23)=f( 00 01 10 )=-7.31326369560904E-02
  f( 25)=f( 00 00 20 )= 0.36718018081853
  f( 28)=f( 30 00 00 )=  57.763775979984
  f( 29)=f( 21 00 00 )=  7.8672760816525
  f( 30)=f( 20 10 00 )= -39.558751646963
  f( 31)=f( 20 01 00 )= -8.5851972495982
  f( 32)=f( 20 00 10 )=  5.5368629088634
  f( 34)=f( 12 00 00 )=  1.1619663150996
  f( 35)=f( 11 10 00 )= -6.2556059328187
  f( 36)=f( 11 01 00 )= -1.2255215456628
  f( 37)=f( 11 00 10 )= 0.49016886958874
  f( 39)=f( 10 20 00 )=  35.416088982804
  f( 40)=f( 10 11 00 )=  6.4189690085460
  f( 41)=f( 10 10 10 )= -1.2599372191073
  f( 43)=f( 10 02 00 )=  1.3109482802407
  f( 44)=f( 10 01 10 )=-0.65883685360157
  f( 46)=f( 10 00 20 )=  1.3007942318064
  f( 49)=f( 03 00 00 )= 0.18974694248181
  f( 50)=f( 02 10 00 )= -1.0978383283479
  f( 51)=f( 02 01 00 )=-0.19227538324372
  f( 52)=f( 02 00 10 )= 1.95028147046971E-02
  f( 54)=f( 01 20 00 )=  6.6424002684595
  f( 55)=f( 01 11 00 )=  1.0825830170125
  f( 56)=f( 01 10 10 )= 0.11764361858597
  f( 58)=f( 01 02 00 )= 0.19791209348012
  f( 59)=f( 01 01 10 )=-4.34729619365357E-02
  f( 61)=f( 01 00 20 )= 0.18486153487155
  f( 64)=f( 00 30 00 )= -41.702065675090
  f( 65)=f( 00 21 00 )= -6.3945034856785
  f( 66)=f( 00 20 10 )= -1.9118281919573
  f( 68)=f( 00 12 00 )= -1.0835478429965
  f( 69)=f( 00 11 10 )= 7.44097018049525E-03
  f( 71)=f( 00 10 20 )=-0.96466932108747
```

```
f( 74)=f( 00 03 00 )=-0.20687996272775
f( 75)=f( 00 02 10 )= 6.91619687513365E-02
f( 77)=f( 00 01 20 )=-0.19811725062739
f( 80)=f( 00 00 30 )= 0.10222992293842
f( 84)=f( 40 00 00 )=  987.42305940068
f( 85)=f( 31 00 00 )=  149.17441518208
f( 86)=f( 30 10 00 )= -817.14443250242
f( 87)=f( 30 01 00 )= -155.88960701602
f( 88)=f( 30 00 10 )=  51.788851508784
f( 90)=f( 22 00 00 )=  23.446959955018
f( 91)=f( 21 10 00 )= -132.18403407589
f( 92)=f( 21 01 00 )= -24.116981465482
f( 93)=f( 21 00 10 )=  5.1674426378288
f( 95)=f( 20 20 00 )=  760.01771966059
f( 96)=f( 20 11 00 )=  134.43688536022
f( 97)=f( 20 10 10 )= -17.374565641115
f( 99)=f( 20 02 00 )=  24.963016717637
f(100)=f( 20 01 10 )= -6.5247030079176
f(102)=f( 20 00 20 )=  10.467002421343
f(105)=f( 13 00 00 )=  3.8368178570398
f(106)=f( 12 10 00 )= -22.229443822602
f(107)=f( 12 01 00 )= -3.8848262988042
f(108)=f( 12 00 10 )= 0.37028415274476
f(110)=f( 11 20 00 )=  131.06725911171
f(111)=f( 11 11 00 )=  22.273464262369
f(112)=f( 11 10 10 )=-0.33961141134695
f(114)=f( 11 02 00 )=  3.9575273822759
f(115)=f( 11 01 10 )=-0.56068899177232
f(117)=f( 11 00 20 )=  1.4683719073412
f(120)=f( 10 30 00 )= -784.81961275476
f(121)=f( 10 21 00 )= -130.08911462416
f(122)=f( 10 20 10 )= -7.5433910969752
f(124)=f( 10 12 00 )= -22.444934383721
f(125)=f( 10 11 10 )=  1.3223592663865
f(127)=f( 10 10 20 )= -7.5787483682165
f(130)=f( 10 03 00 )= -4.0568366193274
f(131)=f( 10 02 10 )= 0.76586488412864
f(133)=f( 10 01 20 )= -1.5822803164614
f(136)=f( 10 00 30 )= 0.87845155172073
f(140)=f( 04 00 00 )= 0.65476439654589
f(141)=f( 03 10 00 )= -3.8957712163317
f(142)=f( 03 01 00 )=-0.65243793289841
f(143)=f( 03 00 10 )=-1.79335782042358E-02
f(145)=f( 02 20 00 )=  23.551891656654
f(146)=f( 02 11 00 )=  3.8436116832359
f(147)=f( 02 10 10 )= 0.40222665433807
f(149)=f( 02 02 00 )= 0.65406265298301
f(150)=f( 02 01 10 )=-1.25336862486411E-02
f(152)=f( 02 00 20 )= 0.23495754879530
f(155)=f( 01 30 00 )= -144.44763007066
f(156)=f( 01 21 00 )= -23.024189289137
f(157)=f( 01 20 10 )= -4.0695905593703
f(159)=f( 01 12 00 )= -3.8140043148183
```

```
  f(160)=f( 01 11 10 )=-0.22832995764224
  f(162)=f( 01 10 20 )= -1.3410517136462
  f(165)=f( 01 03 00 )=-0.65972297139373
  f(166)=f( 01 02 10 )= 4.36519629781559E-02
  f(168)=f( 01 01 20 )=-0.23997834123784
  f(171)=f( 01 00 30 )= 3.87228050187542E-02
  f(175)=f( 00 40 00 )=  897.75692378668
  f(176)=f( 00 31 00 )=  139.99369237471
  f(177)=f( 00 30 10 )=  34.349148324348
  f(179)=f( 00 22 00 )=  22.633592790097
  f(180)=f( 00 21 10 )=  3.0123846700303
  f(182)=f( 00 20 20 )=  8.1529289276730
  f(185)=f( 00 13 00 )=  3.8065681637423
  f(186)=f( 00 12 10 )= 5.73704205846661E-02
  f(188)=f( 00 11 20 )=  1.3184041675672
  f(191)=f( 00 10 30 )= 0.17464081928542
  f(195)=f( 00 04 00 )= 0.66959389314255
  f(196)=f( 00 03 10 )=-7.61197585044712E-02
  f(198)=f( 00 02 20 )= 0.25038528518347
  f(201)=f( 00 01 30 )=-8.02570740372923E-02
  f(205)=f( 00 00 40 )= 0.32029819392531

end of MARYLIE run
```

8.38 Compute Geometry of a Loop

Type Code: geom

Required Parameters:

1. IOPT
   = 1 to compute responses to Data Point elements.
   = 2 to compute local geometry.
   = 3 to do both.
   = 11 or 13 to do the same as for IOPT = 1 or 3, respectively, except that responses to Data Point elements are also put into the array *ucalc* for fitting purposes.
2. SI, initial path length in meters.
3. TI, initial time in seconds.
4. IPSET1
5. IPSET2
6. ISEND
   = 0 to not write out results.
   = 1 to write out results at the terminal.
   = 2 to write out results on file 12.
   = 3 to write out results at the terminal and on file 12.

This command also requires additional parameters whose values are specified by the parameter sets IPSET1 and IPSET2. Their contents are listed below.

Contents of IPSET1:

1. XI, initial x coordinate in meters.
2. YI, initial y coordinate in meters.
3. ZI, initial z coordinate in meters.
4. PHI, first Euler angle in degrees.
5. THETA, second Euler angle in degrees.
6. PSI, third Euler angle in degrees.

Contents of IPSET 2:

1. IFILE
   = file number (a positive integer) on which data is to be written in response to Data Point elements.
2. IMUCH
   = 0 to not write on file IFILE.

= 1 to write on file IFILE path-length and coordinate data in response to Data Point elements.

= 2 to write on file IFILE path-length and coordinate and orientation (local unit vectors) data in response to Data Point elements.

3. JFILE

   = file number (positive integer) on which local geometry information is to be written.

4. JMUCH

   = 0 to not write on file JFILE.

   = 1 to write on file JFILE simple element description and path-length data.

   = 2 to write on file JFILE simple element description and path-length and coordinate information.

   = 3 to write on file JFILE all local geometry information.

5. ISTART

   = positive integer to start writing responses to the Data Point elements (when IOPT = 11 or 13) at location ISTART in the array *ucalc*. (Note: ISTART must be ≥ 1. If not, MaryLie will set ISTART = 1.) Since 14 items (ss, x, y, z, tt, and the three unit vectors $ex\ell$, $ey\ell$, $ez\ell$) are written to *ucalc* in response to each *dp* element, successive write locations are offset by 14.

6. NPIECES

Example:

```
layout     geom
2 , 0 , 0 , 1 , 2 , 0
```

This specifies a command with the user given name *layout*. When invoked it computes the local geometry of the most recently invoked loop, takes additional data from parameter set 1, and writes data into a file and in a manner specified by the data in parameter set 2. For examples of the use of the *geom* command, see sections 10.12 and 10.13.

Description:

When invoked in *#labor*, a command with type code *geom* examines the *curent loop*. (The current loop is the most recently invoked loop in *#labor*. See section 5.10.) This command writes on two files, IFILE and JFILE. These files are not designed for human consumption, but rather are to be used by auxiliary programs such as Poster. (See section 1.4.2.) Their content is described below. Information for human use can be written to the terminal and file 12.

As it scans the entries in the current loop, it writes to IFILE whenever it encounters an element with type code *dp*. (See section 6.26.) The data it writes consists of geometrical information that can be merged with other quantities also computed by MaryLie to make

lattice function and related plots. For examples, see section 10.9 and sections 10.11 through 10.13.

As it scans the entries in the current loop it writes to file JFILE at the beginning and end of every element and relevant command. (It also writes data within elements if NPIECES is greater than one. For example, if NPIECES = 2, it writes at the beginning, middle, and end of each element.) The data it writes consists of element and geometrical information that is used to make composite lattice function and related plots with geometry, or layout drawings. For examples, see sections 10.13 and 10.14.

When writing to file IFILE, the amount of information written depends on the value of IMUCH. Shown below are quantities written when IMUCH = 1 or 2. In this example, the current loop has the contents

```
short
    1*dimoff      1*dp          1*bend        1*dp          1*drs      &
    1*dimon       1*hfq
```

See Exhibit 8.38.

IMUCH = 1

```
 0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08  0.00000E+00
```

A single line is written each time an element with type code *dp* is encountered. The first five entries in each line contain path length; x, y, and z coordinates; and accumulated time of flight values. The last entry contains the value zero.

IMUCH = 2

```
 0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
  1.00000000000000       0.000000000000000E+000  0.000000000000000E+000
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
 0.000000000000000E+000  0.000000000000000E+000   1.00000000000000
 2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08  0.00000E+00
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
```

Four lines are written each time an element with type code *dp* is encountered. The first five entries in the first line contain path length; x, y, and z coordinates; and accumulated time of flight values. The last entry contains the value zero. The second line contains the components of the local $\boldsymbol{e}_x$ unit vector. The third line contains the components of the local $\boldsymbol{e}_y$ unit vector. The fourth line contains the components of the local $\boldsymbol{e}_z$ unit vector.

When writing to file JFILE, the amount of information written depends on the value of JMUCH. Show below are quantities written when JMUCH = 1, 2, or 3.

JMUCH = 1

```
dp       dp           0     1    23
dp          0.0000    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
dp        1 23  0.00000E+00
bend     pbnd         4     1     3
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
bend        2.5495    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
bend      1  3  0.00000E+00
bend      1  3  8.49827E-01
bend      1  3  1.69965E+00
bend      1  3  2.54948E+00
dp       dp           0     1    23
dp          0.0000    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
dp        1 23  2.54948E+00
drs      drft         1     1     1
  0.450000000000000
drs         0.4500    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
drs       1  1  2.54948E+00
drs       1  1  2.69948E+00
drs       1  1  2.84948E+00
drs       1  1  2.99948E+00
hfq      quad         4     1     9
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
hfq         0.5000    2.7200
      1.0000      2.0000      3.0000      4.0000      5.0000      6.0000
hfq       1  9  2.99948E+00
hfq       1  9  3.16615E+00
hfq       1  9  3.33281E+00
hfq       1  9  3.49948E+00
```

Each geometrically relevant element produces a set of printed "header" lines describing it. The first such line gives the name, type code, number of parameters, and two internal MaryLie integers associated with the type code. Subsequent lines provide the parameter values if the number of parameters is nonzero. Next there is a line that repeats the element command name and gives its length and gradient. Finally there is a line that specifies the "dimensions" that are to be associated with this element. (These dimensions are set by previously invoking a command with type code *dims*. See section 7.40.)

The descriptive header lines are followed by lines giving geometrical information. For the case JMUCH = 1, this line repeats the element name and the two internal MaryLie integers, and then gives the accumulated path length. If the entity is an element that can be subdivided (has a length), the information described above is provided at the beginning and end of each element, and at intermediate points if NPIECES > 1. In this example NPIECES = 3.

JMUCH = 2

```
dp       dp           0     1    23
```

```
dp          0.0000    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
dp        1 23  0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00
bend     pbnd         4     1     3
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
bend        2.5495    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
bend      1  3  0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00
bend      1  3  8.49827E-01  1.39113E+01  0.00000E+00  8.43627E-01  3.36981E-09
bend      1  3  1.69965E+00  1.36492E+01  0.00000E+00  1.65038E+00  6.73962E-09
bend      1  3  2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08
dp       dp           0     1    23
dp          0.0000    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
dp        1 23  2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08
drs      drft         1     1     1
  0.450000000000000
drs         0.4500    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
drs       1  1  2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08
drs       1  1  2.69948E+00  1.31369E+01  0.00000E+00  2.50636E+00  1.07042E-08
drs       1  1  2.84948E+00  1.30487E+01  0.00000E+00  2.62772E+00  1.12990E-08
drs       1  1  2.99948E+00  1.29606E+01  0.00000E+00  2.74907E+00  1.18938E-08
hfq      quad         4     1     9
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
hfq         0.5000    2.7200
      1.0000      2.0000      3.0000      4.0000      5.0000      6.0000
hfq       1  9  2.99948E+00  1.29606E+01  0.00000E+00  2.74907E+00  1.18938E-08
hfq       1  9  3.16615E+00  1.28626E+01  0.00000E+00  2.88391E+00  1.25547E-08
hfq       1  9  3.33281E+00  1.27646E+01  0.00000E+00  3.01874E+00  1.32156E-08
hfq       1  9  3.49948E+00  1.26667E+01  0.00000E+00  3.15358E+00  1.38764E-08
```

The descriptive header lines are as before for JMUCH = 1. However, the geometrical information lines now contain, in addtion to accumulated path length, the x, y, and z coordinates and the accumulated time of flight.

JMUCH = 3

```
dp       dp           0     1    23
dp          0.0000    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
dp        1 23  0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00
  1.00000000000000       0.000000000000000E+000  0.000000000000000E+000
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
 0.000000000000000E+000  0.000000000000000E+000   1.00000000000000
bend     pbnd         4     1     3
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
bend        2.5495    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
```

```
bend      1  3  0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00
  1.00000000000000       0.000000000000000E+000  0.000000000000000E+000
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
 0.000000000000000E+000  0.000000000000000E+000   1.00000000000000
bend      1  3  8.49827E-01  1.39113E+01  0.00000E+00  8.43627E-01  3.36981E-09
 0.978147600733806       0.000000000000000E+000  0.207911690817759
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.207911690817759       0.000000000000000E+000  0.978147600733806
bend      1  3  1.69965E+00  1.36492E+01  0.00000E+00  1.65038E+00  6.73962E-09
 0.913545457642601       0.000000000000000E+000  0.406736643075800
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.406736643075800       0.000000000000000E+000  0.913545457642601
bend      1  3  2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
dp       dp           0     1    23
dp          0.0000    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
dp        1 23  2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
drs      drft         1     1     1
  0.450000000000000
drs         0.4500    0.0000
      0.0000      0.0000      0.0000      0.0000      0.0000      0.0000
drs       1  1  2.54948E+00  1.32251E+01  0.00000E+00  2.38501E+00  1.01094E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
drs       1  1  2.69948E+00  1.31369E+01  0.00000E+00  2.50636E+00  1.07042E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
drs       1  1  2.84948E+00  1.30487E+01  0.00000E+00  2.62772E+00  1.12990E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
drs       1  1  2.99948E+00  1.29606E+01  0.00000E+00  2.74907E+00  1.18938E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
hfq      quad         4     1     9
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
hfq         0.5000    2.7200
      1.0000      2.0000      3.0000      4.0000      5.0000      6.0000
hfq       1  9  2.99948E+00  1.29606E+01  0.00000E+00  2.74907E+00  1.18938E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
hfq       1  9  3.16615E+00  1.28626E+01  0.00000E+00  2.88391E+00  1.25547E-08
```

```
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
hfq       1  9  3.33281E+00  1.27646E+01  0.00000E+00  3.01874E+00  1.32156E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
hfq       1  9  3.49948E+00  1.26667E+01  0.00000E+00  3.15358E+00  1.38764E-08
 0.809016994374947       0.000000000000000E+000  0.587785252292473
 0.000000000000000E+000   1.00000000000000       0.000000000000000E+000
-0.587785252292473       0.000000000000000E+000  0.809016994374947
```

The descriptive header lines are as before for JMUCH = 1 or 2. There are also geometrical information lines that are identical to those for JMUCH = 2. Finally, following each such line, there are three lines that contain the components of the local $\boldsymbol{e}_x$, $\boldsymbol{e}_y$, and $\boldsymbol{e}_z$ unit vectors.

If IOPT = 11 or 13, responses to Data Point elements are also put in the array *ucalc* for fitting purposes. See Exhibit 8.38.

```
Exhibit 8.38
***MARYLIE 3.0***
Prerelease Development Version 1/12/00
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 8.38
      This is a MaryLie run that tests the geom type code.  It writes
 geometrical information for the loop short to various files and the
 ucalc array.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hdq      quad
  0.500000000000000      -1.92000000000000      0.000000000000000E+00
```

```
  0.000000000000000E+00
 hcs      sext
  0.500000000000000      0.000000000000000E+00
 vcs      sext
  0.500000000000000      0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 dimoff   dims
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 dimon    dims
   1.00000000000000       2.00000000000000       3.00000000000000
   4.00000000000000       5.00000000000000       6.00000000000000
 dp       dp
 wuca     wuca
   1.00000000000000       30.0000000000000       3.00000000000000
   40.0000000000000
 geom     geom
   13.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 ps1      ps1
   14.0000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 ps2      ps2
   85.0000000000000       2.00000000000000       90.0000000000000
   3.00000000000000       1.00000000000000       3.00000000000000
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
 cgeom
     1*ps1         1*ps2         1*geom
 short
     1*dimoff      1*dp          1*bend        1*dp          1*drs      &
     1*dimon       1*hfq
#lumps
#loops
 lshort
     1*short
```

```
#labor
    1*fileout
    1*lshort
    1*cgeom
    1*wuca
    1*fin

****************************
* Response to wuca command.*
****************************

 k and ucalc(k) for nonzero values of array
          2   14.0000000000000
          6   1.00000000000000
         10   1.00000000000000
         14   1.00000000000000
         15   2.54947999999612
         16   13.2250628791358
         18   2.38501121922983
         19  1.010942404416590E-008
         20  0.809016994374947
         22  0.587785252292473
         24   1.00000000000000
         26 -0.587785252292473
         28  0.809016994374947

end of MARYLIE run
```

8.39 Copy File to Working Array

Type Code: fwa

Required Parameters:

1. IFILE (file number from which data is to be read).
2. NCOL (number of columns in the file).
3. LINES (number of lines in the file).

Example:

```
getdata     fwa
   18 , 6 , 2
```

This specifies a command with the user given name *getdata*. When invoked, the 6 columns of the first 2 lines of file 18 are read in to a dedicated working array.

Description:

The *usr* routines (see section 6.20 and 6.21) have access to the array *wa*. A command with type code *fwa* can be used to place data from an external file into this array.

8.40 Merge Files

Type Code: merf

Required Parameters:

1. INFILE1 (number of first file).
2. INFILE2 (number of second file).
3. IPSET (number of parameter set describing files 1 and 2).
4. NOUTFILE (number of output file).

This command also requires additional parameters whose values are specified by the parameter set IPSET. Its contents are listed below.
Contents of IPSET:

1. NCOL1 (number of columns in first file).
2. ICOL1 (starting column in first file).
3. JCOL1 (ending column in first file).
4. NCOL2 (number of columns in second file).
5. ICOL2 (starting column in second file).
6. JCOL2 (ending column in second file).

Example:

```
merge     merf
   15 , 17 , 1 , 18
```

This specifies a command with the user given name *merge*. When invoked it takes the data in columns ICOL1 through JCOL1 in file INFILE1 and in columns ICOL2 through JCOL2 in INFILE2 and writes them as columns 1 through (JCOL1 − ICOL1 + JCOL2 − ICOL2) in the file NOUTFILE.

Description:

Sometimes, say for purposes of graphing, it is useful to merge data from two different files. A command with type code *merf* can be used for this purpose. The result is a file whose first (JCOL1 − ICOL1) columns are the contents of columns ICOL1 through JCOL1 in the first file and whose remaining (JCOL2 − ICOL2) columns are the contents of columns ICOL2 through JCOL2 in the second file.

8.41 Compute Logarithm of Normal Form

Type Code: lnf

Required Parameters:

1. JOB
   = 1 to compute logarithm of static normal form.
   = 2 to compute logarithm of dynamic normal form.
2. IOPT
   = 1 to compute only f_2 part of Lie generator.
   = 2 to compute full Lie generator.
3. Multiplier
   = 1 for unit multiplication.
   = 2 to multiply Lie generator by $(-1/\pi)$.

Example:

```
log     lnf
1 , 1 , 2
```

This specifies a command with the user given name *log*. When invoked, it computes the logarithm of a static normal form (f_2 part only) and multiplies it by $(-1/\pi)$.

Description:

Let $\mathcal{N}$ be a map in normal form. On occasion it is useful to compute the Lie generator f such that $\mathcal{N} = \exp : f :$. Such an occasion is that of determining full tunes (including integer parts) of a ring. See Exhibit 10.2.2. After a command with type code *lnf* is executed, buffer 1 contains in its array part the Lie generator for $\mathcal{N}$ (multiplied by the multiplier) and the unit matrix is its matrix part. The remaining buffers are empty.

Chapter 9

Catalog of Procedures and Fitting and Optimization Commands

MaryLie 3.0 at present has a fairly extensive repertoire of procedures and fitting and optimization commands. Their type code mnemonics and the subsections that describe them in more detail are listed below. Some examples of the use of procedures and fitting commands are given in chapter 10.

Type Code	Procedure/Command	Subsection
bip	Begin inner procedure.	9.1
bop	Begin outer procedure.	9.2
tip	Terminate inner procedure.	9.3
top	Terminate outer procedure.	9.4
aim	Specify quantities to be fit or optimized and set target values.	9.5
vary	Specify quantities to be varied.	9.6
fit	Carry out fitting operation.	9.7
mss	Minimize sum of squares optimization.	9.8
opt	General optimization.	9.9
mrt0	Merit function (sum of squares).	9.10
mrt1 ⋮ mrt5	Merit functions (user written).	9.11
con1 ⋮ con5	Constraints.	9.12
grad	Compute gradient matrix.	9.13

Type Code	Procedure/Command	Subsection
scan	Scan parameter space.	9.14
cps1 ⋮ cps9	Capture parameter set.	9.15
fps	Free parameter set.	9.16
flag	Change or write out values of flags and defaults.	9.17
rset	Reset menu entries.	9.18
—	Spare.	9.19

Note that the type codes are given in lower case. If entries are made in upper case, they are automatically converted to lower case by PREP and MaryLie.

The purpose of this section is to outline the use of these procedures and commands by MaryLie, and to describe the parameters required to specify them in the #menu component of the Master Input File.

9.1 Begin Inner Procedure

Type Code: bip

Required Parameters:

1. NTIMES (number of times procedure is to be executed).

Example:

```
startido    bip
  5
```

This specifies a command with the user given name *startido*. When invoked it marks the beginning of an inner procedure that is to be carried out 5 times.

Description:

It is possible to set up a logical "do loop" (procedure) within the *labor* component of a MaryLie master input file. This loop begins with a *bip* command and ends with a *tip* command. Such loops are useful for fitting or optimization or scanning or other purposes. See, for examples, sections 10.2.1, 10.3.1, and 10.4.

9.2 Begin Outer Procedure

Type Code: bop

Required Parameters:

1. NTIMES (number of times procedure is to be executed).

Example:

```
startodo     bop
    5
```

This specifies a command with the user given name *startodo*. When invoked it marks the beginning of an outer procedure that is to be carried out 5 times.

Description

It is possible to set up two nested logical "do loops" (procedures) within the *labor* component of a MaryLie master input file. The inner loop begins with a *bip* command and ends with a *tip* command. The outer surrounding loop begins with a *bop* command and ends with a *top* command. This flexibility makes it possible to fit in an inner loop while optimizing or scanning in an outer loop. See, for examples, sections 10.3.2 and 10.3.3.

9.3 Terminate Inner Procedure

Type Code: tip

Required Parameters:

1. IOPT
   = 0 to not run interactively.
   = 1 to allow user to terminate an inner procedure interactively.

Example:

```
stopido     tip
   0
```

This specifies a command with the user given name *stopido*. It marks the end of an inner loop.

Description:

As described in section 9.1, a *tip* command marks the end of an inner logical "do loop". When this command is reached in the *labor* component of the master input file and IOPT = 0, MaryLie returns to the *bip* command at the beginning of the procedure unless NTIMES has been achieved or some other (perhaps fitting or optimization) criterion has been met. If IOPT = 1, the user is asked whether or not to make another pass through the procedure.

9.4 Terminate Outer Procedure

Type Code: top

Required Parameters:

1. IOPT
   = 0 to not run interactively.
   = 1 to allow user to terminate an outer procedure interactively.

Example:

```
stopodo     top
   1
```

This specifies a command with the user given name *stopodo*. It marks the end of an outer loop.

Description:

As described in section 9.2, a *top* command marks the end of an outer logical "do loop". When this command is reached in the *labor* component of the master input file and IOPT = 0, MaryLie returns to the *bop* command at the beginning of the procedure unless NTIMES has been achieved or some other (perhaps optimization or scanning) criterion has been met. If IOPT = 1, the user is asked whether or not to make another pass through the procedure.

9.5 Specify Quantities to be Fit or Optimized and Set Target Values

Type Code: aim

Required Parameters:

1. JOB
   = 1 to specify aims only.
   = 2 to specify aims and set targets.
   = 3 to specify aims and set targets and weights.
2. INFILE (File number from which specifications are to be read.)
   = 0 or 5 to read interactively from the terminal.
   = IFILE (any positive integer) to read from that file, after rewinding that file.
   = −IFILE (any negative integer) to read from file IFILE at its current position without first rewinding.
3. LOGFILE (File number to which specifications are to be written. Ignored unless INFILE = 0 or 5, but still required.)
   = 0 to not write out specifications.
   = JFILE (any positive integer) to write specifications to that file.
4. IQUIET
   = 0 for no effect.
   = 1 to run in quiet mode.
5. ISTORE
   = 1 to put selections in storage associated with inner procedure.
   = 2 to put selections in storage associated with outer procedure.
   = 3 to put selections in auxiliary storage.
6. ISEND (Ignored, but still required, when INFILE = 0 or 5.)
   = 0 to write out nothing.
   = 1 to write out aims at the terminal.
   = 2 to write out aims on file 12.
   = 3 to write out aims at the terminal and on file 12.

Example:

```
desires     aim
  2 1 25 0 1 0
```

This specifies a command with the user given name *desires*. When invoked it specifies aims and sets targets using the contents of file 1 and puts these selections in storage associated with an inner procedure.

Description:

The *aim* command may be used for fitting and optimization procedures as well as the construction of merit functions. It gives the user access to all quantities computed by MaryLie and the ability to set target values for any subset of these quantities. This can be done interactively at run time or in advance by the use of external files. A convenient procedure is to run interactively at first, and then write whatever instructions were employed onto a log file for subsequent automatic use. For examples of the use of *aim*, see sections 10.2.1, 10.3, and 10.4 through 10.6. The Exhibit below shows the responses to *aim* when invoked interactively and help is requested.

```
#comment
  Exhibit 9.5.
  This is a fragmentary MaryLie run to illustrate the initial responses
  to a command with type code aim when invoked interactively and help
  is requested.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  aim      aim
    2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*aim
     1*end

  Enter aim(s) in the form:  symbol = target value
  (Enter a # sign when finished, or type help to review the defined

       symbols)
help

  Symbols are defined for the following categories:
   1. Path length and related quantities, the current map,
      various stored maps, and work buffers in map format.
   2. Standard quantities derived from the current map including
      dispersions and phase slip factors, tunes, chromaticities,
      anharmonicities, twiss parameters, and envelope data.
   3. Particle and Beam parameters.
```

```
   4. User calculated quantities, merit functions, miscellany.
   -----Select help category-----
1
                  ***  Quantities in Map Format  ***

   The current transfer map:
      pl, rt, and st = path length, (real) time of
      flight, and scaled time of flight, respectively.
      r(i,j) = first order (linear) matrix elements i,j=1,6.
      f(i) = Lie polynomial coefficients, i=0,209.

   Stored maps in locations 1 through 9:
      sm1(i,j) ... sm9(i,j) = first order (linear) matrix elements

      i,j=1,6.
      sf1(i) ... sf9(i) = Lie polynomial coefficients, i=0,209.

   Result arrays in buffers 1 through 6:
      bm1(i,j) ... bm6(i,j) = first order (linear) matrix elements

      i,j=1,6.
      bf1(i) ... bf6(i) = Lie polynomial coefficients, i=0,209.

  Enter aim(s) in the form:  symbol = target value
  (Enter a # sign when finished, or type help to review the defined

  symbols)
help

  Symbols are defined for the following categories:
   1. Path length and related quantities, the current map,
      various stored maps, and work buffers in map format.
   2. Standard quantities derived from the current map including
      dispersions and phase slip factors, tunes, chromaticities,
      anharmonicities, twiss parameters, and envelope data.
   3. Particle and Beam parameters.
   4. User calculated quantities, merit functions, miscellany.
   -----Select help category-----
2
       ***  Standard quantities computed by COD, TASM, and TADM ***

  Dispersions and Phase Slip Factors d(i,m):
    d(i,m) = mth order expansion coefficient of the
             ith closed orbit coordinate for m=1,3
             and i=1,4.
    d(5,m) = mth order scaled time of flight
             expansion coefficient for m=1,3.
    d(6,m) = mth order path length expansion
             coefficient for m=1,3.
    d(i,4) = Summed net deviation in the ith closed
             orbit coordinate for the specified
             value of epsilon.
    d(5,4) = Summed net deviation in scaled time of
```

```
          flight for the specified value of
          epsilon.
  d(6,4) = Summed net deviation in path length for
          the specified value of epsilon.
  d(5,5) = Total value of the scaled time of
          flight for the specified value of
          epsilon.
  d(6,5) = Total value of the path length for
          the specified value of epsilon.
  d(5,6) = Total value of the scaled time of
          flight for the design orbit
          (i.e. when epsilon=0).
  d(6,6) = Total value of the path length for
          the design orbit (i.e. when epsilon=0).

Design Tunes:
   tx,ty,ts = horizontal,vertical,synchrotron (temporal) tunes.

Chromaticities:
   cx, cy = horizontal and vertical 1st order chromaticities.
   qx, qy = horizontal and vertical 2nd order (quadratic)

            chromaticities.

Anharmonicities:
   hh, vv, tt, hv, ht, vt = 2nd order anharmonicities.

Betatron amplitudes ba(i,j):
   i = 1,2,3 for horizontal, vertical, and temporal.
   j = 1,2 for 1st and 2nd powers.

True Tunes tn(i,j):
   i = 1,2,3 for horizontal, vertical, and temporal.
   j = 1,2,3 for design, chromatically corrected
       zero amplitude, and finite amplitude.

Twiss parameters tp(i,j):
   j = 1,2,3 for alpha,beta,gamma.
   When computed by TADM:
   i = 1,2,3 for horizontal, vertical, and temporal.
       e.g., tp(1,2) = horizontal beta, etc.
   When computed by TASM:
   i = 1,2, for on energy/momentum horizontal, vertical.
   i = 3,4, for actual horizontal, vertical.
       e.g., tp(1,2) = on energy/momentum horizontal beta, etc.;
             tp(3,2) = actual horizontal beta, etc.

Envelope coefficients and envelopes en(i,j):
   i = 1,2,3 for horizontal, vertical, and temporal.
   j = 1,2,3 for horizontal, vertical, and temporal.
     = 4 for finite amplitude envelopes;
       e.g., en(1,2) = xevc, en(1,4) = xe, etc.
```

```
  Enter aim(s) in the form:  symbol = target value
  (Enter a # sign when finished, or type help to review the defined
  symbols)
help

  Symbols are defined for the following categories:
   1. Path length and related quantities, the current map,
      various stored maps, and work buffers in map format.
   2. Standard quantities derived from the current map including
      dispersions and phase slip factors, tunes, chromaticities,
      anharmonicities, twiss parameters, and envelope data.
   3. Particle and Beam parameters.
   4. User calculated quantities, merit functions, miscellany.
   -----Select help category-----
3
       ***  Particle and Beam Parameters  ***

  Design Beam Parameters:
     ek = design kinetic energy,
     pd = design momentum,
     br = B*rho.

  Relativistic factors:
     be = beta,
     ga = gamma,
     gm = gamma-1.

  Particle coordinates from tracking:
     z(i,j) = ith component of jth particle in tracking buffer.

  Moments and Emittances:
     These are stored in various buffers.
     See item 1 above.

  Enter aim(s) in the form:  symbol = target value
  (Enter a # sign when finished, or type help to review the defined
  symbols)
help

  Symbols are defined for the following categories:
   1. Path length and related quantities, the current map,
      various stored maps, and work buffers in map format.
   2. Standard quantities derived from the current map including
      dispersions and phase slip factors, tunes, chromaticities,
      anharmonicities, twiss parameters, and envelope data.
   3. Particle and Beam parameters.
   4. User calculated quantities, merit functions, miscellany.
   -----Select help category-----
4
    ***  User defined and miscellaneous quantities  ***
    ***  computed by USER and MERIT routines, etc.  ***

     u(i) = ucalc(i) for i=1 to 1024, stored in common/usrdat/
```

```
     umi = val(i) for i=1 to 5, computed by user merit functions
           MRT1 through MRT5 and stored in common/merit/

     lsi = val(i) for i=1 to 3, for least squares merit function
           specified by earlier calls to aim

     ex(i) = contents of extra array for i=1 to 1 to 5:
        ex(1) = momentum compaction,
        ex(2) = transition gamma or its square if imaginary,
        ex(3) = polynomial scalar product,
        ex(4) = symplectic violation,
        ex(5) = matrix norm.

     Quantities associated with ppa:
        fx,fy = ?
        xb,yb = ?
        xa,ya = ?
        xu,yu = ?
        xd,yd = ?

  Enter aim(s) in the form:  symbol = target value
  (Enter a # sign when finished, or type help to review the defined
  symbols)
#

  Do you wish to start over? (y/n) <n>:
n

end of MARYLIE run
```

9.6 Specify Quantities to be Varied

Type Code: vary

Required Parameters:

1. JOB (Controls relation between NV, the number of quantities to be varied, and NA, the number of aims set in *aim*).
   = 1 if NV = NA. (This condition is required for a fitting process.)
   = 2 if NV $\leq$ NA. (This condition is required for a least squares optimization.)
   = 3 if NV is not constrained.
   = −N, with N=1, 2, or 3. These values cause the same restrictions as above on the number of independent variables NV. However, they also allow any number of dependent variables to be attached to the independent variables in a linear way.
2. INFILE (File number from which specifications are to be read).
   = 0 or 5 to read interactively from the terminal.
   = IFILE (any positive integer) to read from that file, after rewinding that file.
   = −IFILE (any negative integer) to read from file IFILE at its current position without first rewinding.
3. LOGFILE (File number to which specifications are to be written. Ignored unless INFILE = 0 or 5, but still required.)
   = 0 to not write out specifications.
   = JFILE (any positive integer) to write specifications to that file.
4. ISB (Scaling and Bounds).
   = 0 for default (unit) scaling and no bounds.
   = 1 to scan command lines for override scale factors (e.g. Amps/Tesla, or simply more uniform size) for each independent variable as used in *fit*, *opt*, *grad*, and *scan*. The dependent variables, if any, are still computed from the independent ones in the original MKS units.
   = 2 to scan command lines for bounds XMIN,XMAX for each independent variable as used in *fit*, *opt*, *grad*, and *scan*. Note that bounds for the dependent variables, if any, are implied by the bounds on the independent variables to which they are attached.
   = 3 to scan command lines for both scales and bounds.
5. ISTORE
   = 1 to put selections in storage associated with inner procedure.
   = 2 to put selections in storage associated with outer procedure.
   = 3 to put selections in auxiliary storage.

6. ISEND (Ignored, but still required, when INFILE = 0 or 5).
   = 0 to write out nothing.
   = 1 to write out at the terminal items to be varied.
   = 2 to write out on file 12 items to be varied.
   = 3 to write out items at the terminal and on file 12.

Example:

```
adjust    vary
 1 1 27 0 1 0
```

This specifies a command with the user given name *adjust*. When invoked it specifies the quantities to be varied using the contents of file 1 and puts these selections in storage associated with an inner procedure.

Description:

The *vary* command may be used for fitting, optimization, and scanning. It gives the user access to all parameters of all items in the *menu* component of the MaryLie master input file. See, for example, sections 10.2.1 and 10.3 through 10.6. Note that if JOB < 0, one may have NV *independent* variables and any number of *dependent* variables whose *increments* are tied to those of the independent variables by a linear relation. For example, see section 10.3.1.

The Exhibit below shows the MaryLie response to *vary* when invoked interactively. It presents in alphabetical order the user-given names of all entries in the *menu* component of the master input file, and invites selection.

```
#comment
  Exhibit 9.6.
  This is a fragmentary MaryLie run to illustrate the initial response
  to a command with type code vary.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  vary     vary
    3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000       1.00000000000000
  drft     drft
    2.00000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
```

```
     1*vary
     1*end
MARYLIE #menu entries available to be varied:
----------------------------------------------------------------------
drft     end      fileout  vary

Enter name and (optional) parameter index for #menu element(s) to be
varied.
Elements named following a $ may be reset only.  Type * to relist
Select up to 100 element(s): (Enter a # sign when finished)
```

9.7 Carry Out Fitting Operation

Type Code: fit

Required Parameters:

1. JOB
   = 1 to use standard fitting algorithm and inner procedure storage.
   = −1 to use standard fitting algorithm and outer procedure storage.
   = N (N an integer > 1). In this case inner procedure storage is used, and the meaning of N is determined by the sign of MPRINT. If MPRINT = M (M an integer ≥ 1), the fitter halves the reach up to N−1 times if there are convergence difficulties. See the description below. If MPRINT = −M (M an integer ≥ 1), the fitter *damps* the step that Newton would prescribe for the first N steps after the feeler steps. See the description below.
   = −N (N an integer > 1) to use outer procedure storage, but otherwise act as in the +N case described above.
2. AUX
   = 0 to use default value of ANTOL = 10^{-9} when MPRINT = M with M > 0, and to use the default damping coefficient value of (1/2) when MPRINT = −M. See the description below.
   > 0 to set ANTOL = AUX when MPRINT = M, and to use AUX for the damping coefficient when MPRINT = −M. See the description below.
3. ERROR
4. DELTA (Controls feeler step size. See the description below.)
5. MPRINT
   = 0 for no output.
   = +M to allow halvings of the reach if there are convergence problems and to print intermediate aims and variables every Mth iteration. See the description below.
   = −M to forbid halvings of the reach if there are convergence problems, and instead damp the feeler steps. Also, print intermediate aims and variables every Mth iteration. See the description below.
6. ISEND
   = 0 to write out nothing.
   = 1 to write brief output at the terminal.
   = 2 to write brief output on file 12.
   = 3 to write brief output both at the terminal and on file 12.
   = 4 to write brief output at the terminal and extensive output on file 12.
   = −J (with J = 1,2,3,4) to achieve the same results as above except the words “brief” and “extensive” are interchanged.

Example:

```
fitit    fit
1  0  1.e-15  1.e-3  1  1
```

This specifies a command with the user given number *fitit*. When invoked it seeks to achieve with an error of less than 10^{-15} the goals specified in a previous *aim* command.

Description:

A *fit* command is used inside a procedure (a logical loop) in conjunction with *aim* and *vary* commands. See, for example, sections 10.2.1, 10.3, 10.4.1, 10.5, and 10.6. The fitter employed in MaryLie uses a modified Newton's method. It usually displays (particularly once a fit is well underway) nearly the quadratic convergence of Newton's method, but is considerably faster.

To facilitate further explanation, it is useful to introduce two definitions. The *aim* command specifies NA quantities to be fit and a target value for each of these quantities (see section 9.5). We view any collection of values for these quantities as a point in an NA-dimensional "*aim*" space. Thus the set of target values is a particular point in *aim* space, which we will call the *target* point. Similarly, the *vary* command specifies NV (with NV = NA) parameters to be varied (see section 9.6). We view any collection of values for these parameters as a point in an NV-dimensional "*vary*" space. Thus, the *initial* set of parameter values before a *fit* has begun is a particular point in vary space. Corresponding to this point in vary space is an initial point in aim space that describes the values of the quantities to be fit when the parameters have their initial values. The goal of a fit command is to find a point in vary space that will produce a corresponding point in aim space that is the desired target point.

When invoked, the fitter begins by making NV independent *feeler* steps in vary space by varying the selected parameters one at a time. If the parameter to be varied is initially zero, the corresponding feeler step has size DELTA. If the parameter is initially nonzero, the corresponding feeler step has size DELTA*(Initial Value of Parameter). For each feeler step to a new point in vary space in the vicinity of the initial point, the corresponding point in aim space is found. Thus, at the end of the feeler step process there are (NV + 1) points in vary space (the initial point + NV feeler-step points) and (NV + 1) corresponding points in aim space. Next the fitter makes a *Newton* step in vary space to a point whose corresponding point in aim space is supposed to be nearer the target point than the initial point in aim space. It also finds the corresponding point in aim space.

At this stage there are (NV + 2) points in vary space and (NV + 2) corresponding points in aim space. The usual Newton procedure would now be to discard all of these points save for the result of the Newton step, and then again repeat the original process by again making NV feeler steps, etc.

The fitter in MaryLie does something else: One of these (NV + 2) points in aim space will be farthest from the target point. The corresponding point in vary space is discarded so that what remains is a set of (NV + 1) points. These points are now used to make a second Newton step, etc. The process is repeated until convergence is achieved in aim space. By

convergence it is meant that a point is found in vary space whose corresponding point in aim space is a distance less than ERROR away from the target point.

It may happen that successive points found by the fitter in aim space do not seem to be converging to the target point. The fitter senses this problem if it occurs. Suppose JOB = ±N with N > 1 and MPRINT = M with M > 0. The fitter then selects the point in aim space that is midway between the initial point and the target point, and treats this point as a new *interim target*. This process is called *halving* the *reach*. The fitter now begins its operation anew trying to achieve the interim target. If convergence problems continue, a new interim target is selected that is midway between the initial point and the previous interim target. A maximum of (N− 1) halvings of the reach is allowed if convergence problems persist. Thus, the aim-space journey from the initial point to the target point can be subdivided up to (N − 1) times. On the other hand, if convergence is achieved for two successive Newton steps, the reach is *doubled* (provided that so doing does not produce an interim target that is beyond the original target).

The whole problem resembles that of getting a dog to move from point A to point B. One places a wiener before its nose at point A, and then drags the wiener to point B in the hope that the dog will follow. This is a very powerful approach that makes the MaryLie fitter remarkably effective.

When the reach is subdivided, there is no need to achieve intermediate targets with high precision. (While the wiener is being dragged, the dog's nose doesn't have to touch the wiener. It only has to remain close.) When operating in the halving mode, the fitter employs a small quantity called ANTOL, and the attempt to achieve an intermediate target is deemed converged when that target is achieved within an error of 10^4 * ANTOL. When AUX = 0, ANTOL is taken to have the default value ANTOL = 10^{-9}. When AUX > 0, ANTOL is taken to have the value ANTOL = AUX.

There is also a perhaps more familiar alternate way to deal with convergence problems. Rather than subdividing the reach in aim space, one hopes that the Newton process is stepping in the right direction, but perhaps with steps that are too large. In that case it might help to compute what step Newton would prescribe, but then actually take a step that is the prescribed step scaled (multiplied) by some *damping* factor. The MaryLie fitter can also employ this option. If MPRINT = −M and there are convergence problems, the fitter does not subdivide the reach, but instead takes steps that are damped compared to what Newton would prescribe. If AUX = 0, the damping factor has a default value of (1/2). If AUX > 0, the value of AUX is used as the damping factor. Damping is employed for the first N steps after the feeler steps.

9.8 Minimize Sum of Squares Optimization

Type Code: mss

Required Parameters:

1. JOB
   = 1 to use nonlinear least squares optimizer.
   Note: When JOB > 0, the inner procedure storage is used. When JOB < 0, the result is the same as when JOB > 0, except that the outer procedure storage is used.
2. VTOL (Error tolerance on value of merit function.)
3. STOL (Error tolerance on step size.)
4. GTOL (Error tolerance on estimated gradients.)
5. DELTA (Feeler step size control.)
6. ISEND
   = 0 to write out nothing.
   = 1 to write brief output at the terminal.
   = 2 to write brief output on file 12.
   = 3 to write brief output both at the terminal and on file 12.
   = 4 to write brief output at the terminal and extensive output on file 12.
   = −N (with N = 1,2,3,4) to achieve the same results as above except the words "brief" and "extensive" are interchanged.

Example:

```
  mssopt     mss
-1  1.E-10  1.E-4  1.E-4  .1  1
```

This specifies a command with the user given name *mssopt*. When invoked it seeks to make the merit function within 10^{-10} of its smallest possible value.

Description:

The *mss* optimizer is designed to work with weighted sum of squares merit functions described by a command with type code *mrt0*. Such merit functions are set up using an *aim* command with JOB = 3. See sections 9.5 and 9.10. For an example of the use of *mss*, see section 10.3.3.

The roles of parameters 2 through 5 are indicated below:

VTOL = Error tolerance on the decrease in the *value* of the merit function. The iteration is considered converged, and the optimization process terminated, when the merit function decrease is < VTOL for NV steps after a full quadratic estimate is formed.

STOL = Error tolerance on step size. The iteration is considered converged, and the optimization process terminated, when all components of the computed step are < STOL.

GTOL = Error tolerance on estimated gradients. The iteration is considered converged, and the optimization process terminated, when all components of the gradient are < GTOL.

DELTA = Feeler step size control. To compute the initial numerical derivatives, on the first NV iterations, any quantity X that is to be varied is set to $X =$ XINITIAL*(1.0 + DELTA), unless XINITIAL = 0.0, in which case $X =$ DELTA.

9.9 General Optimization

Type Code: opt

Required Parameters:

1. JOB
   = 11 to use quadratic step optimizer to minimize first "aim".
   = 12 to use quadratic step optimizer to minimize weighted sum of squares merit function.
   = 13 to use quadratic step optimizer to minimize the square root of weighted sum of squares merit function.
   = 21 to use downhill simplex optimizer to minimize first "aim".
   = 22 to use downhill simplex optimizer to minimize weighted sum of squares merit function.
   = 23 to use downhill simplex optimizer to minimize the square root of weighted sum of squares merit function.
   = 31 to use * optimizer to minimize first "aim".
   = 32 to use * optimizer to minimize weighted sum of squares merit function.
   = 33 to use * optimizer to minimize the square root of weighted sum of squares merit function.

   Note: When JOB > 0, the inner procedure storage is used. When JOB < 0, the result is the same as when JOB > 0, except that the outer procedure storage is used.
2. VTOL (Error tolerance on value of merit function.)
3. STOL (Error tolerance on step size.)
4. GTOL (Error tolerance on estimated gradients.)
5. DELTA (Feeler step size control.)
6. ISEND
   = 0 to write out nothing.
   = 1 to write brief output at the terminal.
   = 2 to write brief output on file 12.
   = 3 to write brief output both at the terminal and on file 12.
   = 4 to write brief output at the terminal and extensive output on file 12.
   = −N (with N = 1,2,3,4) to achieve the same results as above except the words "brief" and "extensive" are interchanged.

Example:

```
  optit     opt
-11  1.E-10  1.E-4  1.E-4  .1  1
```

This specifies a command with the user given name *optit*. When invoked it seeks to make the merit function within 10^{-10} of its smallest possible value using the quadratic step optimizer.

Description:

The *opt* command is intended to allow the user to employ a variety of optimizers for any kind of merit function. For an example of the use of *opt*, see section 10.3.2.

The roles of the parameters 2 through 5 are indicated below:

VTOL = Error tolerance on the decrease in the *value* of the merit function. The iteration is considered converged, and the optimization process terminated, when the merit function decrease is < VTOL for NV steps after a full quadratic estimate is formed.

STOL = Error tolerance on step size. The iteration is considered converged, and the optimization process terminated, when all components of the computed step are < STOL.

GTOL = Error tolerance on estimated gradients. The iteration is considered converged, and the optimization process terminated, when all components of the gradient are < GTOL.

DELTA = Feeler step size control. To compute the initial numerical derivatives, on the first NV iterations, any quantity X that is to be varied is set to $X =$ XINITIAL*(1.0 + DELTA), unless XINITIAL = 0.0, in which case $X =$ DELTA.

When JOB=11, 21, 31, etc., the optimizer attempts to minimize the first "aim" associated with the current procedure and specified by a command with type code *aim*. To achieve that some quantity is the first "aim", it should be entered first, and on a separate line if multiple quantities are to be entered. Other aims may be selected on subsequent input lines. They will be ignored by OPT, but may, for example, be written out by WSQ on each iteration.

When JOB=12, 22, 32, etc., the optimizer attempts to minimize the weighted sum of squares merit function associated with the current procedure and specified by a command with type code *aim*.

When JOB=13, 23, 33, etc., the optimizer attempts to minimize the square root of weighted sum of squares merit function associated with the current procedure and specified by a command with type code *aim*.

9.10 Merit Function (Sum of Squares)

Type Code: mrt0

Required Parameters:

1. KIND
   = 1 for weighted sum of squares merit function.
   = 2 for square root of weighted sum of squares merit function.
2. IPROC
   = 1 to use merit function associated with the inner procedure and specified by a command with type code *aim*.
   = 2 to use merit function associated with the outer procedure and specified by a command with type code *aim*.

Example:

```
merit0    mrt0
   1  2
```

This specifies a command with the user given name *merit0*. When invoked it computes the value of a weighted sum of squares merit function based on information provided by an *aim* command associated with an outer procedure.

Description:

The *mss* optimizer (see section 9.8) is designed to work with a merit function based on a weighted sum of squares. Such a merit function may be set up using an *aim* command with JOB = 3 (see section 9.5). For an example of where this is done, see section 10.3.3. One may also use *mrt0* in connection with *opt*. See section 9.9. Finally, one may set up and use *mrt0* simply to monitor what is going on during the course of some procedure.

9.11 Merit Functions (User Written)

Type Code: mrt1, mrt2, mrt3, mrt4, mrt5

Required Parameters:

1. P1
2. P2
3. P3
4. P4
5. P5
6. P6

Example:

```
merit2    mrt2
0  0  0  0  0  0
```

This specifies a command with the user given name *merit2*. When invoked it computes the value of the merit function specified in the user-written subroutine *mrt2(p)*.

Description:

Merit functions are written by the user to work in conjunction with fitting and/or optimizing procedures, or for other purposes of the user's devising. See, for example, Exhibit 10.3.2a.

9.12 Constraints (User Written)

Type Code: con1, con2, con3, con4, con5

Required Parameters:

1. P1
2. P2
3. P3
4. P4
5. P5
6. P6

Example:

```
couple    con1
0  0  0  0  0  0
```

This specifies a command with the user given name *couple*. When invoked it sets up some relation between the parameters in parameter sets 1 through 9.

Description:

Constraint routines are written by the user. Simple constraints can be specified in the *vary* command. See section 9.6. Completely general relations among parameters can be achieved by user-written constraint routines. These routines have access to all the parameters in parameter sets 1 through 9 (see section 7.25) and can be written to set up any desired relation among them. The values in these parameter sets can then be used to specify the parameter values in any element by using the *random* element option. See section 6.18. Thus, one can set-up any desired relation among all the parameters of all beam-line elements.

The Exhibit below shows a simple constraint routine. In point of fact, it sets up a condition whose net effect could have also been achieved using a suitable *vary* command.

```
Exhibit 9.12, a simple user written constraint subroutine.

c
**********************************************************************
c
       subroutine con2(p)
c impose constraints among parameters in the various parameter sets
       include 'impli.inc'
       include 'param.inc'
       include 'parset.inc'
c
       dimension p(6)
c
```

```
c this constraint relates parameter set p2 to parameter set p1 by
c putting parameter p4 of parameter set p2 equal to minus
c parameter p3 of parameter set p1.
c
c set up control indices
      ipset1=nint(p(1))
      ipset2=nint(p(2))
      ipar1=nint(p(3))
      ipar2=nint(p(4))
c
c relate parameter values
      pst(ipar2,ipset2)=-pst(ipar1,ipset1)
c
      return
      end
c
**********************************************************************
c
```

9.13 Compute Gradient Matrix

Type Code: grad

Required Parameters:

1. JOB

   = 1 to find partial derivatives of all quantities specified in *aim* with respect to all variables specified in *vary*.

   = 2 to find the partial derivatives of the weighted squares merit function associated with all the "aims" related to the current process.

   = 3 to do both.

   Note: When JOB > 0, the inner procedure storage is used. When JOB < 0, the result is the same as when JOB > 0, except that the outer procedure storage is used.

2. ISIDES

   = 1 for single sided gradient.

   = 2 for double sided gradient.

3. IFILE

   = 0 to not write out gradient matrix.

   = any suitable integer to write gradient matrix on file | IFILE |.

   Note: When IFILE is a negative integer, the effect is the same as when IFILE has the corresponding positive value, except that the entries in the matrix are labeled.

4. DELTA

5. SCALE

6. ISEND

   = 0 to write out nothing.

   = 1 to write gradient matrix at the terminal.

   = 2 to write gradient matrix on file 12.

   = 3 to write gradient matrix at the terminal and on file 12.

Example:

```
partials    grad
1 1 0 1.e-4 0 1
```

This specifies a command with the user given name *partials*. When invoked inside a procedure, it computes partial derivatives numerically using *feeler* steps of size DELTA.

Description:

A command with type code *grad* can be used to find numerically the partial derivative of any MaryLie computed quantity with respect to any parameter. This is done within a *procedure*. The quantities to be differentiated are specified with an *aim* command, and the independent differentiation variables are specified with a *vary* command. The Exhibit below illustrates the use of a *grad* command for the simple case of quantities that result from a function of three variables. This function, which happens to be linear, is computed in the user routine *usr14*. (Of course, one could equally well specify any other set of MaryLie computed quantities.) This routine is displayed as well as the MaryLie run that invokes it and computes its gradient matrix of partial derivatives. Observe that the computed derivatives are indeed what is expected. The entries appearing next to the derivatives are the quantities

$$(\text{scale}) * (\text{derivative}/\text{delta})^{1/2}.$$

```
       subroutine user14(p,fa,fm)
c
       include 'impli.inc'
       include 'param.inc'
       include 'usrdat.inc'
c
c Calling arrays
       dimension p(6),fa(0:monoms),fm(6,6)
c
       ucalc(1)=p(3)*p(1) + p(4)*p(2)
       ucalc(2)=p(5)*p(1) + p(6)*p(2)
c
       return
       end

************************************

#comment
  Exhibit 9.13.
  This is a simple MaryLie run to illustrate the use of
  a command with type code grad.  The routine usr14 computes
  a simple linear function (lf) and puts the results in the ucalc
  array according to the rule:

    u(1) = p(3)*p(1) + p(4)*p(2),
    u(2) = p(5)*p(1) + p(6)*p(2).

The aim commands selects u(1) and u(2) as the quantities to
be differentiated.  The vary command selects p(1), p(2), and p(3)
as the variables.  Thus the gradient matrix should contain
the entries below:

1 1    p(3)
2 1    p(5)
1 2    p(4)
2 2    p(6)
```

```
1 3    p(1)
2 3    0

#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  bip      bip
    10.0000000000000
  tip      tip
   0.000000000000000E+00
  aim      aim
    1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000       1.00000000000000
  vary     vary
    3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000       1.00000000000000
  grad     grad
    1.00000000000000       1.00000000000000      0.000000000000000E+00
   1.000000000000000E-03   10.0000000000000       1.00000000000000
  lf       usr14
    1.00000000000000       2.00000000000000       3.00000000000000
    4.00000000000000       5.00000000000000       6.00000000000000
  wuca     wuca
    1.00000000000000       2.00000000000000       1.00000000000000
   0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*lf
     1*wuca
     1*aim
     1*vary
     1*bip
     1*lf
     1*grad
     1*tip
     1*end
  k and ucalc(k) for nonzero values of array
           1   11.0000000000000
           2   17.0000000000000

  Select quantities by entering their symbols
  (Enter a # sign when finished, or type help to review the defined
symbols)
u(1) u(2) #
accept  1:      u(  1)
accept  2:      u(  2)
```

```
  Aims/quantities selected :
No.     item      present value
----------------------------------
  1    u(  1) =       11.0000000
  2    u(  2) =       17.0000000

  Do you wish to start over? (y/n) <n>:

MARYLIE #menu entries available to be varied:
-------------------------------------------------------------------------
aim      end      grad     tip      wuca
bip      fileout  lf       vary

Enter name and (optional) parameter index for #menu element(s) to be
varied.
Elements named following a $ may be reset only.  Type * to relist
Select up to 100 element(s): (Enter a # sign when finished)
lf 1 lf 2 lf 3 #
No.  1 is lf       usr14   .  Parameter 1 out of 6 selected.
        lf(1) =   1.0000000000000
No.  2 is lf       usr14   .  Parameter 2 out of 6 selected.
        lf(2) =   2.0000000000000
No.  3 is lf       usr14   .  Parameter 3 out of 6 selected.
        lf(3) =   3.0000000000000

   Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
  1    lf         usr14        1       1.0000000000000
  2    lf         usr14        2       2.0000000000000
  3    lf         usr14        3       3.0000000000000

  Do you wish to start over? (y/n) <n>:

    1    1   3.000000       547.7226        u(  1) /        lf(1)
    2    1   5.000000       707.1068        u(  2) /        lf(1)
    1    2   4.000000       632.4555        u(  1) /        lf(2)
    2    2   6.000000       774.5967        u(  2) /        lf(2)
    1    3   1.000000       316.2278        u(  1) /        lf(3)
  gradient loop is finished

end of MARYLIE run
```

9.14 Scan Parameter Space

Type Code: scan

Required Parameters:

1. JOB
   = 1 to restore scanned variables to original values on completion of scan.
   = 2 to leave scanned variables in their final state on completion of scan.
   Note: When JOB > 0, the inner procedure storage is used. When JOB < 0, the result is the same as when JOB > 0, except that the outer procedure storage is used.
2. NXSTEP
   = +N to step from X to X + N∗DX.
   = −N to step from X − | N| ∗DX to X + | N| ∗DX.
3. NYSTEP
   = +M to step from Y to Y + M∗DY.
   = −M to step from Y − | M| ∗DY to Y + | M| ∗DY.
4. DX
5. DY
6. ISEND
   = 0 to write out nothing.
   = 1 to write out at the terminal.
   = 2 to write out on file 12.
   = 3 to write out at the terminal and on file 12.

Example:

```
scanit    scan
1  -15  0  10  0  1
```

This specifies a command with the user given name *scanit*. When invoked it steps the quantity "X" from X − 150 to X + 150 in increments of 10.

Description:

It is sometimes useful to study how some or several quantities change as one or two parameters are varied. This can be done using a *scan* command within a *procedure*. See, for example, section 10.4.2. The quantities to be varied are set up using a *vary* command. If one quality is selected, it is interpreted as being "X". If two are selected, the first is taken to be "X" and the second to be "Y". The quantities to be studied must be computed within the procedure and are specified using a *sq* command. See section 8.26.

9.15 Capture Parameter Set

Type Code: cps1, cps2, . . . , cps8, cps9

Required Parameters:

1. P1
2. P2
3. P3
4. P4
5. P5
6. P6

Example and Description:

```
capps1     cps1
  0  0  0  0  0  0
```

This specifies a command with the user given name *capps1*. Suppose parameter set 1 has not previously been *captured* by invoking a command with type code *cps1*. Then, upon invoking *capps1*, its parameter values (in this case 0 0 0 0 0 0) are replaced by those in parameter set 1. At the same time MaryLie sets an internal flag to indicate that parameter set 1 has been captured.

Suppose on the contrary, that parameter set 1 has been captured previously. In that case, invoking *capps1* replaces the parameters in parameter set 1 with those in *capps1*. The internal flag is left unchanged.

Application:

Suppose that one wishes to vary the contents of parameter set 1 in connection with a fitting or optimization or scanning operation, and then return the contents to their original values. This can be done by invoking *capps1* before the operation, and then invoking it again after the operation.

Exhibit:

The Exhibit below illustrates the action of a command with type code *cps1*. To explain what is being done, it is easiest to list the *labor* portion of the master input file along with an explanation of what each entry does. Then, by looking at the output of the MaryLie run, one can see that what has been commanded in *labor* has actually occurred.

```
#labor
     1*fileout  (write out master input file)
     1*oldps1 (set the contents of parameter set 1)
     1*capps1 (capture these values from the parameter set, and flag the
```

```
     parameter set as captured)
     1*fileout (again write out master input file to see that the
     parameters in capps1 have changed)
     1*setps1 ( capture parameter set 1 with a different command having
     type code cps1)
     1*wps1 (observe that this command has now reset the contents of
     parameter set 1)
     1*newps1 (again reset the contents using a ps1 command)
     1*wps1 (observe that this command has now reset the contents of
     parameter set 1
     1*capps1 ( capture parameter set 1 with the command  capps1)
     1*wps1 (observe that  the contents of parameter set 1 have been
     restored to the old values)
     1*newps1 (set the contents of parameter set 1 to new values)
     1*fps1 (free parameter set 1; see section 9.16)
     1*wps1 (observe that the command newps1 has had the intended effect)
     1*capps1 (recapture parameter set 1)
     1*fileout (again write out master input file to see that the
     parameters in capps1 have changed)
     1*end

#comment
  Exhibit 9.15.
  This is a fragmentary MaryLie run to illustrate a simple use of the cpsj
  type code.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  capps1   cps1
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  setps1   cps1
    8.00000000000000       9.00000000000000       10.0000000000000
    11.0000000000000       12.0000000000000       13.0000000000000
  fps1     fps
    1.00000000000000
  wps1     wps
    1.00000000000000       1.00000000000000
  oldps1   ps1
    1.00000000000000       2.00000000000000       3.00000000000000
    4.00000000000000       5.00000000000000       6.00000000000000
  newps1   ps1
    14.0000000000000       15.0000000000000       16.0000000000000
    17.0000000000000       18.0000000000000       19.0000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*oldps1
```

```
     1*capps1
     1*fileout
     1*setps1
     1*wps1
     1*newps1
     1*wps1
     1*capps1
     1*wps1
     1*newps1
     1*fps1
     1*wps1
     1*capps1
     1*fileout
     1*end

#comment
  Exhibit 9.15.
  This is a fragmentary MaryLie run to illustrate a simple use of the cpsj
  type code.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  capps1   cps1
    1.00000000000000       2.00000000000000       3.00000000000000
    4.00000000000000       5.00000000000000       6.00000000000000
  setps1   cps1
    8.00000000000000       9.00000000000000       10.0000000000000
    11.0000000000000       12.0000000000000       13.0000000000000
  fps1     fps
    1.00000000000000
  wps1     wps
    1.00000000000000       1.00000000000000
  oldps1   ps1
    1.00000000000000       2.00000000000000       3.00000000000000
    4.00000000000000       5.00000000000000       6.00000000000000
  newps1   ps1
    14.0000000000000       15.0000000000000       16.0000000000000
    17.0000000000000       18.0000000000000       19.0000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*oldps1
     1*capps1
     1*fileout
     1*setps1
     1*wps1
     1*newps1
     1*wps1
```

```
     1*capps1
     1*wps1
     1*newps1
     1*fps1
     1*wps1
     1*capps1
     1*fileout
     1*end
values of parameters in the parameter set          1
    8.00000000000000       9.00000000000000       10.0000000000000
    11.0000000000000       12.0000000000000       13.0000000000000
values of parameters in the parameter set          1
    14.0000000000000       15.0000000000000       16.0000000000000
    17.0000000000000       18.0000000000000       19.0000000000000
values of parameters in the parameter set          1
    1.00000000000000       2.00000000000000       3.00000000000000
    4.00000000000000       5.00000000000000       6.00000000000000
values of parameters in the parameter set          1
    14.0000000000000       15.0000000000000       16.0000000000000
    17.0000000000000       18.0000000000000       19.0000000000000

#comment
  Exhibit 9.15.
  This is a fragmentary MaryLie run to illustrate a simple use of the cpsj
  type code.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  capps1   cps1
    14.0000000000000       15.0000000000000       16.0000000000000
    17.0000000000000       18.0000000000000       19.0000000000000
  setps1   cps1
    8.00000000000000       9.00000000000000       10.0000000000000
    11.0000000000000       12.0000000000000       13.0000000000000
  fps1     fps
    1.00000000000000
  wps1     wps
    1.00000000000000       1.00000000000000
  oldps1   ps1
    1.00000000000000       2.00000000000000       3.00000000000000
    4.00000000000000       5.00000000000000       6.00000000000000
  newps1   ps1
    14.0000000000000       15.0000000000000       16.0000000000000
    17.0000000000000       18.0000000000000       19.0000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*oldps1
```

```
     1*capps1
     1*fileout
     1*setps1
     1*wps1
     1*newps1
     1*wps1
     1*capps1
     1*wps1
     1*newps1
     1*fps1
     1*wps1
     1*capps1
     1*fileout
     1*end

end of MARYLIE run
```

9.16 Free Parameter Set

Type Code: fps

Required Parameters:

1. IPSET (an integer from 1 to 9 specifying a particular parameter set).

Example:

```
fps1     fps
  1
```

This specifies a command with the user given name *fps1*. When invoked it frees parameter set 1.

Description:

When a parameter set is captured using a command with type code *cps1*, *cps2*, ···, *cps9*, an internal flag (associated with the respective parameter set) is set to indicate that it has been captured. See section 9.15. A command with type code *fps* sets this flag back to its uncaptured value. See Exhibit 9.15.

9.17 Change or Write Out Values of Flags and Defaults

Type Code: flag

Required Parameters:

1. JOB
   = 0 to change control flags and/or defaults.
   = 1 to write values at the terminal.
   = 2 to write values on file 12.
   = 3 to write values at the terminal and on file 12.
2. IMBAD
3. IQUIET
4. 0
5. 0
6. 0

Example:

```
seeflags    flag
  1  0  0  0  0  0
```

This is a command with the user given name *seeflags*. When invoked it displays the current values of the flags at the terminal.

Description:

MaryLie has various internal flags used for control and to indicate the outcome of various operations. The following flags or default values may be interest to the user:

Flag	Default Value	Other Values
IMBAD	0	= 1 if there is a *dhqr2* return error flag when it is called by *eig4* (a routine that finds the eigenvalues and vectors of a 4×4 matrix).
		= 2 if *eig4* finds eigenvalues off the unit circle.
		= 3 if there is a *dhqr2* return error flag when it is called by *eig6* (a routine that finds the eigenvalues and vectors of a 6×6 matrix).
		= 4 if *eig6* finds eigenvalues off the unit circle.
IQUIET	0	= 1 to inhibit writing in the *ppa* subroutine. See section 8.36.
		= 1 to inhibit writing in the *aim* subroutine. See section 9.5.

A command with type code *flag* can be used to examine or change these values. Its operation is similar to type codes *inf* and *zer*. See sections 7.35 and 7.36.

9.18 Reset Menu Entries

Type Code: rset

Required Parameters:

1. JOB (Selects task to be done within *rset*. Must be present, but not yet used.)
2. INFILE (File number from which specifications are to be read).
   = 0 or 5 to read interactively from the terminal.
   = IFILE (any positive integer) to read from that file, after rewinding that file.
   = −IFILE (any negative integer) to read from file IFILE at its current position without first rewinding.
3. LOGFILE (File number to which specifications are to be written).
   = 0 to not write out specifications.
   = JFILE (any positive integer to write specifications to that file.
4. LISTFILE (Extra file to which reset information may be written).
   = 0 to not write out additional information.
   = KFILE (any positive integer) to write additional information to that file.
5. ISEND (Ignored, but still required, when INFILE=0 or 5).
   = 0 to write out nothing.
   = 1 to write out responses at the terminal.
   = 2 to write out responses on file 12.
   = 3 to write out responses at the terminal and on file 12.

Example:

```
restart     rset
  0  0  0  15  1
```

This specifies a command with the user given name *restart*. When invoked it accepts instructions from the terminal about which *menu* entries are to be reset and with what values. It also writes responses at the terminal and on file 15.

Description:

Upon occasion during a MaryLie run the user may wish to reset the parameter values of various menu entries. This can be done with a command having type code *rset*. Information as to what to reset and to what values may be read from an external file or provided interactively. When *rset* is invoked interactively, MaryLie presents in alphabetical order the user-given names of all entries in the *menu* component of the master input file, and invites instructions.

The exhibit below shows the MaryLie responses to *rset* when invoked interactively. It also puts out the master input file after accepting instructions to illustrate that the instructions have been followed. Finally, the contents of LISTFILE are also shown.

```
#comment
  Exhibit 9.18.
  This is a fragmentary MaryLie run to illustrate the response to a
  command with type code rset.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
#menu
  restart  rset
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
    15.0000000000000       1.00000000000000
  drift    drft
    2.00000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*restart
     1*fileout
     1*end
MARYLIE #menu entries available to be varied:
----------------------------------------------------------------------
drift    end      fileout  restart

To RESET #menu items, enter name and (optional) parameter index.
Type * to relist, or a # sign when finished.
drift 1 #
Enter new value for parameter 1 of drift    <   2.0000000    >:
2.5

The following  1 #menu item(s) have been reset:
No.   Item       Type     Parameter   New value
-------------------------------------------------------------
  1    drift      drft         1       2.5000000000000

#comment
  Exhibit 9.18.
  This is a fragmentary MaryLie run to illustrate the response to a
command
  with type code rset.
#beam
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
  0.000000000000000E+000
```

```
#menu
  restart  rset
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
    15.0000000000000       1.00000000000000
  drift    drft
    2.50000000000000
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  end      end
#labor
     1*fileout
     1*restart
     1*fileout
     1*end

end of MARYLIE run

contents of LISTFILE (file 15)

The following  1 #menu item(s) have been reset:
No.   Item       Type     Parameter   New value
------------------------------------------------------------
  1    drift      drft         1       2.5000000000000
```

9.19 Spare

Type Code:

Required Parameters:

Not yet available.

Chapter 10

Additional Examples of Use of MaryLie

10.1 Overview

Chapter 2 presented some simple examples of the use of MaryLie. The purpose of this chapter is to present additional and more advanced examples that employ some of the advanced commands, various procedures, and fitting and optimization commands. For convenience, the examples and their section numbers are listed below:

10.2 Tunes, Chromaticities, and Tune Foot Prints 564
10.2.1 Fitting Tunes and Chromaticities for a Storage Ring 564
10.2.2 Computation of Full Tunes 574
10.2.3 Computation of Tune Foot Print 584
10.3 Octupole Correction and Optimization of a Quadrupole Spot Forming System 590
10.3.1 Fitting Three Quadrupole and Three Octupole Strengths 590
10.3.2 Fitting and Optimizing Seven Octupole Strengths 605
10.3.3 Optimizing Using the MSS Optimizer 625
10.4 Sextupole Correction of a Solenoidal Spot Forming System 642
10.4.1 Fitting Solenoid and Sextupole Strengths 642
10.4.2 Scanning the Sextupole Strength 662
10.5 Fitting at Multiple Lattice Locations 667
10.6 Construction of Third-Order Achromats 678
10.6.1 Construction of Ordinary Third-Order Achromat 678
10.6.2 Construction of Complete Third-Order Achromat 688
10.7 Study of Final Focus Test Beam Facility 704
10.8 Calculation of Dynamic Aperture for a Storage Ring 728
10.8.1 Dynamic Aperture of Tevatron with Strong Distortion Sextupoles . . 729
10.8.2 History of Lost Particle 761
10.9 Production of Lattice Function Plots 773
10.10 Production of Ray Plots 790
10.11 Production of Beam Moment Plots 800
10.12 Production of Composite Plots with Geometry 815

10.12.1 Lattice Function Plots with Geometry 815
10.12.2 Ray Plots with Geometry 828
10.12.3 Beam Moment Plots with Geometry 838
10.13 Production of Layout Drawings 850

Again, as was the case with the examples of Chapter 2, each of these examples could be a research program in its own right; and again in general, this research has not been done. Consequently, the parameter values employed may not be optimal, and are not necessarily even of physical interest.

The MaryLie 3.0 input and output for these examples will be given in detail, and may be of interest to the reader for later reference. However, in most cases, only the highlights of the calculations themselves will be described.

10.2 Tunes, Chromaticities, and Tune Foot Prints

10.2.1 Fitting Tunes and Chromaticities for a Storage Ring

This example demonstrates the fitting of tunes and chromaticities for a storage ring. For simplicity, we treat the case of the small ring described previously in section 2.6. Fitting in MaryLie is done within *logical* loops (also called *procedures*) set up in the #labor component of the master input file. Both inner and outer logical loops are possible in MaryLie. In this example only inner loops (procedures) are used.

An inner logical loop (inner procedure) begins with a command having the type code *bip* (begin inner procedure), and ends with a command having the type code *tip* (terminate inner procedure). The logical loop itself should contain commands that produce whatever computations are required followed by a command having the type code *fit*. Prior to entering the logical loop, commands with the type codes *aim* and *vary* are used to specify what goals are to be achieved, and what quantities are to be varied to meet these goals, respectively.

In this example, there are two successive inner logical loops. See the #labor component of the master input file of the following annotated transcript of a MaryLie run. Text in the transcript surrounded by asterisks indicates comment statements, and text in italics indicates user responses entered interactively during the course of the MaryLie run.

Before entering the first logical loop, the command *aim* is used to specify the desired values of .55 and .30 for the horizontal and vertical tunes, respectively. Then the command *vary* is used to specify that the second parameter (the strength parameter) is to be varied in the quadrupoles *hdq* and *hfq*. See the text in italics to this effect. The tunes are then fit in the first logical loop: First the current transfer map is set to the identity map using the command *clear*. Then the map is recalculated using the line *arc*. Next the tunes and chromaticities for this map are computed using *qtasm*. (To minimize excessive output, *qtasm* is a *quiet* version of *tasm*.) Finally, *fit* adjusts the parameters specified by *vary* to meet the goals specified by *aim*.

Similarly, before entering the second logical loop, the command *aim* is used to specify chromaticities, and *vary* is used to specify sextupole parameters. The chromaticities are then fit in the second logical loop.

After fitting is completed, the command *next* is used to write out a new master input file in which the quadrupoles and sextupoles have the parameter values required to achieve the desired goals.

In this example *aim* and *vary* are used interactively with information supplied by the user at the terminal during the course of a run. If many runs are to be made, it may be more convenient to take this information from external files. These files can be written automatically by first doing one interactive run. See sections 9.5 and 9.6.

NOTE WELL: Each procedure (logical loop) must have within it whatever items are required to compute the quantities that are to be fit. (In this example these items are *clear*, *arc*, and *qtasm*.) Simply specifying in *aim* what is to be fit is not enough. There must be items within the logical loop itself that carry out the necessary associated computations. Failure to provide these items is a common mistake for beginning MaryLie users. If the quantities to be fit do not actually change during a pass through a fitting loop (as a result of failing to compute them or for any other reason) the fitter terminates with the error message "ERROR: Faulty Fit Set Up, Aim Space Point NOT Moving!"

```
***MARYLIE 3.0***
Prerelease Development Version 9/28/91
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
Exhibit 10.2.1
 This is an example of fitting the tunes and chromaticities of
 a small static storage ring. The storage ring is that of section
 2.6 with the RF cavities omitted.  In this example there are two
 successive fitting logical loops.  The first is used to fit the tunes,
 and the second is used to fit first-order chromaticities.  Since
 the four arcs in the lattice are identical in this example, the fitting is done
 for a single arc.
#beam
  4.787400000236000
  2807.643747712000
  1.000000000000000
  1.000000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       12.0000000000000       1.00000000000000
```

```
   1.00000000000000
 hdq      quad
  0.300000000000000      -12.0000000000000       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000      0.000000000000000E+00
 vcs      sext
  0.200000000000000      0.000000000000000E+00
 tasm     tasm
   1.00000000000000      0.000000000000000E+00   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 qtasm    tasm
   1.00000000000000      0.000000000000000E+00   1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 next     pmif
   1.00000000000000       14.0000000000000       3.00000000000000
 clear    iden
 bip      bip
   10.0000000000000
 tip      tip
  0.000000000000000E+00
 aim      aim
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  1.000000000000000E+00   1.00000000000000
 vary     vary
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  1.000000000000000E+00   1.00000000000000
 fit      fit
   10.0000000000000       1.00000000000000      1.000000000000000E-10
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 statring
     4*arc
#lumps
#loops
#labor
    1*fileout
    1*arc
    1*tasm
    1*aim
    1*vary
    1*bip
    1*clear
```

```
    1*arc
    1*qtasm
    1*fit
    1*tip
    1*tasm
    1*aim
    1*vary
    1*bip
    1*clear
    1*arc
    1*qtasm
    1*fit
    1*tip
    1*tasm
    1*next
    1*fin

twiss analysis of static map
det in fxpt is     0.1143E+02

tunes and chromaticities for delta defined in terms of P sub tau:

horizontal tune =  0.5454673831290581
first order horizontal chromaticity =  0.6139095067255258
second order horizontal chromaticity =   15.26625653955741
horizontal tune when delta =  0.0000000000000000E+00
 0.5454673831290581

vertical tune =  0.3258033686659295
first order vertical chromaticity =  0.7723024395232480
second order vertical chromaticity =   14.02884623374476
vertical tune when delta =  0.0000000000000000E+00
 0.3258033686659295

tune separation when delta=  0.0000000000000000E+00
 0.2196640144631286

normalized anharmonicities
 hhn=   5.274519546308877
 vvn=   3.130453324297847
 hvn=   5.453118605262739
det in fxpt is     0.1143E+02

*******************************************************************
*                          Fitting of tunes:                      *
*******************************************************************

 Enter aim(s) in the form:  symbol = target value
 (Enter a # sign when finished, or type help to review the defined symbols)
```

tx=.55 ty=.30 #

```
accept  1:          tx =   0.55000000000000
accept  2:          ty =   0.30000000000000
```

```
 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1        tx =      0.545467383         0.550000000
 2        ty =      0.325803369         0.300000000

 Do you wish to start over? (y/n) <n>:

MARYLIE #menu entries available to be varied:
-------------------------------------------------------------------------
aim      clear    dr3      fin      hdq      next     tip
bend     dr1      dr4      fit      hfq      qtasm    vary
bip      dr2      fileout  hcs      mapout   tasm     vcs

Enter name and (optional) parameter index for #menu element(s) to be varied.
Elements named following a $ may be reset only.  Type * to relist
Selection of   2 more elements is required
```

hdq 2 hfq 2

```
No.  1 is hdq      quad    .  Parameter 2 out of 4 selected.
      hdq(2) =  -12.000000000000
No.  2 is hfq      quad    .  Parameter 2 out of 4 selected.
      hfq(2) =   12.000000000000

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    hdq        quad         2      -12.000000000000
 2    hfq        quad         2       12.000000000000

 Do you wish to start over? (y/n) <n>:

det in fxpt is     0.1143E+02
det in fxpt is     0.1143E+02
Error on iteration    0 is  2.58034E-02        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1146E+02
det in fxpt is     0.1146E+02
Error on iteration    1 is  2.64593E-02        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1141E+02
det in fxpt is     0.1141E+02
Error on iteration    2 is  2.55305E-02        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1020E+02
det in fxpt is     0.1020E+02
Error on iteration    3 is  3.46288E-04        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    4 is  1.77058E-05        reach cut by   1
-------------------------------------------------------
```

```
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    5 is  4.46101E-07        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    6 is  3.46236E-10        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    7 is  6.11733E-14        reach cut by   1
-------------------------------------------------------
After    8 total iterations, final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-------------------------------------------------------
 1        tx =      0.550000000         0.550000000
 2        ty =      0.300000000         0.300000000

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   hdq        quad       2      -11.478299798312
 2   hfq        quad       2       11.865782461835

 Maximum error is     6.117329E-14
 Maximum allowed was  1.000000E-10

twiss analysis of static map
det in fxpt is     0.1022E+02

tunes and chromaticities for delta defined in terms of P sub tau:

horizontal tune =  0.5499999999999388
first order horizontal chromaticity =  0.5737079441800512
second order horizontal chromaticity =   14.18463858679287
horizontal tune when delta =  0.0000000000000000E+00
 0.5499999999999388

vertical tune =  0.3000000000000010
first order vertical chromaticity =  0.7682664023936413
second order vertical chromaticity =   13.40743836885656
vertical tune when delta =  0.0000000000000000E+00
 0.3000000000000010

tune separation when delta=  0.0000000000000000E+00
 0.2499999999999378

normalized anharmonicities
 hhn=   4.322238132571990
 vvn=   3.161697724154100
 hvn=   4.720728217616701
```

```
det in fxpt is     0.1022E+02

*******************************************************************
*                     Fitting of chromaticities:                  *
*******************************************************************

 Enter aim(s) in the form:  symbol = target value
 (Enter a # sign when finished, or type help to review the defined symbols)
```

cx=0 cy=0 #

```
accept  1:          cx =   0.00000000000000E+00
accept  2:          cy =   0.00000000000000E+00

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1        cx =      0.573707944         0.000000000E+00
 2        cy =      0.768266402         0.000000000E+00

 Do you wish to start over? (y/n) <n>:

MARYLIE #menu entries available to be varied:
-----------------------------------------------------------------------
aim      clear    dr3      fin      hdq      next     tip
bend     dr1      dr4      fit      hfq      qtasm    vary
bip      dr2      fileout  hcs      mapout   tasm     vcs

Enter name and (optional) parameter index for #menu element(s) to be varied.
Elements named following a $ may be reset only.  Type * to relist
Selection of   2 more elements is required
```

hcs 2 vcs 2

```
No.  1 is hcs      sext    .  Parameter 2 out of 2 selected.
      hcs(2) =  0.00000000000000E+00
No.  2 is vcs      sext    .  Parameter 2 out of 2 selected.
      vcs(2) =  0.00000000000000E+00

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    hcs        sext         2      0.00000000000000E+00
 2    vcs        sext         2      0.00000000000000E+00

 Do you wish to start over? (y/n) <n>:

det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    0 is  0.76827            reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    1 is  0.76827            reach cut by   1
```

```
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    2 is  0.76827            reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    3 is  5.48311E-09        reach cut by   1
-------------------------------------------------------
det in fxpt is     0.1022E+02
det in fxpt is     0.1022E+02
Error on iteration    4 is  1.96576E-16        reach cut by   1
-------------------------------------------------------
After    5 total iterations, final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1        cx =     -1.766974823E-17     0.000000000E+00
 2        cy =     -1.965759491E-16     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   hcs        sext       2       9.3326886892598
 2   vcs        sext       2      -32.913553962081

 Maximum error is     1.965759E-16
 Maximum allowed was  1.000000E-10

twiss analysis of static map
det in fxpt is     0.1022E+02

tunes and chromaticities for delta defined in terms of P sub tau:

horizontal tune =  0.5499999999999388
first order horizontal chromaticity = -1.7669748230352870E-17
second order horizontal chromaticity =   14.30787737885859
horizontal tune when delta =  0.0000000000000000E+00
 0.5499999999999388

vertical tune =  0.3000000000000010
first order vertical chromaticity = -1.9657594906267568E-16
second order vertical chromaticity =   13.06004194144445
vertical tune when delta =  0.0000000000000000E+00
 0.3000000000000010

tune separation when delta=  0.0000000000000000E+00
 0.2499999999999378

normalized anharmonicities
 hhn=   5.416022776957640
 vvn=   5.317610103391216
```

```
 hvn=  -12.32819282671744
det in fxpt is     0.1022E+02

*****************************************************************
*      Writing out of new master input file with adjusted      *
*                        parameter values:                      *
*****************************************************************

#comment
 Exhibit 10.2.
 This is an example of fitting the tunes and chromaticities of
 a small static storage ring. The storage ring is that of Example
 2.6 with the RF cavities omitted.  In this example there are two
 successive fitting loops.  The first is used to fit the tunes,
 and the second is used to fit first-order chromaticities.  Since
 the four arcs are identical is this example, the fitting is done
 for a single arc.
#beam
  4.787400000236000
  2807.643747712000
  1.000000000000000
  1.000000000000000
#menu
 dr1      drft
  0.750000000000000
 dr2      drft
  0.320000000000000
 dr3      drft
  0.350000000000000
 dr4      drft
  0.200000000000000
 bend     nbnd
   90.0000000000000      0.000000000000000E+00  0.500000000000000
   4.00000000000000       1.00000000000000       1.00000000000000
 hfq      quad
  0.300000000000000       11.8657824618354       1.00000000000000
   1.00000000000000
 hdq      quad
  0.300000000000000      -11.4782997983117       1.00000000000000
   1.00000000000000
 hcs      sext
  0.200000000000000       9.33268868925980
 vcs      sext
  0.200000000000000      -32.9135539620812
 tasm     tasm
   1.00000000000000      0.000000000000000E+00   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 qtasm    tasm
   1.00000000000000      0.000000000000000E+00   1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 fin      end
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
```

```
  0.000000000000000E+00   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 next     pmif
   1.00000000000000       14.0000000000000       3.00000000000000
 clear    iden
 bip      bip
   10.0000000000000
 tip      tip
  0.000000000000000E+00
 aim      aim
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  1.000000000000000E+00   1.00000000000000
 vary     vary
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  1.000000000000000E+00   1.00000000000000
 fit      fit
   10.0000000000000       1.00000000000000      1.000000000000000E-10
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
#lines
 arc
     1*dr3         1*hcs         1*dr4         1*hfq         1*dr2      &
     1*hdq         1*dr4         1*vcs         1*dr3         1*bend     &
     1*dr1         1*hdq         1*dr2         1*hfq         1*dr1
 statring
     4*arc
#lumps
#loops
#labor
    1*fileout
    1*arc
    1*tasm
    1*aim
    1*vary
    1*bip
    1*clear
    1*arc
    1*qtasm
    1*fit
    1*tip
    1*tasm
    1*aim
    1*vary
    1*bip
    1*clear
    1*arc
    1*qtasm
    1*fit
    1*tip
    1*tasm
    1*next
    1*fin
```

```
end of MARYLIE run
```

10.2.2 Computation of Full Tunes

As described in section 8.2, from a knowledge of only the one-turn map for a ring it is possible to compute only the fractional part of the tune. To define full tunes (including their integer parts) it is necessary to make a sufficiently fine subdivision of the ring and to have access to the maps for these subdivisions. Consider the ring shown schematically in figure 2.1 below. The one-turn map $\mathcal{M}_1$ starting and ending at location *1* can be written in the form

$$\mathcal{M}_1 = \mathcal{M}_{12}\mathcal{M}_{23}\mathcal{M}_{34}\cdots\mathcal{M}_{n1}. \tag{10.2.1}$$

Similarly, the one-turn maps for locations *2*, *3*, etc. can be written in the form

$$\mathcal{M}_2 = \mathcal{M}_{23}\mathcal{M}_{34}\cdots\mathcal{M}_{n1}\mathcal{M}_{12}, \tag{10.2.2}$$

$$\mathcal{M}_3 = \mathcal{M}_{34}\mathcal{M}_{45}\cdots\mathcal{M}_{n1}\mathcal{M}_{12}\mathcal{M}_{23}, \text{ etc.} \tag{10.2.3}$$

Let $\mathcal{A}_1$ be the transforming map that brings $\mathcal{M}_1$ to normal form,

$$\mathcal{N} = \mathcal{A}_1\mathcal{M}_1\mathcal{A}_1^{-1}. \tag{10.2.4}$$

Similarly, let $\mathcal{A}_2$, $\mathcal{A}_3$, etc. be the transforming maps that bring $\mathcal{M}_2$, $\mathcal{M}_3$, etc. to normal form,

$$\mathcal{N} = \mathcal{A}_2\mathcal{M}_2\mathcal{A}_2^{-1}, \tag{10.2.5}$$

$$\mathcal{N} = \mathcal{A}_3\mathcal{M}_3\mathcal{A}_3^{-1}, \text{ etc.} \tag{10.2.6}$$

Note that in all cases the normal form $\mathcal{N}$ is the *same* because the maps $\mathcal{M}_1$, $\mathcal{M}_2$, etc. are all conjugate. For example, there is the relation

$$\mathcal{M}_2 = (\mathcal{M}_{12})^{-1}\mathcal{M}_1\mathcal{M}_{12}. \tag{10.2.7}$$

Now use (2.1) and (2.4) to write $\mathcal{N}$ in the form

$$\begin{aligned}\mathcal{N} &= \mathcal{A}_1\mathcal{M}_1\mathcal{A}_1^{-1} = \mathcal{A}_1\mathcal{M}_{12}\mathcal{M}_{23}\cdots\mathcal{M}_{n1}\mathcal{A}_1^{-1}\\ &= \mathcal{A}_1\mathcal{M}_{12}\mathcal{A}_2^{-1}\mathcal{A}_2\mathcal{M}_{23}\mathcal{A}_3^{-1}\cdots\mathcal{A}_n\mathcal{M}_{n1}\mathcal{A}_1^{-1}\\ &= \mathcal{N}_{12}\mathcal{N}_{23}\cdots\mathcal{N}_{n1}.\end{aligned} \tag{10.2.8}$$

Here the maps $\mathcal{N}_{12}$, $\mathcal{N}_{23}$, etc. are defined by the relations

$$\mathcal{N}_{12} = \mathcal{A}_1\mathcal{M}_{12}\mathcal{A}_2^{-1}, \tag{10.2.9}$$

$$\mathcal{N}_{23} = \mathcal{A}_2\mathcal{M}_{23}\mathcal{A}_3^{-1}, \text{ etc.} \tag{10.2.10}$$

It can be shown all the maps $\mathcal{N}_{12}$, $\mathcal{N}_{23}$, etc. commute with each other and with $\mathcal{N}$, and are all in *normal* form.

If the ring is sufficiently subdivided, the subdivision maps $\mathcal{M}_{12}$, $\mathcal{M}_{23}$, etc. will be near the identity. Correspondingly the members of the map pairs $\mathcal{A}_i$, $\mathcal{A}_{i+1}$ will be close to each

Figure 10.2.1: Schematic drawing of a ring showing Poincare surfaces of section (locations) *1* through n.

other. Consequently, the maps $\mathcal{N}_{12}$, $\mathcal{N}_{23}$, etc. given by relations of the form (2.9) and (2.10) will all be close to the identity. It follows that they can be written uniquely in the exponential form

$$\mathcal{N}_{12} = \exp(: h_{12} :), \tag{10.2.11}$$

$$\mathcal{N}_{23} = \exp(: h_{23} :), \text{ etc.} \tag{10.2.12}$$

where the Lie operators $: h_{12} :$, $: h_{23} :$, etc. are all small and all commute.

Based on the observations above, $\mathcal{N}$ can be written in the form

$$\mathcal{N} = \exp(: h :) \tag{10.2.13}$$

with h given by the sum

$$h = h_{12} + h_{23} + \cdots h_{n1}. \tag{10.2.14}$$

Full tunes can now be defined in terms of the coefficients of the *quadratic* parts of h.

Exhibit 10.2.2 illustrates this procedure for the Proton Storage Ring (PSR) using, for convenience, the subdivision introduced in section 10.9. (Actually, this subdivision is far finer than necessary. It is usually sufficient to use just the maps for individual elements without subdividing them further. Indeed, even a coarser subdivision may be adequate. What is required is that the principal value of the logarithm of each $\mathcal{N}_{i,i+1}$ be the correct value.) The line *benchmk* computes tunes for the *nsex* sector (period) map and the one-turn *ring* map to be used later as comparison values. The line *setup* initializes various maps to the identity map, computes the one-turn map and stores it as the lump *luring*, and computes the corresponding $\mathcal{A}$ and stores it as "script A before".

The stage is now set for MaryLie to step its way around the ring using the line *%ring*. At each location marked by a %, *work* is invoked to carry out the following tasks:

Compute the Poincare surface of section map $\mathcal{M}_i$ for that location.

Compute and store $\mathcal{A}_i$ (script A after) and $\mathcal{A}_i^{-1}$ for that location.

Compute $\mathcal{N}_{i-1,i} = \mathcal{A}_{i-1}\mathcal{M}_{i-1,i}\mathcal{A}_i^{-1}$, the transformed current map.

Compute the "phase advances" (scaled exponents) for $\mathcal{N}_{i-1,i}$ with the aid of the *lnf* command (see section 8.41), and add these phase advances to the running total.

Update $\mathcal{A}$.

Finally, the line *tunes* accesses and displays the final *total* tunes. Observe that $f(7)$ is 10 times the horizontal tune for the sector *nsex*, and $f(18)$ is 10 times the vertical tune for the sector *nsex*. This is as it should be, because the full PSR is made of 10 identical sectors (apart from sextupoles whose contribution to tunes is the same as the drift spaces they replace). Note also that the fractional horizontal and vertical tunes computed from the one-turn map *ring* agree with the fractional parts of $f(9)$ and $f(18)$, respectively.

```
Exhibit 10.2.2
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.2.2.
       This is a MaryLie run that illustrates the computation of full tunes
  (including integer parts) using the PSR as an example.  To compute full tunes
  of a circulating lattice, it is necessary to subdivide the lattice into
  sufficiently small pieces.  In this case the dipoles and the long and
  medium-long drifts are split into 5 pieces.  Quads and sextupoles are
  split in two.  This splitting is the same as that used in exhibit 10.9.
  For the purposes of this example such a fine subdivision of the lattice
  is in fact unnecessary.  It would have been sufficient to have simply used
  the maps for the individual beam-line elements.

  The following acronyms are used:
  scurmap = store current map
  gcurmap = get current map
  scummap = store cumulative map
  gcummap = get cumulative map
  sab = store script A before
  gab = get script A before
  saa = store script A after
  gaa = get script A after
  saai = store script A after inverse
  gaai = get script A after inverse
  spha = store phase advance
  gpha = get phase advance
```

```
 benchmk = bench mark
 cpssmap = compute Poincare surface of section map
 caa&aai = compute script A afterward and script A afterward inverse
 tcurmap = transform current map
 adpha = add phase advance

#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 cdrml    drft
   1.48646000000000
 drml/5   drft
  0.297292000000000
 cdrl     drft
   2.28646000000000
 drl/5    drft
  0.457292000000000
 cbend    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 inprot   prot
   18.0000000000000       1.00000000000000
 outprot  prot
   18.0000000000000       2.00000000000000
 infrng   frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       1.00000000000000
 outfrng  frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       2.00000000000000
 ingbdy   gbdy
  0.000000000000000E+00   18.0000000000000      0.000000000000000E+00
   1.20000000000000
 outgbdy  gbdy
  0.000000000000000E+00  0.000000000000000E+00   18.0000000000000
   1.20000000000000
 sbend/5  nbnd
   7.20000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000      0.000000000000000E+00  0.000000000000000E+00
 chfq     quad
  0.500000000000000       2.72000000000000       1.00000000000000
   1.00000000000000
 inhfq    quad
  0.250000000000000       2.72000000000000       1.00000000000000
  0.000000000000000E+00
 outhfq   quad
```

```
 0.250000000000000       2.72000000000000      0.000000000000000E+00
  1.00000000000000
chdq     quad
 0.500000000000000      -1.92000000000000       1.00000000000000
  1.00000000000000
inhdq    quad
 0.250000000000000      -1.92000000000000       1.00000000000000
 0.000000000000000E+00
outhdq   quad
 0.250000000000000      -1.92000000000000      0.000000000000000E+00
  1.00000000000000
chcs     sext
 0.500000000000000      0.000000000000000E+00
hcs/2    sext
 0.250000000000000      0.000000000000000E+00
cvcs     sext
 0.500000000000000      0.000000000000000E+00
vcs/2    sext
 0.250000000000000      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
snor     snor
 0.000000000000000E+00   2.00000000000000      0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00
tasm     tasm
  1.00000000000000      1.000000000000000E-03   1.00000000000000
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
iden     iden
inv      inv
scurmap  stm
  1.00000000000000
gcurmap  gtm
  1.00000000000000       1.00000000000000
scummap  stm
  2.00000000000000
gcummap  gtm
  1.00000000000000       2.00000000000000
sab      stm
  3.00000000000000
gab      gtm
  1.00000000000000       3.00000000000000
saa      stm
  4.00000000000000
gaa      gtm
  1.00000000000000       4.00000000000000
saai     stm
  5.00000000000000
gaai     gtm
  1.00000000000000       5.00000000000000
spha     stm
  6.00000000000000
gpha     gtm
  2.00000000000000       6.00000000000000
```

```
 gbuf1    gbuf
   2.00000000000000       1.00000000000000
 padd     padd
  0.000000000000000E+00   6.00000000000000       6.00000000000000
 wcl10    wcl
   3.00000000000000       10.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-07  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fout     ptm
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 lnf      lnf
   1.00000000000000       1.00000000000000       2.00000000000000
 fin      end
#lines
 drml
     5*drml/5
 drl
     5*drl/5
 bend
     1*inprot      1*infrng      1*ingbdy      5*sbend/5     1*outgbdy  &
     1*outfrng     1*outprot
 hfq
     1*inhfq       1*outhfq
 hdq
     1*inhdq       1*outhdq
 hcs
     2*hcs/2
 vcs
     2*vcs/2
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
 %ring
     1*%           1*drl/5       1*%           1*drl/5       1*%        &
     1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
     1*%           1*inhdq       1*%           1*outhdq      1*%        &
```

```
1*drs        1*%          1*inprot     1*%          1*infrng   &
1*%          1*ingbdy     1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*outgbdy    1*%        &
1*outfrng    1*%          1*outprot    1*%          1*drs      &
1*%          1*inhfq      1*%          1*outhfq     1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*inhdq      1*%          1*outhdq     1*%          1*drs      &
1*%          1*inprot     1*%          1*infrng     1*%        &
1*ingbdy     1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*outgbdy    1*%          1*outfrng  &
1*%          1*outprot    1*%          1*drs        1*%        &
1*inhfq      1*%          1*outhfq     1*%          1*drvs     &
1*%          1*hcs/2      1*%          1*hcs/2      1*%        &
1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
1*%          1*drml/5     1*%          1*drml/5     1*%        &
1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
1*%          1*drml/5     1*%          1*drml/5     1*%        &
1*vcs/2      1*%          1*vcs/2      1*%          1*drvs     &
1*%          1*inhdq      1*%          1*outhdq     1*%        &
1*drs        1*%          1*inprot     1*%          1*infrng   &
1*%          1*ingbdy     1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*outgbdy    1*%        &
1*outfrng    1*%          1*outprot    1*%          1*drs      &
1*%          1*inhfq      1*%          1*outhfq     1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*inhdq      1*%          1*outhdq     1*%          1*drs      &
1*%          1*inprot     1*%          1*infrng     1*%        &
1*ingbdy     1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*outgbdy    1*%          1*outfrng  &
1*%          1*outprot    1*%          1*drs        1*%        &
1*inhfq      1*%          1*outhfq     1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*inhdq    &
1*%          1*outhdq     1*%          1*drs        1*%        &
1*inprot     1*%          1*infrng     1*%          1*ingbdy   &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*outgbdy    1*%          1*outfrng    1*%        &
1*outprot    1*%          1*drs        1*%          1*inhfq    &
1*%          1*outhfq     1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
```

```
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*inhdq      1*%        &
1*outhdq     1*%          1*drs        1*%          1*inprot   &
1*%          1*infrng     1*%          1*ingbdy     1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*outgbdy    1*%          1*outfrng    1*%          1*outprot  &
1*%          1*drs        1*%          1*inhfq      1*%        &
1*outhfq     1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*inhdq      1*%          1*outhdq   &
1*%          1*drs        1*%          1*inprot     1*%        &
1*infrng     1*%          1*ingbdy     1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*outgbdy  &
1*%          1*outfrng    1*%          1*outprot    1*%        &
1*drs        1*%          1*inhfq      1*%          1*outhfq   &
1*%          1*drvs       1*%          1*hcs/2      1*%        &
1*hcs/2      1*%          1*drml/5     1*%          1*drml/5   &
1*%          1*drml/5     1*%          1*drml/5     1*%        &
1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
1*%          1*drml/5     1*%          1*drml/5     1*%        &
1*drml/5     1*%          1*vcs/2      1*%          1*vcs/2    &
1*%          1*drvs       1*%          1*inhdq      1*%        &
1*outhdq     1*%          1*drs        1*%          1*inprot   &
1*%          1*infrng     1*%          1*ingbdy     1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*outgbdy    1*%          1*outfrng    1*%          1*outprot  &
1*%          1*drs        1*%          1*inhfq      1*%        &
1*outhfq     1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*inhdq      1*%          1*outhdq   &
1*%          1*drs        1*%          1*inprot     1*%        &
1*infrng     1*%          1*ingbdy     1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*outgbdy  &
1*%          1*outfrng    1*%          1*outprot    1*%        &
1*drs        1*%          1*inhfq      1*%          1*outhfq   &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*inhdq      1*%          1*outhdq     1*%        &
1*drs        1*%          1*inprot     1*%          1*infrng   &
1*%          1*ingbdy     1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*outgbdy    1*%        &
```

```
     1*outfrng     1*%           1*outprot     1*%           1*drs      &
     1*%           1*inhfq       1*%           1*outhfq      1*%        &
     1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
     1*%           1*drl/5       1*%           1*drl/5       1*%
 benchmk
     1*nsex        1*tasm        1*iden        1*ring        1*tasm     &
     1*iden
 setup
     1*setzero     1*iden        1*scurmap     1*scummap     1*spha     &
     1*luring      1*snor        1*gbuf1       1*sab         1*iden
 %
     1*work
 work
     1*scurmap     1*cpssmap     1*caa&aai     1*tcurmap     1*adpha    &
     1*update      1*iden
 cpssmap
     1*iden        1*gcummap     1*gcurmap     1*scummap     1*inv      &
     1*luring      1*gcummap
 caa&aai
     1*snor        1*gbuf1       1*saa         1*inv         1*saai
 tcurmap
     1*iden        1*gab         1*gcurmap     1*gaai
 adpha
     1*lnf         1*gbuf1       1*padd
 update
     1*iden        1*gaa         1*sab
 tunes
     1*gpha        1*fout
#lumps
 luring
     1*ring
#loops
 lring
     1*ring
#labor
    1*fileout
    1*benchmk
    1*setup
    1*%ring
    1*tunes
    1*fin

*********************************************
* Calculation of tunes for the nsex sector: *
*********************************************

 twiss analysis of static map

 tunes and chromaticities for delta defined in terms of P sub tau:

 horizontal tune =  0.225410281174760
 first order horizontal chromaticity =  0.110337520026812
 second order horizontal chromaticity =   3.75197479724374
```

```
 horizontal tune when delta =  1.000000000000000E-003
  0.225524370669584

 vertical tune =  0.225543770585176
 first order vertical chromaticity =  0.250986502239161
 second order vertical chromaticity =   2.18869156252695
 vertical tune when delta =  1.000000000000000E-003
  0.225796945778978

 tune separation when delta=  1.000000000000000E-003
 -2.725751093941298E-004

 normalized anharmonicities
  hhn=  0.481212294862244
  vvn=  0.287510196121498
  hvn=  0.489194351436008

*******************************************
* Calculation of (fractional) tunes using *
* the one-turn map for the whole ring:    *
*******************************************

 twiss analysis of static map

 tunes and chromaticities for delta defined in terms of P sub tau:

 horizontal tune =  0.254102811747597
 first order horizontal chromaticity =   1.10337520026812
 second order horizontal chromaticity =   37.5197479724370
 horizontal tune when delta =  1.000000000000000E-003
  0.255243706695837

 vertical tune =  0.255437705851762
 first order vertical chromaticity =   2.50986502239162
 second order vertical chromaticity =   21.8869156252694
 vertical tune when delta =  1.000000000000000E-003
  0.257969457789779

 tune separation when delta=  1.000000000000000E-003
 -2.725751093941298E-003

 normalized anharmonicities
  hhn=   4.81212294862237
  vvn=   2.87510196121501
  hvn=   4.89194351436007

lump luring   constructed and stored.( 1)

****************
* Total tunes: *
****************

nonzero elements in generating polynomial are :
```

```
 f(  7)=f( 20 00 00 )=  2.2541028117476
 f( 13)=f( 02 00 00 )=  2.2541028117476
 f( 18)=f( 00 20 00 )=  2.2554377058518
 f( 22)=f( 00 02 00 )=  2.2554377058518

end of MARYLIE run
```

10.2.3 Computation of Tune Foot Print

Exhibit 8.2 in section 8.2 illustrated the use of a *tasm* command to compute the tunes for a single particle in a static ring. It is also possible to compute all the tunes for a collection of particles. This can be done by preparing a particle distribution and repeatedly using a tasm command within a logical loop. The particle distribution is stored on an external file, and particle phase-space coordinates are read in particle-by-particle from this file with the aid of a random parameter set command. A *tasm* command then takes these coordinates from the parameter set and computes the tunes for each particle. Finally, the tunes for each particle can be written on an external file (with the aid of *sq* and *wsq* commands) for subsequent plotting to produce a tune footprint for the particle distribution. (See section 7.26 for a description of the use of random parameter sets.) Exhibits 10.2.3a and 10.2.3b below illustrate this process for a simple example. Exhibit 10.2.3a displays the instructions for the *sq* command that along with the *wsq* command are used to write out the array entries tn(1,3) and tn(2,3). (See sections 8.26 and 9.5). Exhibit 10.2.3b shows the MaryLie run itself. Note that in this example a simple set of preliminary commands was used to generate a particle distribution. However, this distribution could have come from anywhere.

Figures 10.2.3.1 through 10.2.3.3 display the particle distribution. Figure 10.2.3.4 shows the footprint in tune space produced by this distribution for the case of the PSR. Finally, note that a similar process could have been carried out for a dynamic ring by using a *tadm* command. In this case, the tune footprint would be three dimensional.

Exhibit 10.2.3 a: Contents of file 15 that provides instructions for the *sq* command.

```
1 tn(1,3)
2 tn(2,3)
#
```

Exhibit 10.2.3b: MaryLie run illustrating the computation of tunes for a distribution of particles.

```
#comment
  Exhibit 10.2.3
  This MaryLie run illustrates the computation of a tune footprint.
  For simplicity, a tune footprint is computed for the PSR ring
  of Section 2.5. This run consists of three parts:

   a) Generation of the 1-turn map.

   b) Generation of a particle distribution.  This is done by the line
```

```
      makeic.  For convenience, a gaussian distribution is used. The
      particle distribution is written on file 14.

   c) Computation of the tune footprint for this distribution.  This
      is done using a logical loop.  Rays are read in one at a time
      from file 14 using the random parameter set command rptdata.
      Instructions for sq, which is invoked before entering the loop,
      are taken from file 15.  Tunes are written on file 16 for
      subsequent plotting.

#beam
   4.86914813175970
  0.849425847892200
   1.00000000000000
   1.00000000000000
#menu
  drvs     drft
   0.300000000000000
  drs      drft
   0.450000000000000
  drml     drft
    1.48646000000000
  drl      drft
    2.28646000000000
  bend     pbnd
    36.0000000000000      0.000000000000000E+00  0.500000000000000
    1.20000000000000
  hfq      quad
   0.500000000000000       2.72000000000000       1.00000000000000
    1.00000000000000
  hdq      quad
   0.500000000000000      -1.92000000000000       1.00000000000000
    1.00000000000000
  hcs      sext
   0.500000000000000      0.000000000000000E+00
  vcs      sext
   0.500000000000000      0.000000000000000E+00
  fileout  pmif
    1.00000000000000       12.0000000000000       3.00000000000000
  mapout   ptm
    3.00000000000000       3.00000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000
  tasmpt   tasm
    1.00000000000000       1.00000000000000       1.00000000000000
   0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  qtasmpt  tasm
    1.00000000000000       1.00000000000000       1.00000000000000
   0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  ptdata   ps1
   1.000000000000000E-02  0.000000000000000E+00  1.000000000000000E-02
   0.000000000000000E+00  0.000000000000000E+00  1.000000000000000E-03
  rptdata  rps1
    14.0000000000000      0.000000000000000E+00
```

```
  clear    iden
  zer      zer
   0.000000000000000E+00  1.000000000000000E-10  1.000000000000000E-10
   0.000000000000000E+00
  fin      end
  sho3     shoa
    3.00000000000000       1.00000000000000      0.000000000000000E+00
  bip      bip
    100.000000000000
  tip      tip
   0.000000000000000E+00
  sq       sq
    15.0000000000000      0.000000000000000E+00   1.00000000000000
    1.00000000000000
  wsq      wsq
    1.00000000000000       1.00000000000000       16.0000000000000
    1.00000000000000       1.00000000000000      0.000000000000000E+00
  bgendat  ps1
   1.000000000000000E-04  1.000000000000000E-04  1.000000000000000E-04
    5.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  bgen     bgen
    6.00000000000000       123.000000000000       100.000000000000
    137.000000000000       1.00000000000000       1.00000000000000
  raysout  rt
   0.000000000000000E+00   14.0000000000000      0.000000000000000E+00
   0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
#lines
  nsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drl
  tsex
      1*drl         1*hdq         1*drs         1*bend        1*drs      &
      1*hfq         1*drvs        1*hcs         1*drml
  lsex
      1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
      1*bend        1*drs         1*hfq         1*drl
  half
      1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
  ring
      2*half
  makeic
      1*bgendat     1*bgen        1*raysout
#lumps
#loops
#labor
     1*fileout
     1*ring
     1*makeic
     1*sq
     1*bip
     1*rptdata
     1*qtasmpt
     1*wsq
```

```
     1*tip
     1*fin

         100 rays generated

  numerically computed values of selected moments
  values of <x*x>, <x*px>, <px*px>:
  9.425467645700800E-005 -1.354881408024196E-005  1.017819260324027E-004
  values of <y*y>, <y*py>, <py*py>:
  1.082160291148366E-004  6.573236882795287E-006  8.852331615780860E-005
  values of <t*t>, <t*pt>, <pt*pt>:
  1.134396798850281E-004  2.404587894362543E-005  9.217736080450674E-005

  analytically computed values of selected moments
  values of <x*x>, <x*px>, <px*px>:
  1.000000000000000E-004  0.000000000000000E+000  1.000000000000000E-004
  values of <y*y>, <y*py>, <py*py>:
  1.000000000000000E-004  0.000000000000000E+000  1.000000000000000E-004
  values of <t*t>, <t*pt>, <pt*pt>:
  1.000000000000000E-004  0.000000000000000E+000  1.000000000000000E-004

  In subroutine sq
accept  1:     tn(1,3)
accept  2:     tn(2,3)

  Aims/quantities selected :
No.     item      present value
-----------------------------------
  1   tn(1,3) =      0.000000000E+00
  2   tn(2,3) =      0.000000000E+00

end of MARYLIE run
```

Figure 10.2.3.1: Horizontal projection of phase-space distribution produced by plotting the first two columns of file 14.

Figure 10.2.3.2: Vertical projection of phase-space distribution produced by plotting the third and fourth columns of file 14.

Figure 10.2.3.3: Temporal projection of phase-space distribution produced by plotting the last two columns of file 14.

Figure 10.2.3.4: Tune footprint associated with the distribution shown in Figures 10.2.3.1 through 10.2.3.3 and produced by plotting the contents of file 16.

10.3 Octupole Correction and Optimization of a Quadrupole Spot Forming System

Section 2.2 illustrated the importance of certain third-order aberrations for governing the performance of a spot forming system. See figure 2.2.2. In the examples of this section these aberrations are corrected with the aid of octupoles. In subsection 10.3.1 the simple quadrupoles in the triplet of figure 2.2.1 are replaced with combined-function quadrupoles that are capable of having an octupole component. See section 6.24. The octupole components of these combined-function quadrupoles are then varied to correct the detrimental third-order aberrations. In subsection 10.3.2 four additional octupoles are added to the system and all seven octupole strengths are then varied to find an optimized fit. Finally, in subsection 10.3.3, the same optimized fitting is carried out using the *mss* optimizer.

10.3.1 Fitting Three Quadrupole and Three Octupole Strengths

The first part of the MaryLie run for this example, shown in Exhibit 10.3.1, is devoted to fitting the quadrupole strengths in order to achieve the paraxial spot-forming condition $r(1,1) = r(3,3) = 0$. In this case there are three quadrupole strengths to be adjusted with the constraint that the outer quadrupoles should have the same strengths. This is accomplished by taking three steps:

1. At the beginning of the MaryLie run the inner and outer quads are given the same nominal strengths (.1 Tesla/meter).

2. The "vary" command *qvary* that is used to vary the quadrupoles has the value JOB=−1 for its first parameter. See section 9.6.

3. The variation of the third quadrupole strength, parm3(1), is required to be the same as that of the first quadrupole strength. See the italicized text entry "*1 1*" in the quadrupole fitting part of the Exhibit.

The second part of the run is devoted to adjusting the three normal octupole components of the three combined function quadrupoles to achieve the goal $f(140) = f(149) = f(195) = 0$ for the Lie generators of the net transfer map for the full system. (It can be shown that these are the Lie generators for the offensive geometric aberrations of a spot-forming system.) In this case there are no constraints and the "vary" command *ovary* is used with JOB=1 to vary the strength of the octupole components.

In the third part of the run rays are traced through the octupole-corrected spot-forming system using the net transfer map for the full system. The results of this ray trace are shown in figure 10.3.1.1, which is to be compared with figure 2.2.3. Evidently, within the achieved numerical accuracy of the paraxial fitting requirement $r(1,1) = r(3,3) = 0$, all the rays of figure 2.2.2 are now brought to a point. This is because the ray trace has been done through third order using a single map, and all the detrimental geometric third-order aberrations for a spot-forming system have been fit to zero.

Of course, there are also detrimental higher-order aberrations beyond third order. An estimate of the effect of these aberrations can be had by tracing rays element-by-element

rather than using a single map. In this way, although the map for each element is still used only through third order, at least the higher-order aberrations associated with concatenating these elements can be estimated.

As illustrated in the fourth part of the run, element-by-element ray tracing can be achieved by invoking the loop *onebyone* whose content is *cspot*, and use of the "circ" command *circ15*. The result of this element-by-element ray trace is shown in figure 10.3.1.2, which should also be compared with figure 2.2.3. Now, thanks to octupole correction, the spot size has shrunk from its initial value of 4×10^{-7} meters (see figure 2.2.3) to an estimated value of 1×10^{-8} meters (see figure 10.3.1.2).

```
***MARYLIE 3.0***
Prerelease Development Version 5/25/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.3.1.
 This is a MARYLIE run that demonstrates the use of three octupoles to
 correct three offensive aberrations in the simple spot forming system
 of Section 2.2.  To do this the simple quadrupoles are replaced with
 combined function quadrupoles having a possible octupole component.
 This run does four things:

  a) It adjusts quad strengths as an example of fitting of matrix
     elements and the use of simple constraints.

  b) It adjusts three octupole strengths to remove ("zero out") three
     Lie generators corresponding to three third-order geometrical
     aberrations that are detrimental to the performance of a
     spot forming system.  These Lie generators are:

     f(140)=f( 04 00 00 ),
     f(149)=f( 02 02 00 ), and
     f(195)=f( 00 04 00 ).

  c) It traces rays through this corrected system using the total
     transfer map.

  d) To give a rough indication of residual fifth-order aberration
     effects, rays are also traced element-by-element through the
     corrected system.  Doing so illustrates as well use of the
     element-by-element tracking ("turtle") feature of MaryLie
     acomplished with the aid of a loop and a 'circ' command.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
```

```
#menu
 drs      drft
  0.500000000000000
 drl      drft
   20.0026000000000
 cfq1     cfqd
   1.50000000000000       1.00000000000000       1.00000000000000
   1.00000000000000
 cfq2     cfqd
   3.00000000000000       2.00000000000000       1.00000000000000
   1.00000000000000
 cfq3     cfqd
   1.50000000000000       3.00000000000000       1.00000000000000
   1.00000000000000
 parm1    ps1
  0.100000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 parm2    ps2
 -0.100000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 parm3    ps3
  0.100000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace14  rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000      0.000000000000000E+00
 circ15   circ
  0.000000000000000E+00   15.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000       3.00000000000000
 clear    iden
 bip      bip
   20.0000000000000
 tip      tip
  0.000000000000000E+00
 qaim     aim
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 oaim     aim
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 qvary    vary
  -1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 ovary    vary
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
```

```
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 fit      fit
   1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
 end      end
#lines
 ctrip
     1*cq1         1*drs         1*cq2         1*drs         1*cq3
 cspot
     1*ctrip       1*drl
 cq1
     1*parm1       1*cfq1
 cq2
     1*parm2       1*cfq2
 cq3
     1*parm3       1*cfq3
#lumps
#loops
 onebyone
     1*cspot
#labor
    1*fileout
    1*cspot
    1*mapout
    1*qaim
    1*qvary
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*oaim
    1*ovary
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*mapout
    1*fileout
    1*raysin
    1*trace14
    1*clear
    1*raysin
    1*onebyone
    1*circ15
    1*end

matrix for map is :

-9.79758E-02  2.45188E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-4.39027E-02  7.80192E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -8.27252E-01  2.04751E+01  0.00000E+00  0.00000E+00
```

```
 0.00000E+00  0.00000E+00 -8.03617E-02  7.80192E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.46784E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-6.04808059081979E-02
 f( 38)=f( 11 00 01 )=  2.0122232539117
 f( 53)=f( 02 00 01 )= -67.414521760868
 f( 67)=f( 00 20 01 )=-0.17640226595271
 f( 70)=f( 00 11 01 )=  7.4697013709437
 f( 76)=f( 00 02 01 )= -117.36243475098
 f( 83)=f( 00 00 03 )= -392.90849771004
 f( 84)=f( 40 00 00 )=-2.11757899035411E-03
 f( 85)=f( 31 00 00 )= 0.20262469843156
 f( 90)=f( 22 00 00 )= -7.4710486909263
 f( 95)=f( 20 20 00 )=-2.19642741685004E-02
 f( 96)=f( 20 11 00 )= 0.98611756284480
 f( 99)=f( 20 02 00 )= -11.141058580412
 f(104)=f( 20 00 02 )=-0.24588274282846
 f(105)=f( 13 00 00 )=  125.73129325848
 f(110)=f( 11 20 00 )=  1.0668779239009
 f(111)=f( 11 11 00 )= -48.200211504437
 f(114)=f( 11 02 00 )=  547.95111488459
 f(119)=f( 11 00 02 )=  10.503188819264
 f(140)=f( 04 00 00 )= -818.08252738262
 f(145)=f( 02 20 00 )= -13.386602379412
 f(146)=f( 02 11 00 )=  608.50033286562
 f(149)=f( 02 02 00 )= -6966.2862966306
 f(154)=f( 02 00 02 )= -352.37791726722
 f(175)=f( 00 40 00 )=-7.53266954192657E-03
 f(176)=f( 00 31 00 )= 0.67996172743022
 f(179)=f( 00 22 00 )= -23.083862921095
 f(184)=f( 00 20 02 )=-0.96482279237247
 f(185)=f( 00 13 00 )=  349.22093373843
 f(190)=f( 00 11 02 )=  53.067411640374
 f(195)=f( 00 04 00 )= -1989.0172460778
 f(200)=f( 00 02 02 )= -862.77140055689
 f(209)=f( 00 00 04 )= -1533.0379170756

***********************************************************************
*                     Fitting of quad strengths:                      *
*  Note that strengths of quads 1 and 2 are varied, and the strength  *
*  of quad 3 is constrained to be the same as that of quad 1.         *
***********************************************************************

 Enter aim(s) in the form:  symbol = target value
 (Enter a # sign when finished, or type help to review the defined symbols)
```

r(1,1)=0 r(3,3)=0 #

```
accept  1:      r(1,1)
accept  2:      r(3,3)
```

```
 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    r(1,1) =     -9.797576941E-02     0.000000000E+00
 2    r(3,3) =     -0.827251756         0.000000000E+00

 Do you wish to start over? (y/n) <n>:

MARYLIE #menu entries available to be varied:
-------------------------------------------------------------------------
bip      cfq3     drl      fileout  oaim     parm2    qvary    trace14
cfq1     circ15   drs      fit      ovary    parm3    raysin
cfq2     clear    end      mapout   parm1    qaim     tip

Enter name and (optional) parameter index for #menu element(s) to be varied.
Elements named following a $ may be reset only.  Type * to relist
Selection of   2 more elements is required
```

parm1 1 parm2 1

```
No.  1 is parm1    ps1     .  Parameter 1 out of 6 selected.
    parm1(1) =  0.10000000000000
No.  2 is parm2    ps2     .  Parameter 1 out of 6 selected.
    parm2(1) = -0.10000000000000

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    parm1      ps1          1      0.10000000000000
 2    parm2      ps2          1     -0.10000000000000

 Do you wish to start over? (y/n) <n>:


Define up to  98 dependent variables V by entering their names
and (optional) parameter indices. (Enter a # sign when finished)
```

parm3 1 #

```
No.  1 is parm3    ps3     .  Parameter 1 out of 6 selected.
Enter ID (1 to   2) of independent variable x , and the derivative dV/dx:
```

1 1

```
  Dependent #menu elements selected:
No.  Element    Type     Parameter   Present value       IDV  Slope
-------------------------------------------------------------------------
 1   parm3      ps3        1      0.10000000000000        1   1.00000

 Do you wish to start over? (y/n) <n>:

 target           1 =  0.000000000000000E+000
```

```
 target           2 =  0.000000000000000E+000
Iter   1 Error= 8.2725E-01,  Step= 1.0000E-04,  SubErr= 8.2725E-01 @cut=    1
-------------------------------------------------------
Iter   2 Error= 8.2124E-01,  Step=-1.0000E-04,  SubErr= 8.2124E-01 @cut=    1
-------------------------------------------------------
Iter   3 Error= 8.3724E-01,  Step= 2.0197E-02,  SubErr= 8.3724E-01 @cut=    1
-------------------------------------------------------
Iter   4 Error= 4.8878E-02,  Step= 1.6411E-03,  SubErr= 4.8878E-02 @cut=    1
-------------------------------------------------------
Iter   5 Error= 4.7241E-03,  Step= 1.5524E-04,  SubErr= 4.7241E-03 @cut=    1
-------------------------------------------------------
Iter   6 Error= 1.6004E-04,  Step= 5.9429E-06,  SubErr= 1.6004E-04 @cut=    1
-------------------------------------------------------
Iter   7 Error= 1.3818E-06,  Step= 4.7931E-08,  SubErr= 1.3818E-06 @cut=    1
-------------------------------------------------------
Iter   8 Error= 7.6755E-10,  Step= 2.7713E-11,  SubErr= 7.6755E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   9 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    r(1,1) =     -2.553512957E-15     0.000000000E+00
 2    r(3,3) =     -1.143529715E-14     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   parm1      ps1        1      8.63000353805416E-02
 2   parm2      ps2        1     -8.28945310775849E-02
 3   parm3      ps3        1      8.63000353805416E-02    1    1.0000

 Maximum error is     1.143530E-14
 Maximum allowed was  1.000000E-10

*************************************************************************
*            Fitting of octupole strengths:                             *
*************************************************************************

 Enter aim(s) in the form:  symbol = target value
 (Enter a # sign when finished, or type help to review the defined symbols)
```

f(140)=0 f(149)=0 f(195)=0 #

```
accept  1:      f(140)
accept  2:      f(149)
accept  3:      f(195)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =      -636.933699         0.000000000E+00
 2    f(149) =      -5304.72550         0.000000000E+00
```

```
 3    f(195) =      -1394.45593         0.000000000E+00

 Do you wish to start over? (y/n) <n>:

MARYLIE #menu entries available to be varied:
-----------------------------------------------------------------------
bip      cfq3     drl      fileout  oaim     parm2    qvary    trace14
cfq1     circ15   drs      fit      ovary    parm3    raysin
cfq2     clear    end      mapout   parm1    qaim     tip

Enter name and (optional) parameter index for #menu element(s) to be varied.
Elements named following a $ may be reset only.  Type * to relist
Selection of   3 more elements is required
```

parm1 5 parm2 5 parm3 5

```
No.  1 is parm1    ps1     .  Parameter 5 out of 6 selected.
    parm1(5) =  0.00000000000000E+00
No.  2 is parm2    ps2     .  Parameter 5 out of 6 selected.
    parm2(5) =  0.00000000000000E+00
No.  3 is parm3    ps3     .  Parameter 5 out of 6 selected.
    parm3(5) =  0.00000000000000E+00

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    parm1      ps1          5      0.00000000000000E+00
 2    parm2      ps2          5      0.00000000000000E+00
 3    parm3      ps3          5      0.00000000000000E+00

 Do you wish to start over? (y/n) <n>:

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 5.3047E+03,  Step= 1.0000E-05,  SubErr= 5.3047E+03 @cut=    1
-------------------------------------------------------
Iter   2 Error= 5.2978E+03,  Step= 1.0000E-05,  SubErr= 5.2978E+03 @cut=    1
-------------------------------------------------------
Iter   3 Error= 5.2931E+03,  Step= 1.0000E-05,  SubErr= 5.2931E+03 @cut=    1
-------------------------------------------------------
Iter   4 Error= 5.3007E+03,  Step= 5.7594E-01,  SubErr= 5.3007E+03 @cut=    1
-------------------------------------------------------
Iter   5 Error= 2.1723E-08,  Step= 6.0969E-13,  SubErr= 2.1723E-08 @cut=    1
-------------------------------------------------------
Quit on iteration   6 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-------------------------------------------------
 1    f(140) =      1.168842800E-12     0.000000000E+00
 2    f(149) =      6.771472272E-12     0.000000000E+00
 3    f(195) =     -4.437339385E-12     0.000000000E+00
```

```
 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   parm1      ps1        5     -0.23700261617188
 2   parm2      ps2        5     -3.62329120367890E-02
 3   parm3      ps3        5      0.52366882583228

 Maximum error is     6.771472E-12
 Maximum allowed was  1.000000E-10

*************************************************************************
*  Transfer map for the corrected system:  Note that r(1,1), r(3,3),      *
*  f(140), f(149), and f(195) are now vanishingly small as desired.       *
*************************************************************************

matrix for map is :

-2.55351E-15  2.47288E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-4.04388E-02  8.08880E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.14353E-14  2.28173E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -4.38264E-02  8.76642E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.46784E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-4.57660844478320E-02
 f( 38)=f( 11 00 01 )=  1.4345989929097
 f( 53)=f( 02 00 01 )= -59.832856603695
 f( 67)=f( 00 20 01 )=-0.10955102975777
 f( 70)=f( 00 11 01 )=  4.6810988356401
 f( 76)=f( 00 02 01 )= -88.881855045722
 f( 83)=f( 00 00 03 )= -392.90849771004
 f( 84)=f( 40 00 00 )=-0.12327408289743
 f( 85)=f( 31 00 00 )=  8.7910508067722
 f( 90)=f( 22 00 00 )= -211.07885008370
 f( 95)=f( 20 20 00 )= 0.68595569362657
 f( 96)=f( 20 11 00 )= -25.219718510725
 f( 99)=f( 20 02 00 )=  221.73626436929
 f(104)=f( 20 00 02 )=-0.19073155606124
 f(105)=f( 13 00 00 )=  1706.8172870476
 f(110)=f( 11 20 00 )= -22.188718833106
 f(111)=f( 11 11 00 )=  736.23108430665
 f(114)=f( 11 02 00 )= -5357.5423986594
 f(119)=f( 11 00 02 )=  7.8836884544112
 f(140)=f( 04 00 00 )= 1.16884280032536E-12
 f(145)=f( 02 20 00 )=  139.95819449778
 f(146)=f( 02 11 00 )= -3163.2230792371
 f(149)=f( 02 02 00 )= 6.77147227179375E-12
 f(154)=f( 02 00 02 )= -309.42966067238
 f(175)=f( 00 40 00 )=-9.75407684979115E-02
 f(176)=f( 00 31 00 )=  6.5454816213134
```

```
 f(179)=f( 00 22 00 )= -146.85444082835
 f(184)=f( 00 20 02 )=-0.58685439648812
 f(185)=f( 00 13 00 )=  1101.5239192045
 f(190)=f( 00 11 02 )=  32.524413735691
 f(195)=f( 00 04 00 )=-4.43733938482183E-12
 f(200)=f( 00 02 02 )= -601.71802504631
 f(209)=f( 00 00 04 )= -1533.0379170756

************************************************************************
*     Writing out new master input file with adjusted                  *
*                        parameter values:                             *
************************************************************************

#comment
 Exhibit 10.3.1.
 This is a MARYLIE run that demonstrates the use of three octupoles to
 correct three offensive aberrations in the simple spot forming system
 of Section 2.2.  To do this the simple quadrupoles are replaced with
 combined function quadrupoles having a possible octupole component.
 This run does four things:

  a) It adjusts quad strengths as an example of fitting of matrix
     elements and the use of simple constraints.

  b) It adjusts three octupole strengths to remove ("zero out") three
     Lie generators corresponding to three third-order geometrical
     aberrations that are detrimental to the performance of a
     spot forming system.  These Lie generators are:

     f(140)=f( 04 00 00 ),
     f(149)=f( 02 02 00 ), and
     f(195)=f( 00 04 00 ).

  c) It traces rays through this corrected system using the total
     transfer map.

  d) To give a rough indication of residual fifth-order aberration
     effects, rays are also traced element-by-element through the
     corrected system.  Doing so illustrates as well use of the
     element-by-element tracking ("turtle") feature of MaryLie
     acomplished with the aid of a loop and a 'circ' command.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 drs      drft
  0.500000000000000
 drl      drft
```

```
  20.0026000000000
cfq1     cfqd
  1.50000000000000       1.00000000000000       1.00000000000000
  1.00000000000000
cfq2     cfqd
  3.00000000000000       2.00000000000000       1.00000000000000
  1.00000000000000
cfq3     cfqd
  1.50000000000000       3.00000000000000       1.00000000000000
  1.00000000000000
parm1    ps1
 8.630003538054155E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -0.237002616171883      0.000000000000000E+00
parm2    ps2
-8.289453107758495E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -3.623291203678902E-02  0.000000000000000E+00
parm3    ps3
 8.630003538054155E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.523668825832279      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
trace14  rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
circ15   circ
 0.000000000000000E+00   15.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000       3.00000000000000
clear    iden
bip      bip
  20.0000000000000
tip      tip
 0.000000000000000E+00
qaim     aim
  2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
oaim     aim
  2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
qvary    vary
 -1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
ovary    vary
  1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
fit      fit
  1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
 1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
```

```
 end      end
#lines
 ctrip
     1*cq1         1*drs         1*cq2         1*drs         1*cq3
 cspot
     1*ctrip       1*drl
 cq1
     1*parm1       1*cfq1
 cq2
     1*parm2       1*cfq2
 cq3
     1*parm3       1*cfq3
#lumps
#loops
 onebyone
     1*cspot
#labor
    1*fileout
    1*cspot
    1*mapout
    1*qaim
    1*qvary
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*oaim
    1*ovary
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*mapout
    1*fileout
    1*raysin
    1*trace14
    1*clear
    1*raysin
    1*onebyone
    1*circ15
    1*end

*************************************************************************
*              Ordinary ray trace using total map:                      *
*************************************************************************

  200 ray(s) read in from file  13

*************************************************************************
*              Element-by-element ray trace:                            *
*************************************************************************
```

```
  200 ray(s) read in from file  13
circulating through onebyone :
 parm1    cfq1     drs      parm2    cfq2
 drs      parm3    cfq3     drl

end of MARYLIE run
```

Figure 10.3.1.1: Focal spot pattern, for octupole corrected Spot Forming System, produced by two incoming cylinders of parallel rays. Third-order single map calculation.

Figure 10.3.1.2: Focal spot pattern, for octupole corrected Spot Forming System, produced by two incoming cylinders of parallel rays. Third-order element-by-element calculation.

10.3.2 Fitting and Optimizing Seven Octupole Strengths

The higher-order aberrations that produce the finite spot size in figure 10.3.1.2 arise from "cross talk" between the third-order terms associated with the hard-edge fringe fields of the quads and the third-order terms of the various octupoles. In principle these higher-order terms could be minimized by the use of more, but *weaker*, octupoles. Ideally one might consider employing more octupoles with all octupole strengths adjusted in such a way as to remove detrimental third-order aberrations while simultaneously minimizing higher-order aberrations. However, this cannot be done using MaryLie 3.0 since it only computes aberrations through third-order. (It can be done with MaryLie 5.0.) What can be done with MaryLie 3.0 is to adjust *multiple* octupoles to fit the detrimental third-order aberrations to zero (as before) while simultaneously minimizing the sum of the squares of the octupole strengths. In so doing, octupole strengths are generally reduced, the cross talk between them is also reduced, and correspondingly one might hope that higher-order aberrations are also reduced.

This is indeed the case. Exhibit 10.3.2 shows a MaryLie run that carries out the optimization just described. As before in Exhibit 10.3.1, there are 3 combined-function quads with octupole components. In addition, 4 ordinary octupoles are placed around the 3 combined function quadrupoles in the triplet. See the contents of the line *ctrip* in the MaryLie master input file in Exhibit 10.3.2c.

As illustrated in Exhibit 10.3.2a, a user-written merit function is placed in *mrt2*. See section 9.11. This merit function has access to the contents of the array *ucalc* through use of the include statement 'usrdat.inc'. The array *ucalc* is a multipurpose array. In this case the first 7 entries of this array are given the values of the octupole strengths of the 3 combined-function quads and the 4 ordinary octupoles. This is achieved by the use of a "vary" command with JOB=3 and ISTORE=3 to select the octupole strength parameters of the three combined function quads and the four octupoles. See section 9.6. This "vary" command is given the user name *selparm* as a mnemonic for "select parameters". Subsequently a "wsq" command with JOB=2, ISTORE=3, and IFILE=−1 is used to transfer these parameter values to the first seven storage locations in the array *ucalc*. See section 8.27. This command is given the user name *setucalc*.

The response to the "vary" command *selparm* described above could have been made interactively as in Exhibit 10.3.1. However, it can also be obtained from an external file. See section 9.6. In this example an external file (file 19) is employed to illustrate use of this feature. See Exhibit 10.3.2b.

For the MaryLie run itself it is necessary to specify what must be fit in response to the command *fitaim* and what must be optimized in response to the command *optaim*. It is also necessary to specify what must be varied in response to the commands *fitvary* and *optvary*. See the contents of the lines *setup* and *work* in the *#lines* component of the master input file shown in Exhibit 10.3.2g. The responses to these commands could also have been made interactively as in Exhibit 10.3.1. Alternatively, as is done in this example, they may be provided from external files. See sections 9.5 and 9.6 and Exhibits 10.3.2c through 10.3.2f.

The desired fitting and optimization are carried out by invoking the line *setup* followed by two *nested* logical loops (procedures) in *#labor*. The relevant excerpt from *#labor* is shown

below:

setup
bop
bip
clear
cspot
fit
tip
work
opt
top

Fitting is done in an inner procedure which begins with *bip* and terminates with *tip*. Optimization is done in an outer procedure which begins with *bop* and terminates with *top*. See sections 9.1 through 9.4.

NOTE WELL: Each procedure (logical loop) must have within it whatever items are required to compute the quantity that is to be optimized. (In this example these items are specified by *work*.) Simply specifying in *aim* what is to be optimized is not enough. There must be items within the logical loop itself that carry out the necessary associated computations. Failure to provide these items is a common mistake for beginning MaryLie users. If the quantity to be optimized does not actually change during a pass through an optimizing loop (as a result of failing to compute it or for any other reason) the optimizer terminates with the error message "ERROR: Faulty Opt Set Up, Aim Space Point NOT Moving!"

Finally, after optimization is complete, one last fitting procedure is again carried out to ensure that the detrimental aberrations are indeed completely removed for the current setting of the four ordinary octupoles. The relevant excerpt from *#labor* for this procedure is also shown below:

work
bip
clear
cspot
fit
tip

By looking at the output of the MaryLie run, Exhibit 10.3.2g, one sees that before optimization (when the ordinary octupoles are off and the octupole components of the combined function quads have the fitted values found in Exhibit 10.3.1) the merit function has the value 4.7387×10^{-2}. After optimization, when all octupoles are powered, the merit function has been reduced to the value 3.258891×10^{-2}. Does this optimization actually improve the spot forming performance of the system? Figure 10.3.2.1, which should be compared with figure 10.3.1.2, shows the focal spot pattern for the octupole corrected and optimized Spot Forming System. Evidently optimization has reduced the size of the spot pattern by almost a factor of two.

Exhibit 10.3.2a
Merit function for use by opt:

```
      subroutine mrt2(p)
c
      include 'impli.inc'
      include 'merit.inc'
      include 'usrdat.inc'
c
c Calling arrays:
      dimension p(6)
c
c set up sum of squares merit function based
c on contents of ucalc
c
c get strengths of octupole components of cfqds
      res1=ucalc(1)
      res2=ucalc(2)
      res3=ucalc(3)
c get strengths of remaining octupoles
      res4=ucalc(4)
      res5=ucalc(5)
      res6=ucalc(6)
      res7=ucalc(7)
c form sum of squares of cfqds octupole strengths
      sscf=res1**2 + res2**2 + res3**2
c form sum of squares of remaining octupole strengths
      sso=res4**2 + res5**2 + res6**2 + res7**2
c form full sum of squares
      sst = sscf + sso
c divide by 7 (the number of octupoles)
      wss = sst/7.d0
c set value of merit function
      val(2)=wss
c
      return
      end
```

Exhibit 10.3.2b
Contents of file 19 that provides instructions
about what parameters are to be selected in response
to the command selparm:

```
 parm1      5
 parm2      5
 parm3      5
 oct1       2
 oct2       2
 oct3       2
 oct4       2
  #
```

```
Exhibit 10.3.2c
Contents of file 20 that provides instructions
about what aims are to be selected in response
to the command fitaim:

 1    f(140) =      0.000000000E+00
 2    f(149) =      0.000000000E+00
 3    f(195) =      0.000000000E+00
 #


Exhibit 10.3.2d
Contents of file 21 that provides instructions
about what aims are to be selected in response
to the command optaim:

 1       um2 =      0.000000000E+00
 #


Exhibit 10.3.2e
Contents of file 22 that provides instructions
about what parameters are to be selected in response
to the command fitvary:

 parm1      5
 parm2      5
 parm3      5
  #


Exhibit 10.3.2f
Contents of file 23 that provides instructions
about what parameters are to be selected in response
to the command optvary:

 oct1       2
 oct2       2
 oct3       2
 oct4       2
  #

Exhibit 10.3.2g
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.3.2
 This is a MARYLIE run that demonstrates the use of seven octupoles to
```

```
 correct three offensive aberrations in the simple spot forming system
 of Section 2.2.  As in Exhibit 10.3.1, each of the three quadrupoles
 also has an octupole component.  In addition, four octupoles are placed
 around the combined function quadrupoles in the triplet.  This run does
 two things:

  a) It adjusts seven octupole strengths to remove ("zero out") three
     Lie generators corresponding to three third-order geometrical
     aberrations that are detrimental to the performance of a
     spot forming system.  These Lie generators are:

     f(140)=f( 04 00 00 ),
     f(149)=f( 02 02 00 ), and
     f(195)=f( 00 04 00 ).

     At the same time the sum of the squares of all seven octupole
     strengths is minimized.  The optimizer is the quadratic step
     optimizer in opt.  This is done by fitting on the octupole strengths
     of the combined function quadrupoles using an inner logical
     loop (procedure) and optimizing on the strengths of the remaining
     four octupoles in an outer procedure.  The merit function used with
     the optimizer is (1/7)*(the sum of the squares of all seven octupole
     strengths).  This quantity is computed in mrt2.  A "vary" command
     (selparm) with ISTORE=3 is used to select the octupole strength
     parameters of the three combined function quads and the four octupoles.
     A "wsq" command (setucalc) with ISTORE=3 and IFILE=-1 is used to
     transfer these selected parameter values to the first 7 storage
     locations in the array ucalc.  The contents of ucalc are used by mrt2
     to compute the merit function.

  b) To give a rough indication of residual fifth-order aberration
     effects, rays are also traced element-by-element through the
     corrected system.  Doing so illustrates as well use of the
     element-by-element tracking ("turtle") feature of MaryLie
     acomplished with the aid of a loop and a 'circ' command.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.125000000000000
 drs      drft
  0.500000000000000
 drml     drft
   19.6276000000000
 drl      drft
   20.0026000000000
 cfq1     cfqd
```

```
  1.50000000000000       1.00000000000000       1.00000000000000
  1.00000000000000
cfq2     cfqd
  3.00000000000000       2.00000000000000       1.00000000000000
  1.00000000000000
cfq3     cfqd
  1.50000000000000       3.00000000000000       1.00000000000000
  1.00000000000000
oct1     octm
 0.250000000000000      0.000000000000000E+00
oct2     octm
 0.250000000000000      0.000000000000000E+00
oct3     octm
 0.250000000000000      0.000000000000000E+00
oct4     octm
 0.250000000000000      0.000000000000000E+00
parm1    ps1
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -0.237002616171880      0.000000000000000E+00
parm2    ps2
-8.289453107758520E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -3.623291203678830E-02  0.000000000000000E+00
parm3    ps3
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.523668825832260      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
trace14  rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
circ15   circ
 0.000000000000000E+00   15.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000       3.00000000000000
clear    iden
bip      bip
  20.0000000000000
tip      tip
 0.000000000000000E+00
bop      bop
  30.0000000000000
top      top
 0.000000000000000E+00
selparm  vary
  3.00000000000000       19.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
setucalc wsq
  2.00000000000000       3.00000000000000      -1.00000000000000
```

```
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
fitaim   aim
  2.00000000000000       20.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
optaim   aim
  2.00000000000000       21.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   2.00000000000000       1.00000000000000
fitvary  vary
  1.00000000000000       22.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
optvary  vary
  3.00000000000000       23.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   2.00000000000000       1.00000000000000
fit      fit
  1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
 1.000000000000000E-04  0.000000000000000E+00   1.00000000000000
opt      opt
 -11.0000000000000      1.000000000000000E-10  1.000000000000000E-04
 1.000000000000000E-04  0.100000000000000       1.00000000000000
merit2   mrt2
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
wmerit2  wmrt
  2.00000000000000       3.00000000000000
end      end
#lines
ctrip
    1*oct1        1*drvs        1*cq1         1*drvs        1*oct2     &
    1*drvs        1*cq2         1*drvs        1*oct3        1*drvs     &
    1*cq3         1*drvs        1*oct4
cspot
    1*ctrip       1*drml
cq1
    1*parm1       1*cfq1
cq2
    1*parm2       1*cfq2
cq3
    1*parm3       1*cfq3
work
    1*setucalc    1*merit2      1*wmerit2
setup
    1*selparm     1*work        1*fitaim      1*optaim      1*fitvary  &
    1*optvary
#lumps
#loops
onebyone
    1*cspot
#labor
   1*fileout
   1*cspot
   1*mapout
   1*setup
   1*bop
```

```
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*work
    1*opt
    1*top
    1*work
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*work
    1*clear
    1*cspot
    1*mapout
    1*fileout
    1*clear
    1*raysin
    1*onebyone
    1*circ15
    1*end

matrix for map is :

-5.66214E-15  2.47288E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-4.04388E-02  7.93716E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.95399E-14  2.28173E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -4.38264E-02  8.60207E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.50212E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-4.67424231122905E-02
 f( 38)=f( 11 00 01 )=  1.4345989929097
 f( 53)=f( 02 00 01 )= -59.832856603695
 f( 67)=f( 00 20 01 )=-0.11069779980158
 f( 70)=f( 00 11 01 )=  4.6810988356401
 f( 76)=f( 00 02 01 )= -88.881855045722
 f( 83)=f( 00 00 03 )= -398.36503473393
 f( 84)=f( 40 00 00 )=-0.12327420824976
 f( 85)=f( 31 00 00 )=  8.7910508067718
 f( 90)=f( 22 00 00 )= -211.07885008368
 f( 95)=f( 20 20 00 )= 0.68595539915843
 f( 96)=f( 20 11 00 )= -25.219718510723
 f( 99)=f( 20 02 00 )=  221.73626436928
 f(104)=f( 20 00 02 )=-0.19664222372908
 f(105)=f( 13 00 00 )=  1706.8172870474
 f(110)=f( 11 20 00 )= -22.188718833104
 f(111)=f( 11 11 00 )=  736.23108430660
```

```
 f(114)=f( 11 02 00 )= -5357.5423986589
 f(119)=f( 11 00 02 )=  8.0005227170256
 f(140)=f( 04 00 00 )= 7.62074847671101E-10
 f(145)=f( 02 20 00 )=  139.95819449777
 f(146)=f( 02 11 00 )= -3163.2230792366
 f(149)=f( 02 02 00 )=-5.03137442819934E-09
 f(154)=f( 02 00 02 )= -309.42966067238
 f(175)=f( 00 40 00 )=-9.75409414333912E-02
 f(176)=f( 00 31 00 )=  6.5454816213129
 f(179)=f( 00 22 00 )= -146.85444082834
 f(184)=f( 00 20 02 )=-0.59751982937518
 f(185)=f( 00 13 00 )=  1101.5239192044
 f(190)=f( 00 11 02 )=  32.728267833300
 f(195)=f( 00 04 00 )= 7.92969245821951E-10
 f(200)=f( 00 02 02 )= -601.71802504632
 f(209)=f( 00 00 04 )= -1554.3280602063

****************************************************
* Selection of parameters in response to selparm: *
****************************************************

No.  1 is parm1    ps1     .  Parameter 5 out of 6 selected.
    parm1(5) = -0.23700261617188
No.  2 is parm2    ps2     .  Parameter 5 out of 6 selected.
    parm2(5) = -3.62329120367883E-02
No.  3 is parm3    ps3     .  Parameter 5 out of 6 selected.
    parm3(5) =  0.52366882583226
No.  4 is oct1     octm    .  Parameter 2 out of 2 selected.
     oct1(2) =  0.00000000000000E+00
No.  5 is oct2     octm    .  Parameter 2 out of 2 selected.
     oct2(2) =  0.00000000000000E+00
No.  6 is oct3     octm    .  Parameter 2 out of 2 selected.
     oct3(2) =  0.00000000000000E+00
No.  7 is oct4     octm    .  Parameter 2 out of 2 selected.
     oct4(2) =  0.00000000000000E+00

**************************************************
* Value of merit function before optimization: *
**************************************************

value of merit function           2 is  4.738744330507415E-002

*************************************************************************
* Specifying in fitaim that f(140), f(149), and f(195) should be zero: *
*************************************************************************

accept  1:      f(140)
accept  2:      f(149)
accept  3:      f(195)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
```

```
 1    f(140) =      7.620748477E-10     0.000000000E+00
 2    f(149) =     -5.031374428E-09     0.000000000E+00
 3    f(195) =      7.929692458E-10     0.000000000E+00

*****************************************************************************
* Specifying in optaim that um2, the merit function, should be minimized: *
*****************************************************************************

accept  1:         um2

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1       um2 =      4.738744331E-02     0.000000000E+00


*********************************************************
* Specifying in fitvary that the fifth parameters in the *
* parameter sets 1 through 3 should be varied by fit:    *
*********************************************************

No.  1 is parm1    ps1     .  Parameter 5 out of 6 selected.
    parm1(5) = -0.23700261617188
No.  2 is parm2    ps2     .  Parameter 5 out of 6 selected.
    parm2(5) = -3.62329120367883E-02
No.  3 is parm3    ps3     .  Parameter 5 out of 6 selected.
    parm3(5) =  0.52366882583226

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    parm1      ps1          5     -0.23700261617188
 2    parm2      ps2          5     -3.62329120367883E-02
 3    parm3      ps3          5      0.52366882583226

*****************************************************************
* Specifying in optvary that the second parameters in the      *
* octupole elements oct1 through oct4 should be varied by opt: *
*****************************************************************

No.  1 is oct1     octm    .  Parameter 2 out of 2 selected.
     oct1(2) =  0.00000000000000E+00
No.  2 is oct2     octm    .  Parameter 2 out of 2 selected.
     oct2(2) =  0.00000000000000E+00
No.  3 is oct3     octm    .  Parameter 2 out of 2 selected.
     oct3(2) =  0.00000000000000E+00
No.  4 is oct4     octm    .  Parameter 2 out of 2 selected.
     oct4(2) =  0.00000000000000E+00

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    oct1       octm         2      0.00000000000000E+00
```

```
 2    oct2       octm         2      0.00000000000000E+00
 3    oct3       octm         2      0.00000000000000E+00
 4    oct4       octm         2      0.00000000000000E+00

**************************************************
* Beginning of fitting and optimization process: *
**************************************************

**************************
* First fitting process: *
**************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 5.0314E-09,  Step=-2.3700E-05,  SubErr= 5.0314E-09 @cut=    1
-------------------------------------------------------
Iter   2 Error= 1.6360E+01,  Step=-3.6233E-06,  SubErr= 1.6360E+01 @cut=    1
-------------------------------------------------------
Iter   3 Error= 4.2241E+00,  Step= 5.2367E-05,  SubErr= 4.2241E+00 @cut=    1
-------------------------------------------------------
Iter   4 Error= 2.1115E+01,  Step= 5.2367E-05,  SubErr= 2.1115E+01 @cut=    1
-------------------------------------------------------
Iter   5 Error= 2.7610E-10,  Step= 5.6728E-15,  SubErr= 2.7610E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   6 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =     -2.355804440E-11     0.000000000E+00
 2    f(149) =     -3.358735512E-11     0.000000000E+00
 3    f(195) =     -9.745093621E-12     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   parm1      ps1        5     -0.23700261617188
 2   parm2      ps2        5     -3.62329120367892E-02
 3   parm3      ps3        5      0.52366882583228

 Maximum error is     3.358736E-11
 Maximum allowed was  1.000000E-10
value of merit function           2 is  4.738744330507717E-002

***************************************
* Beginning of optimization process: *
***************************************

 OPT is initialized

   1=iter   F=  4.738744330508E-02  DF=  4.738744E-02  DX=  4.072297E-02
```

```
XNEW:  -8.0554614611E-02 -9.2573944632E-02 -0.1525865604      4.0722971665E-02
-------------------------------------------------------

***************************
* Second fitting process: *
***************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 2.9632E+04,  Step=-2.3700E-05,  SubErr= 2.9632E+04 @cut=    1
-------------------------------------------------------
Iter   2 Error= 2.9648E+04,  Step=-3.6233E-06,  SubErr= 2.9648E+04 @cut=    1
-------------------------------------------------------
Iter   3 Error= 2.9636E+04,  Step= 5.2367E-05,  SubErr= 2.9636E+04 @cut=    1
-------------------------------------------------------
Iter   4 Error= 2.9611E+04,  Step= 6.2288E-02,  SubErr= 2.9611E+04 @cut=    1
-------------------------------------------------------
Iter   5 Error= 2.5191E-07,  Step= 4.1437E-12,  SubErr= 2.5191E-07 @cut=    1
-------------------------------------------------------
Iter   6 Error= 1.7298E-10,  Step= 1.5157E-14,  SubErr= 1.7298E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   7 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =      1.481126333E-11     0.000000000E+00
 2    f(149) =      1.996625087E-12     0.000000000E+00
 3    f(195) =     -1.997690902E-11     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   parm1      ps1        5     -0.23860001986872
 2   parm2      ps2        5     -3.13582643927528E-02
 3   parm3      ps3        5      0.58579797083584

 Maximum error is     1.997691E-11
 Maximum allowed was  1.000000E-10
value of merit function           2 is  6.301036755178095E-002

*************************************
* Second pass through the optimizer: *
*************************************


   2=iter   F=  6.301036755178E-02  DF=  1.562292E-02  DX=  9.026208E-02
XNEW:   0.1203632232     -9.0738312807E-02 -1.1053048434E-02  0.1309850496
-------------------------------------------------------

**************************
```

```
* Third fitting process: *
*************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 3.9617E+04,  Step=-2.3860E-05,  SubErr= 3.9617E+04 @cut=    1
-------------------------------------------------------
Iter   2 Error= 3.9601E+04,  Step=-3.1358E-06,  SubErr= 3.9601E+04 @cut=    1
-------------------------------------------------------
Iter   3 Error= 3.9613E+04,  Step= 5.8580E-05,  SubErr= 3.9613E+04 @cut=    1
-------------------------------------------------------
Iter   4 Error= 3.9641E+04,  Step= 5.6977E-02,  SubErr= 3.9641E+04 @cut=    1
-------------------------------------------------------
Iter   5 Error= 2.0548E-07,  Step= 1.2945E-12,  SubErr= 2.0548E-07 @cut=    1
-------------------------------------------------------
Iter   6 Error= 1.0396E-10,  Step= 5.0826E-15,  SubErr= 1.0396E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   7 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-------------------------------------------------------
 1    f(140) =      3.413802574E-11     0.000000000E+00
 2    f(149) =     -6.007638831E-11     0.000000000E+00
 3    f(195) =     -2.334488158E-11     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   parm1      ps1        5     -0.29476105240516
 2   parm2      ps2        5     -3.51579943703615E-02
 3   parm3      ps3        5      0.59467937850852

 Maximum error is     6.007639E-11
 Maximum allowed was  1.000000E-10
value of merit function           2 is  6.882338940091765E-002

******************
* ETC, ETC, ETC. *
******************

  14=iter   F=  3.260264197907E-02  DF= -5.341392E-05  DX=  8.732209E-04
XNEW:  -0.1521671061      0.1813548907      0.1214744146      1.1100786946E-02
-------------------------------------------------------

*************************************************************
* Last fitting process in optimization-fitting procedure: *
*************************************************************


 target           1 =  0.000000000000000E+000
```

```
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 1.3272E+03,  Step=-1.5708E-05,  SubErr= 1.3272E+03 @cut=    1
-------------------------------------------------------
Iter   2 Error= 1.3380E+03,  Step=-4.0540E-06,  SubErr= 1.3380E+03 @cut=    1
-------------------------------------------------------
Iter   3 Error= 1.3319E+03,  Step= 3.6328E-05,  SubErr= 1.3319E+03 @cut=    1
-------------------------------------------------------
Iter   4 Error= 1.3125E+03,  Step= 2.2521E-03,  SubErr= 1.3125E+03 @cut=    1
-------------------------------------------------------
Iter   5 Error= 9.6571E-09,  Step= 1.0386E-12,  SubErr= 9.6571E-09 @cut=    1
-------------------------------------------------------
Quit on iteration   6 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-------------------------------------------------------
 1    f(140) =      8.177281074E-11     0.000000000E+00
 2    f(149) =      8.237321936E-11     0.000000000E+00
 3    f(195) =      1.004352157E-11     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   parm1      ps1        5     -0.15490720001740
 2   parm2      ps2        5     -4.04928706913628E-02
 3   parm3      ps3        5      0.36271588688665

 Maximum error is     8.237322E-11
 Maximum allowed was  1.000000E-10
value of merit function           2 is  3.258891617766553E-002

************************************
* Last pass through the optimizer: *
************************************

  15=iter   F=  3.258891617767E-02  DF= -1.372580E-05  DX=  7.954995E-05
XNEW:  -0.1522469818      0.1813821208      0.1213325660      1.1180336891E-02
-------------------------------------------------------

done
relative gradient is less than gtol.
After   15 total iterations, final values are:
Final RMS error for loop 2 is  3.2588916178E-02

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   oct1       octm       2     -0.15224698182631
 2   oct2       octm       2      0.18138212079769
 3   oct3       octm       2      0.12133256602590
 4   oct4       octm       2      1.11803368908222E-02
```

```
 Final change was -1.372580E-05  final step was  7.954995E-05
 Tolerances: ftol= 1.000000E-10            xtol= 1.000000E-04
value of merit function          2 is  3.258913378939155E-002

**********************************************
* Final fitting procedure after optimization: *
**********************************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 1.3039E+01,  Step=-1.5491E-05,  SubErr= 1.3039E+01 @cut=    1
-------------------------------------------------------
Iter   2 Error= 2.3733E+01,  Step=-4.0493E-06,  SubErr= 2.3733E+01 @cut=    1
-------------------------------------------------------
Iter   3 Error= 1.7760E+01,  Step= 3.6272E-05,  SubErr= 1.7760E+01 @cut=    1
-------------------------------------------------------
Iter   4 Error= 1.5858E+00,  Step= 3.5221E-05,  SubErr= 1.5858E+00 @cut=    1
-------------------------------------------------------
Quit on iteration   5 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-------------------------------------------------
 1    f(140) =     -9.404033108E-12     0.000000000E+00
 2    f(149) =      1.063682475E-11     0.000000000E+00
 3    f(195) =     -1.996980359E-11     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   parm1      ps1        5     -0.15489553304785
 2   parm2      ps2        5     -4.04896976522865E-02
 3   parm3      ps3        5      0.36271907726308

 Maximum error is     1.996980E-11
 Maximum allowed was  1.000000E-10

*********************************************************************
* Final value of merit function for optimized and corrected system: *
*********************************************************************

value of merit function          2 is  3.258891135949125E-002

*****************************************************************
* Transfer map for optimized and corrected Spot Forming System: *
*****************************************************************

matrix for map is :

-5.66214E-15  2.47288E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
-4.04388E-02  7.93716E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.95399E-14  2.28173E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -4.38264E-02  8.60207E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.50212E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-4.67424231122905E-02
 f( 38)=f( 11 00 01 )=  1.4345989929097
 f( 53)=f( 02 00 01 )= -59.832856603695
 f( 67)=f( 00 20 01 )=-0.11069779980158
 f( 70)=f( 00 11 01 )=  4.6810988356401
 f( 76)=f( 00 02 01 )= -88.881855045722
 f( 83)=f( 00 00 03 )= -398.36503473393
 f( 84)=f( 40 00 00 )=-8.89083353541492E-02
 f( 85)=f( 31 00 00 )=  6.3613003164667
 f( 90)=f( 22 00 00 )= -153.33711587259
 f( 95)=f( 20 20 00 )= 0.49139317427674
 f( 96)=f( 20 11 00 )= -18.071264442261
 f( 99)=f( 20 02 00 )=  158.73936098854
 f(104)=f( 20 00 02 )=-0.19664222372908
 f(105)=f( 13 00 00 )=  1245.5445056207
 f(110)=f( 11 20 00 )= -15.974726045653
 f(111)=f( 11 11 00 )=  530.15850256210
 f(114)=f( 11 02 00 )= -3846.3731133617
 f(119)=f( 11 00 02 )=  8.0005227170256
 f(140)=f( 04 00 00 )=-9.40403310778493E-12
 f(145)=f( 02 20 00 )=  101.88774855927
 f(146)=f( 02 11 00 )= -2311.9488850312
 f(149)=f( 02 02 00 )= 1.06368247543287E-11
 f(154)=f( 02 00 02 )= -309.42966067238
 f(175)=f( 00 40 00 )=-6.95583699768062E-02
 f(176)=f( 00 31 00 )=  4.6744564689799
 f(179)=f( 00 22 00 )= -105.05879704592
 f(184)=f( 00 20 02 )=-0.59751982937518
 f(185)=f( 00 13 00 )=  789.61713892920
 f(190)=f( 00 11 02 )=  32.728267833300
 f(195)=f( 00 04 00 )=-1.99698035885376E-11
 f(200)=f( 00 02 02 )= -601.71802504632
 f(209)=f( 00 00 04 )= -1554.3280602063

************************************
* Writing out new master input file *
* with optimized parameter values:  *
************************************

#comment
 Exhibit 10.3.2
 This is a MARYLIE run that demonstrates the use of seven octupoles to
 correct three offensive aberrations in the simple spot forming system
 of Section 2.2.  As in Exhibit 10.3.1, each of the three quadrupoles
 also has an octupole component.  In addition, four octupoles are placed
```

```
 around the combined function quadrupoles in the triplet.  This run does
 two things:

  a) It adjusts seven octupole strengths to remove ("zero out") three
     Lie generators corresponding to three third-order geometrical
     aberrations that are detrimental to the performance of a
     spot forming system.  These Lie generators are:

     f(140)=f( 04 00 00 ),
     f(149)=f( 02 02 00 ), and
     f(195)=f( 00 04 00 ).

     At the same time the sum of the squares of all seven octupole
     strengths is minimized.  The optimizer is the quadratic step
     optimizer in opt.  This is done by fitting on the octupole strengths
     of the combined function quadrupoles using an inner logical
     loop (procedure) and optimizing on the strengths of the remaining
     four octupoles in an outer procedure.  The merit function used with
     the optimizer is (1/7)*(the sum of the squares of all seven octupole
     strengths).  This quantity is computed in mrt2.  A "vary" command
     (selparm) with ISTORE=3 is used to select the octupole strength
     parameters of the three combined function quads and the four octupoles.
     A "wsq" command (setucalc) with ISTORE=3 and IFILE=-1 is used to
     transfer these selected parameter values to the first 7 storage
     locations in the array ucalc.  The contents of ucalc are used by mrt2
     to compute the merit function.

  b) To give a rough indication of residual fifth-order aberration
     effects, rays are also traced element-by-element through the
     corrected system.  Doing so illustrates as well use of the
     element-by-element tracking ("turtle") feature of MaryLie
     acomplished with the aid of a loop and a 'circ' command.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.125000000000000
 drs      drft
  0.500000000000000
 drml     drft
   19.6276000000000
 drl      drft
   20.0026000000000
 cfq1     cfqd
   1.50000000000000       1.00000000000000       1.00000000000000
   1.00000000000000
 cfq2     cfqd
```

```
  3.00000000000000       2.00000000000000       1.00000000000000
  1.00000000000000
cfq3     cfqd
  1.50000000000000       3.00000000000000       1.00000000000000
  1.00000000000000
oct1     octm
 0.250000000000000     -0.152246981826309
oct2     octm
 0.250000000000000      0.181382120797693
oct3     octm
 0.250000000000000      0.121332566025899
oct4     octm
 0.250000000000000      1.118033689082220E-02
parm1    ps1
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -0.154895533047847      0.000000000000000E+00
parm2    ps2
-8.289453107758520E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -4.048969765228652E-02  0.000000000000000E+00
parm3    ps3
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.362719077263076      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
trace14  rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
circ15   circ
 0.000000000000000E+00   15.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000       3.00000000000000
clear    iden
bip      bip
  20.0000000000000
tip      tip
 0.000000000000000E+00
bop      bop
  30.0000000000000
top      top
 0.000000000000000E+00
selparm  vary
  3.00000000000000       19.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
setucalc wsq
  2.00000000000000       3.00000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
fitaim   aim
  2.00000000000000       20.0000000000000      0.000000000000000E+00
```

```
 0.000000000000000E+00   1.00000000000000       1.00000000000000
optaim   aim
  2.00000000000000       21.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   2.00000000000000       1.00000000000000
fitvary  vary
  1.00000000000000       22.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
optvary  vary
  3.00000000000000       23.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   2.00000000000000       1.00000000000000
fit      fit
  1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
 1.000000000000000E-04  0.000000000000000E+00   1.00000000000000
opt      opt
 -11.0000000000000      1.000000000000000E-10  1.000000000000000E-04
 1.000000000000000E-04  0.100000000000000       1.00000000000000
merit2   mrt2
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
wmerit2  wmrt
  2.00000000000000       3.00000000000000
end      end
#lines
 ctrip
     1*oct1        1*drvs        1*cq1         1*drvs        1*oct2     &
     1*drvs        1*cq2         1*drvs        1*oct3        1*drvs     &
     1*cq3         1*drvs        1*oct4
 cspot
     1*ctrip       1*drml
 cq1
     1*parm1       1*cfq1
 cq2
     1*parm2       1*cfq2
 cq3
     1*parm3       1*cfq3
 work
     1*setucalc    1*merit2      1*wmerit2
 setup
     1*selparm     1*work        1*fitaim      1*optaim      1*fitvary  &
     1*optvary
#lumps
#loops
 onebyone
     1*cspot
#labor
    1*fileout
    1*cspot
    1*mapout
    1*setup
    1*bop
    1*bip
    1*clear
    1*cspot
```

```
    1*fit
    1*tip
    1*work
    1*opt
    1*top
    1*work
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*work
    1*clear
    1*cspot
    1*mapout
    1*fileout
    1*clear
    1*raysin
    1*onebyone
    1*circ15
    1*end

*********************************
* Element-by element ray trace: *
*********************************

  200 ray(s) read in from file  13
circulating through onebyone :
 oct1     drvs     parm1    cfq1     drvs
 oct2     drvs     parm2    cfq2     drvs
 oct3     drvs     parm3    cfq3     drvs
 oct4     drml

end of MARYLIE run
```

Figure 10.3.2.3: Focal spot pattern, for octupole corrected and optimized Spot Forming System, produced by two incoming cylinders of parallel rays. Third-order element-by-element calculation.

10.3.3 Optimizing Using the MSS Optimizer

Since the merit function in the last example was a sum of squares, see Exhibit 10.3.2a, it is tempting to carry out optimization using the nonlinear least squares optimizer in *mss* that is designed for this purpose (see section 9.8) rather than *opt*. Exhibits 10.3.3a and 10.3.3b describe a MaryLie run that carries out this alternate approach. As before external files 19, 20, 22, and 23, that are identical to those shown in Exhibits 10.3.2b, 10.3.2c, 10.3.2e, and 10.3.2f, respectively, are used to respond to the commands *selparm*, *fitaim*, *fitvary*, and *optvary*, respectively. In particular, the contents of *ucalc* are established as before. However, the response to *optaim*, given by the contents of file 21, differs as shown in Exhibit 10.3.3a. Now the weighted sum of squares merit function is set up directly from the contents of *ucalc* by *optaim* using JOB=3. (The optimizer *mss* requires that a weighted sum of squares merit function be set up in *aim* because it makes use of the ingredients of the merit function as well as its total value.) See sections 9.5, 9.8, and 9.10.

Exhibit 10.3.b shows the MaryLie run itself. Evidently the final fitted and optimized octupole values are essentially identical to those found in Exhibit 10.3.2f. However, only

6 passes through the *mss* optimizer were required in contrast to the 15 passes through *opt* in Exhibit 10.3.2f. We conclude that, at least for this example, the nonlinear least squares optimizer in *mss* is indeed superior when the merit function is a weighted sum of squares.

NOTE WELL: Each procedure (logical loop) must have within it whatever items are required to compute the quantities that are to be optimized. (In this example these items are specified by *work*.) Simply specifying in *aim* what is to be optimized is not enough. There must be items within the logical loop itself that carry out the necessary associated computations. Failure to provide these items is a common mistake for beginning MaryLie users. If the quantity to be optimized does not actually change during a pass through a optimizing loop (as a result of failing to compute it or for any other reason) the optimizer terminates with the error message "ERROR: Faulty Mss Set Up, Aim Space Point NOT Moving!"

Two comments, one about programming and one about physics, can be made at this point. With regard to programming, if one is content to use weighted-sum-of-squares merit functions, then they can always be set up for use with either *mss* or *opt* simply by using *aim* with JOB=3 as was done in this example. That is, there is then no need to write merit function routines as in Exhibit 10.3.2a. See sections 9.5 and 9.9. With regard to physics, it is evident from Exhibits 10.3.2g and 10.3.3b that, after optimization (sum of squares minimization), the octupole strengths of cfqd2 and oct4 are rather small compared to the others. This suggests these elements are rather *ineffective* as correctors. (After sum-of-squares minimization, correctors with large strengths must be effective since there would be otherwise no reason for them to be large considering that their large values have the undesirable feature of making the sum-of-squares merit function large. Correspondingly, weak correctors must be ineffective.) Consequently, it would be worth exploring the optimized solution when only 5 octupole correctors (those in cdqd1 and cfqd3 and oct1, oct2, and oct3) are used. This solution is likely to be almost as good as that with 7 octupole correctors.

```
Exhibit 10.3.3a
Contents of file 21 that provides instructions
about what aims are to be selected in response
to the command optaim:

 1    u(  1) =      0.000000000E+00      1.00000000
 2    u(  2) =      0.000000000E+00      1.00000000
 3    u(  3) =      0.000000000E+00      1.00000000
 4    u(  4) =      0.000000000E+00      1.00000000
 5    u(  5) =      0.000000000E+00      1.00000000
 6    u(  6) =      0.000000000E+00      1.00000000
 7    u(  7) =      0.000000000E+00      1.00000000
 #

Exhibit 10.3.3b
***MARYLIE 3.0***
Prerelease Development Version 1/12/00
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
```

```
#comment
 Exhibit 10.3.3
 This is a MARYLIE run that demonstrates the use of seven octupoles to
 correct three offensive aberrations in the simple spot forming system
 of Section 2.2.  It is similar to that in Exhibit 10.3.2 except that
 the sum of squares merit function is set up in aim and the mss optimizer
 is used in place of opt.  Specifically, this run adjusts seven octupole
 strengths to remove ("zero out") three Lie generators corresponding to
 three third-order geometrical aberrations that are detrimental to the
 performance of a spot forming system.  These Lie generators are:

     f(140)=f( 04 00 00 ),
     f(149)=f( 02 02 00 ), and
     f(195)=f( 00 04 00 ).

 At the same time the sum of the squares of all seven octupole
 strengths is minimized.  The optimizer is the nonlinear least squares
 optimizer in mss.  As in Exhibit 10.3.2, this is done by fitting on
 the octupole strengths of the combined function quadrupoles using an
 inner logical loop (procedure) and optimizing on the strengths of the
 remaining four octupoles in an outer procedure.  A "vary" command
 (selparm) with ISTORE=3 is used to select the octupole strength parameters
 of the three combined function quads and the four octupoles.  A "wsq"
 command (setucalc) with ISTORE=3 and IFILE=-1 is used to transfer these
 selected parameter values to the first 7 storage locations in the array
 ucalc.  A merit function, which is again (1/7)*(the sum of the squares of
 all seven octupole strengths) and is available as mrt0, is set up in optaim
 (with JOB=3) based on the entries in ucalc.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.125000000000000
 drs      drft
  0.500000000000000
 drml     drft
   19.6276000000000
 drl      drft
   20.0026000000000
 cfq1     cfqd
   1.50000000000000       1.00000000000000       1.00000000000000
   1.00000000000000
 cfq2     cfqd
   3.00000000000000       2.00000000000000       1.00000000000000
   1.00000000000000
 cfq3     cfqd
   1.50000000000000       3.00000000000000       1.00000000000000
```

```
  1.00000000000000
oct1     octm
 0.250000000000000      0.000000000000000E+00
oct2     octm
 0.250000000000000      0.000000000000000E+00
oct3     octm
 0.250000000000000      0.000000000000000E+00
oct4     octm
 0.250000000000000      0.000000000000000E+00
parm1    ps1
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -0.237002616171880      0.000000000000000E+00
parm2    ps2
-8.289453107758520E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -3.623291203678830E-02  0.000000000000000E+00
parm3    ps3
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.523668825832260      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
trace14  rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
circ15   circ
 0.000000000000000E+00   15.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000       3.00000000000000
clear    iden
bip      bip
  20.0000000000000
tip      tip
 0.000000000000000E+00
bop      bop
  30.0000000000000
top      top
 0.000000000000000E+00
selparm  vary
  3.00000000000000       19.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
setucalc wsq
  2.00000000000000       3.00000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
fitaim   aim
  2.00000000000000       20.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
optaim   aim
  3.00000000000000       21.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   2.00000000000000       1.00000000000000
```

```
 fitvary  vary
   1.00000000000000       22.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 optvary  vary
   3.00000000000000       23.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   2.00000000000000       1.00000000000000
 fit      fit
   1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
  1.000000000000000E-04  0.000000000000000E+00   1.00000000000000
 mss      mss
  -1.00000000000000      1.000000000000000E-10  1.000000000000000E-04
  1.000000000000000E-04  0.100000000000000       1.00000000000000
 merit0   mrt0
   1.00000000000000       2.00000000000000
 wmerit0  wmrt
  0.000000000000000E+00   3.00000000000000
 merit2   mrt2
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 wmerit2  wmrt
   2.00000000000000       3.00000000000000
 end      end
#lines
 ctrip
     1*oct1        1*drvs        1*cq1         1*drvs        1*oct2     &
     1*drvs        1*cq2         1*drvs        1*oct3        1*drvs     &
     1*cq3         1*drvs        1*oct4
 cspot
     1*ctrip       1*drml
 cq1
     1*parm1       1*cfq1
 cq2
     1*parm2       1*cfq2
 cq3
     1*parm3       1*cfq3
 work
     1*setucalc    1*merit0      1*wmerit0
 setup
     1*selparm     1*fitaim      1*setucalc    1*optaim      1*work     &
     1*fitvary     1*optvary
#lumps
#loops
 onebyone
     1*cspot
#labor
    1*fileout
    1*cspot
    1*mapout
    1*setup
    1*bop
    1*bip
    1*clear
    1*cspot
```

```
    1*fit
    1*tip
    1*work
    1*mss
    1*top
    1*work
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*work
    1*clear
    1*cspot
    1*mapout
    1*fileout
    1*end

matrix for map is :

-5.66214E-15  2.47288E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-4.04388E-02  7.93716E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.95399E-14  2.28173E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -4.38264E-02  8.60207E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.50212E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-4.67424231122905E-02
 f( 38)=f( 11 00 01 )=  1.4345989929097
 f( 53)=f( 02 00 01 )= -59.832856603695
 f( 67)=f( 00 20 01 )=-0.11069779980158
 f( 70)=f( 00 11 01 )=  4.6810988356401
 f( 76)=f( 00 02 01 )= -88.881855045722
 f( 83)=f( 00 00 03 )= -398.36503473393
 f( 84)=f( 40 00 00 )=-0.12327420824976
 f( 85)=f( 31 00 00 )=  8.7910508067718
 f( 90)=f( 22 00 00 )= -211.07885008368
 f( 95)=f( 20 20 00 )= 0.68595539915843
 f( 96)=f( 20 11 00 )= -25.219718510723
 f( 99)=f( 20 02 00 )=  221.73626436928
 f(104)=f( 20 00 02 )=-0.19664222372908
 f(105)=f( 13 00 00 )=  1706.8172870474
 f(110)=f( 11 20 00 )= -22.188718833104
 f(111)=f( 11 11 00 )=  736.23108430660
 f(114)=f( 11 02 00 )= -5357.5423986589
 f(119)=f( 11 00 02 )=  8.0005227170256
 f(140)=f( 04 00 00 )= 7.62074847671101E-10
 f(145)=f( 02 20 00 )=  139.95819449777
 f(146)=f( 02 11 00 )= -3163.2230792366
 f(149)=f( 02 02 00 )=-5.03137442819934E-09
 f(154)=f( 02 00 02 )= -309.42966067238
```

```
 f(175)=f( 00 40 00 )=-9.75409414333912E-02
 f(176)=f( 00 31 00 )=  6.5454816213129
 f(179)=f( 00 22 00 )= -146.85444082834
 f(184)=f( 00 20 02 )=-0.59751982937518
 f(185)=f( 00 13 00 )=  1101.5239192044
 f(190)=f( 00 11 02 )=  32.728267833300
 f(195)=f( 00 04 00 )= 7.92969245821951E-10
 f(200)=f( 00 02 02 )= -601.71802504632
 f(209)=f( 00 00 04 )= -1554.3280602063

***************************************************
* Selection of parameters in response to selparm: *
***************************************************

No.  1 is parm1    ps1     .  Parameter 5 out of 6 selected.
    parm1(5) = -0.23700261617188
No.  2 is parm2    ps2     .  Parameter 5 out of 6 selected.
    parm2(5) = -3.62329120367883E-02
No.  3 is parm3    ps3     .  Parameter 5 out of 6 selected.
    parm3(5) =  0.52366882583226
No.  4 is oct1     octm    .  Parameter 2 out of 2 selected.
     oct1(2) =  0.00000000000000E+00
No.  5 is oct2     octm    .  Parameter 2 out of 2 selected.
     oct2(2) =  0.00000000000000E+00
No.  6 is oct3     octm    .  Parameter 2 out of 2 selected.
     oct3(2) =  0.00000000000000E+00
No.  7 is oct4     octm    .  Parameter 2 out of 2 selected.
     oct4(2) =  0.00000000000000E+00

*************************************************************************
* Specifying in fitaim that f(140), f(149), and f(195) should be zero: *
*************************************************************************

accept  1:      f(140)
accept  2:      f(149)
accept  3:      f(195)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =      7.620748477E-10     0.000000000E+00
 2    f(149) =     -5.031374428E-09     0.000000000E+00
 3    f(195) =      7.929692458E-10     0.000000000E+00

*****************************************************************
* Setting up weighted sum of squares merit function in optaim: *
*****************************************************************

accept  1:      u(  1)
accept  2:      u(  2)
accept  3:      u(  3)
accept  4:      u(  4)
accept  5:      u(  5)
```

```
accept  6:      u(  6)
accept  7:      u(  7)

 Aims selected :
No.     item        present value        target value        weights
-----------------------------------------------------------------------
 1    u(  1) =     -0.237002616         0.000000000E+00        1.0000
 2    u(  2) =     -3.623291204E-02     0.000000000E+00        1.0000
 3    u(  3) =      0.523668826         0.000000000E+00        1.0000
 4    u(  4) =      0.000000000E+00     0.000000000E+00        1.0000
 5    u(  5) =      0.000000000E+00     0.000000000E+00        1.0000
 6    u(  6) =      0.000000000E+00     0.000000000E+00        1.0000
 7    u(  7) =      0.000000000E+00     0.000000000E+00        1.0000

************************************************
* Value of merit function before optimization. *
************************************************

value of merit function           0 is  4.738744330507415E-002

*************************************************************
* Specifying in fitvary that the fifth parameters in the *
* parameter sets 1 through 3 should be varied by fit:    *
*************************************************************

No.  1 is parm1    ps1     .  Parameter 5 out of 6 selected.
    parm1(5) = -0.23700261617188
No.  2 is parm2    ps2     .  Parameter 5 out of 6 selected.
    parm2(5) = -3.62329120367883E-02
No.  3 is parm3    ps3     .  Parameter 5 out of 6 selected.
    parm3(5) =  0.52366882583226

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    parm1      ps1          5     -0.23700261617188
 2    parm2      ps2          5     -3.62329120367883E-02
 3    parm3      ps3          5      0.52366882583226

*****************************************************************
* Specifying in optvary that the second parameters in the      *
* octupole elements oct1 through oct4 should be varied by mss: *
*****************************************************************

No.  1 is oct1     octm    .  Parameter 2 out of 2 selected.
     oct1(2) =  0.00000000000000E+00
No.  2 is oct2     octm    .  Parameter 2 out of 2 selected.
     oct2(2) =  0.00000000000000E+00
No.  3 is oct3     octm    .  Parameter 2 out of 2 selected.
     oct3(2) =  0.00000000000000E+00
No.  4 is oct4     octm    .  Parameter 2 out of 2 selected.
     oct4(2) =  0.00000000000000E+00
```

```
  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    oct1       octm         2      0.00000000000000E+00
 2    oct2       octm         2      0.00000000000000E+00
 3    oct3       octm         2      0.00000000000000E+00
 4    oct4       octm         2      0.00000000000000E+00

**************************************************
* Beginning of fitting and optimization process: *
**************************************************

**************************
* First fitting process: *
**************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 5.0314E-09,  Step=-2.3700E-05,  SubErr= 5.0314E-09 @cut=    1
-------------------------------------------------------
Iter   2 Error= 1.6360E+01,  Step=-3.6233E-06,  SubErr= 1.6360E+01 @cut=    1
-------------------------------------------------------
Iter   3 Error= 4.2241E+00,  Step= 5.2367E-05,  SubErr= 4.2241E+00 @cut=    1
-------------------------------------------------------
Iter   4 Error= 2.1115E+01,  Step= 5.2367E-05,  SubErr= 2.1115E+01 @cut=    1
-------------------------------------------------------
Iter   5 Error= 2.7610E-10,  Step= 5.6728E-15,  SubErr= 2.7610E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   6 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =     -2.355804440E-11     0.000000000E+00
 2    f(149) =     -3.358735512E-11     0.000000000E+00
 3    f(195) =     -9.745093621E-12     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------
 1   parm1      ps1        5     -0.23700261617188
 2   parm2      ps2        5     -3.62329120367892E-02
 3   parm3      ps3        5      0.52366882583228

 Maximum error is     3.358736E-11
 Maximum allowed was  1.000000E-10
value of merit function           0 is  4.738744330507717E-002

**************************************
* Beginning of optimization process: *
**************************************
```

```
 MSS is initialized

 MSS iteration    1 F=   0.217686571      DF=  0.217687
XNEW:
-------------------------------------------------------
  optimizer return:           0=info
  XV= -8.055461461143337E-002 -9.257394463233393E-002 -0.152586560413296
 4.072297166529814E-002

***************************
* Second fitting process: *
***************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 2.9632E+04,  Step=-2.3700E-05,  SubErr= 2.9632E+04 @cut=    1
-------------------------------------------------------
Iter   2 Error= 2.9648E+04,  Step=-3.6233E-06,  SubErr= 2.9648E+04 @cut=    1
-------------------------------------------------------
Iter   3 Error= 2.9636E+04,  Step= 5.2367E-05,  SubErr= 2.9636E+04 @cut=    1
-------------------------------------------------------
Iter   4 Error= 2.9611E+04,  Step= 6.2288E-02,  SubErr= 2.9611E+04 @cut=    1
-------------------------------------------------------
Iter   5 Error= 2.5191E-07,  Step= 4.1437E-12,  SubErr= 2.5191E-07 @cut=    1
-------------------------------------------------------
Iter   6 Error= 1.7298E-10,  Step= 1.5157E-14,  SubErr= 1.7298E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   7 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =      1.481126333E-11     0.000000000E+00
 2    f(149) =      1.996625087E-12     0.000000000E+00
 3    f(195) =     -1.997690902E-11     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   parm1      ps1        5     -0.23860001986872
 2   parm2      ps2        5     -3.13582643927528E-02
 3   parm3      ps3        5      0.58579797083584

 Maximum error is     1.997691E-11
 Maximum allowed was  1.000000E-10
value of merit function           0 is  6.301036755178098E-002

*********************************************************
* ETC. ETC. ETC.                                        *
*********************************************************
```

```
 MSS iteration    5 F=   0.219037440      DF=  1.954982E-03
XNEW:   -0.138875126      -2.607438137E-02  -2.042575327E-03  -0.155860897
-------------------------------------------------------
  optimizer return:           0=info
  XV= -0.152246981826715       0.181382120797905       0.121332566025053
 1.118033689145392E-002
 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 4.0896E+04,  Step=-1.8731E-05,  SubErr= 4.0896E+04 @cut=    1
-------------------------------------------------------
Iter   2 Error= 4.0883E+04,  Step=-3.5496E-06,  SubErr= 4.0883E+04 @cut=    1
-------------------------------------------------------
Iter   3 Error= 4.0892E+04,  Step= 5.0521E-05,  SubErr= 4.0892E+04 @cut=    1
-------------------------------------------------------
Iter   4 Error= 4.0916E+04,  Step= 1.4626E-01,  SubErr= 4.0916E+04 @cut=    1
-------------------------------------------------------
Iter   5 Error= 4.7552E-07,  Step= 1.8134E-11,  SubErr= 4.7552E-07 @cut=    1
-------------------------------------------------------
Iter   6 Error= 1.2387E-10,  Step= 4.1088E-15,  SubErr= 1.2387E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   7 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =     -2.952660338E-11     0.000000000E+00
 2    f(149) =      7.305089866E-11     0.000000000E+00
 3    f(195) =      6.462386182E-12     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   parm1      ps1        5     -0.15489553304781
 2   parm2      ps2        5     -4.04896976522688E-02
 3   parm3      ps3        5      0.36271907726309

 Maximum error is     7.305090E-11
 Maximum allowed was  1.000000E-10
value of merit function           0 is  3.258891135949227E-002

*******************************
* Last pass through optimizer: *
*******************************

 MSS iteration    6 F=   0.180523991      DF= -3.851345E-02
XNEW:   -0.152246982       0.181382121       0.121332566       1.118033689E-02
-------------------------------------------------------
  optimizer return:           8=info
  XV= -0.152246981826680       0.181382120797860       0.121332566025083
 1.118033689140618E-002
```

```
done
absolute x error lies within stol.
After    6 total iterations, final values are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    u(  1) =     -0.154895533         0.000000000E+00
 2    u(  2) =     -4.048969765E-02     0.000000000E+00
 3    u(  3) =      0.362719077         0.000000000E+00
 4    u(  4) =     -0.152246982         0.000000000E+00
 5    u(  5) =      0.181382121         0.000000000E+00
 6    u(  6) =      0.121332566         0.000000000E+00
 7    u(  7) =      1.118033689E-02     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   oct1       octm       2     -0.15224698182668
 2   oct2       octm       2      0.18138212079786
 3   oct3       octm       2      0.12133256602508
 4   oct4       octm       2      1.11803368914062E-02

 Final change was    -3.851345E-02
value of merit function           0 is  3.258891135948936E-002

***********************************************
* Final fitting procedure after optimization: *
***********************************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
Iter   1 Error= 1.1144E-09,  Step=-1.5490E-05,  SubErr= 1.1144E-09 @cut=    1
-------------------------------------------------------
Iter   2 Error= 1.0693E+01,  Step=-4.0490E-06,  SubErr= 1.0693E+01 @cut=    1
-------------------------------------------------------
Iter   3 Error= 4.7204E+00,  Step= 3.6272E-05,  SubErr= 4.7204E+00 @cut=    1
-------------------------------------------------------
Iter   4 Error= 1.4625E+01,  Step= 3.6272E-05,  SubErr= 1.4625E+01 @cut=    1
-------------------------------------------------------
Iter   5 Error= 1.1772E-10,  Step= 3.7818E-15,  SubErr= 1.1772E-10 @cut=    1
-------------------------------------------------------
Quit on iteration   6 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =     -4.686029342E-12     0.000000000E+00
 2    f(149) =      8.987655065E-11     0.000000000E+00
 3    f(195) =      2.420463829E-11     0.000000000E+00
```

```
 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   parm1      ps1        5     -0.15489553304782
 2   parm2      ps2        5     -4.04896976522696E-02
 3   parm3      ps3        5      0.36271907726312

 Maximum error is     8.987655E-11
 Maximum allowed was  1.000000E-10

*********************************************************************
* Final value of merit function for optimized and corrected system: *
*********************************************************************

value of merit function           0 is  3.258891135949328E-002

matrix for map is :

-5.66214E-15  2.47288E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-4.04388E-02  7.93716E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.95399E-14  2.28173E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -4.38264E-02  8.60207E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  2.50212E+02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=-4.67424231122905E-02
 f( 38)=f( 11 00 01 )=  1.4345989929097
 f( 53)=f( 02 00 01 )= -59.832856603695
 f( 67)=f( 00 20 01 )=-0.11069779980158
 f( 70)=f( 00 11 01 )=  4.6810988356401
 f( 76)=f( 00 02 01 )= -88.881855045722
 f( 83)=f( 00 00 03 )= -398.36503473393
 f( 84)=f( 40 00 00 )=-8.89083353541736E-02
 f( 85)=f( 31 00 00 )=  6.3613003164683
 f( 90)=f( 22 00 00 )= -153.33711587263
 f( 95)=f( 20 20 00 )= 0.49139317427684
 f( 96)=f( 20 11 00 )= -18.071264442264
 f( 99)=f( 20 02 00 )=  158.73936098856
 f(104)=f( 20 00 02 )=-0.19664222372908
 f(105)=f( 13 00 00 )=  1245.5445056211
  f(110)=f( 11 20 00 )= -15.974726045656
 f(111)=f( 11 11 00 )=  530.15850256220
 f(114)=f( 11 02 00 )= -3846.3731133623
 f(119)=f( 11 00 02 )=  8.0005227170256
 f(140)=f( 04 00 00 )=-4.68602934233786E-12
 f(145)=f( 02 20 00 )=  101.88774855930
 f(146)=f( 02 11 00 )= -2311.9488850317
 f(149)=f( 02 02 00 )= 8.98765506462951E-11
 f(154)=f( 02 00 02 )= -309.42966067238
 f(175)=f( 00 40 00 )=-6.95583699768157E-02
```

```
 f(176)=f( 00 31 00 )=  4.6744564689805
 f(179)=f( 00 22 00 )= -105.05879704593
 f(184)=f( 00 20 02 )=-0.59751982937518
 f(185)=f( 00 13 00 )=  789.61713892929
 f(190)=f( 00 11 02 )=  32.728267833300
 f(195)=f( 00 04 00 )= 2.42046382936678E-11
 f(200)=f( 00 02 02 )= -601.71802504632
 f(209)=f( 00 00 04 )= -1554.3280602063
#comment
 Exhibit 10.3.3.
 This is a MARYLIE run that demonstrates the use of seven octupoles to
 correct three offensive aberrations in the simple spot forming system
 of Section 2.2.  It is similar to that in Exhibit 10.3.2 except that
 the sum of squares merit function is set up in aim and the mss optimizer
 is used in place of opt.  Specifically, this run adjusts seven octupole
 strengths to remove ("zero out") three Lie generators corresponding to
 three third-order geometrical aberrations that are detrimental to the
 performance of a spot forming system.  These Lie generators are:

     f(140)=f( 04 00 00 ),
     f(149)=f( 02 02 00 ), and
     f(195)=f( 00 04 00 ).

 At the same time the sum of the squares of all seven octupole
 strengths is minimized.  The optimizer is the nonlinear least squares
 optimizer in mss.  As in Exhibit 10.3.2, this is done by fitting on
 the octupole strengths of the combined function quadrupoles using an
 inner logical loop (procedure) and optimizing on the strengths of the
 remaining four octupoles in an outer procedure.  A "vary" command
 (selparm) with ISTORE=3 is used to select the octupole strength parameters
 of the three combined function quads and the four octupoles.  A "wsq"
 command (setucalc) with ISTORE=3 and IFILE=-1 is used to transfer these
 selected parameter values to the first 7 storage locations in the array
 ucalc.  A merit function, which is again (1/7)*(the sum of the squares of
 all seven octupole strengths) and is available as mrt0, is set up in optaim
 (with JOB=3) based on the entries in ucalc.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.125000000000000
 drs      drft
  0.500000000000000
 drml     drft
   19.6276000000000
 drl      drft
   20.0026000000000
```

```
cfq1     cfqd
  1.50000000000000       1.00000000000000       1.00000000000000
  1.00000000000000
cfq2     cfqd
  3.00000000000000       2.00000000000000       1.00000000000000
  1.00000000000000
cfq3     cfqd
  1.50000000000000       3.00000000000000       1.00000000000000
  1.00000000000000
oct1     octm
 0.250000000000000     -0.152246981826680
oct2     octm
 0.250000000000000      0.181382120797860
oct3     octm
 0.250000000000000      0.121332566025083
oct4     octm
 0.250000000000000      1.118033689140618E-02
parm1    ps1
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -0.154895533047822      0.000000000000000E+00
parm2    ps2
-8.289453107758520E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00 -4.048969765226963E-02  0.000000000000000E+00
parm3    ps3
 8.630003538054180E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.362719077263124      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
trace14  rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
circ15   circ
 0.000000000000000E+00   15.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000       3.00000000000000
clear    iden
bip      bip
  20.0000000000000
tip      tip
 0.000000000000000E+00
bop      bop
  30.0000000000000
top      top
 0.000000000000000E+00
selparm  vary
  3.00000000000000       19.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
setucalc wsq
```

```
   2.00000000000000       3.00000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 fitaim   aim
   2.00000000000000       20.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 optaim   aim
   3.00000000000000       21.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   2.00000000000000       1.00000000000000
 fitvary  vary
   1.00000000000000       22.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 optvary  vary
   3.00000000000000       23.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   2.00000000000000       1.00000000000000
 fit      fit
   1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
  1.000000000000000E-04  0.000000000000000E+00   1.00000000000000
 mss      mss
  -1.00000000000000      1.000000000000000E-10  1.000000000000000E-04
  1.000000000000000E-04  0.100000000000000       1.00000000000000
 merit0   mrt0
   1.00000000000000       2.00000000000000
 wmerit0  wmrt
  0.000000000000000E+00   3.00000000000000
 merit2   mrt2
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 wmerit2  wmrt
   2.00000000000000       3.00000000000000
 end      end
#lines
 ctrip
     1*oct1        1*drvs        1*cq1         1*drvs        1*oct2     &
     1*drvs        1*cq2         1*drvs        1*oct3        1*drvs     &
     1*cq3         1*drvs        1*oct4
 cspot
     1*ctrip       1*drml
 cq1
     1*parm1       1*cfq1
 cq2
     1*parm2       1*cfq2
 cq3
     1*parm3       1*cfq3
 work
     1*setucalc    1*merit0      1*wmerit0
 setup
     1*selparm     1*fitaim      1*setucalc    1*optaim      1*work     &
     1*fitvary     1*optvary
#lumps
#loops
 onebyone
     1*cspot
#labor
```

```
    1*fileout
    1*cspot
    1*mapout
    1*setup
    1*bop
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*work
    1*mss
    1*top
    1*work
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*work
    1*clear
    1*cspot
    1*mapout
    1*fileout
    1*end

end of MARYLIE run
```

10.4 Sextupole Correction of a Solenoidal Spot Forming System

Consider the simple solenoid spot-forming system shown in figure 10.4.1. It may be viewed as the prototype for a scanning electron microscope, and consists of two solenoids with a beam crossover between them. The major geometric aberrations for such a system are those of third order arising from unavoidable solenoidal fringe-field effects. See section 6.23. The MaryLie examples of this section illustrate that the third-order aberrations detrimental to a spot-forming system can be removed by placing a properly powered sextupole corrector at the crossover.

Figure 10.4.1: A simple spot-forming system consisting of two solenoids and suitably chosen drifts.

10.4.1 Fitting Solenoid and Sextupole Strengths

The MaryLie run of Exhibit 10.4.1 illustrates how the two solenoid strengths are fit to achieve a spot-forming system with an intermediate beam crossover. This fitting is simplified by setting IOPT=1 for each solenoid so that the paraxial rotation parts of their respective maps are artificially removed. See section 6.23. In this way one does not have to worry about complications caused by coupling of the x and y plane motions. Once fitting of solenoid strengths has been achieved, IOPT for each solenoid is reset to IOPT=0 by use of a *rset* command. See section 9.18.

After the solenoid strengths have been fit, rays are traced through the system (with the sextupole off) to evaluate the effect of aberrations. Similar to the case of figure 2.2.1 of section 2.2, the incoming rays lie on two cylinders parallel to the optic axis. In this case the cylinders have radii of 50 and 100 $\times 10^{-6}$ meters, respectively. See figure 10.4.1. The result of this ray trace is shown in figure 10.4.1.1. While the radii of the incoming cylinders have a ratio of 2 to 1, the ratio of the two final circles shown in figure 10.4.1.1 is 8 to 1. This is what is to be expected from third-order geometrical aberrations since $2^3 = 8$.

Note that, for this and later parts of the run, the solenoid maps (whose computation requires numerical integration) are stored as the lumps *llens1* and *llens2* so that they do not

Figure 10.4.1.1: Final focal spot patterns produced by the simple (uncorrected) Solenoid Spot-Forming System of figure 10.4.1.

have to be continually recomputed. See section 5.9.

Subsequently the sextupole corrector strength is set to achieve the condition

$$f(140) = f(149) = f(195) = 0$$

in order to remove the aberrations detrimental to a spot-forming system. Rays are then traced through this system using the total transfer map to verify that correction has indeed occurred. The results of this ray trace are not shown, but are similar to those of figure 10.3.1.1: up to fitting and roundoff errors, all rays are now brought to a point focus. Finally, to estimate the effect of fourth and yet higher-order aberrations not computed by MaryLie 3.0 for the total transfer map, rays are traced element-by-element using the loop *onebyone* and the command circ18. The results of this ray trace are shown in figure 10.4.1.2. Comparison of figures 10.4.1.1 and 10.4.1.2 shows that the spot size has shrunk by about an order of magnitude as a result of sextupole correction.

Exhibits 10.4.1a and 10.4.1b show the contents of the external instruction files associated with the commands *aim1* and *vary1* that are used to fit the strength for the solenoid *sol1* in the line *sect1f*. A beam crossover is achieved by requiring the condition R(1,1) = 0 for the map associated with the line *sect1f*. Exhibits 10.4.1c and 10.4.1d show the instruction files for the commands *aim2* and *vary2* that are used to fit the strength for the solenoid *sol2* in the line *sect2f*. Note that imaging for the map associated with *sect2f* is achieved by requiring the condition R(1,2)=0 for this map. Exhibit 10.4.1e shows the instruction file for the reset command *rset*. As described earlier, it is invoked after the fitting of solenoid strengths is complete. Next, Exhibits 10.4.1f and 10.4.1g show the instruction files for the commands *aim3* and *vary3* that are used to set the strength of the sextupole corrector. Note that all the instructions provided by all these files could also have been entered interactively, if desired. Finally, Exhibit 10.4.1h show the MaryLie run itself along with explanatory remarks.

Figure 10.4.1.2: Estimated final focal spot patterns for sextupole-corrected system. Third order element-by-element calculation.

```
Exhibit 10.4.1a
Contents of file 021 that provides instructions for the command aim1:

 1    r(1,1) =      0.000000000E+00
 #


Exhibit 10.4.1b
Contents of file 022 that provides instructions for the command vary1:

 par1    4
  #


Exhibit 10.4.1c
Contents of file 023 that provides instructions for the command aim2:

 1    r(1,2) =      0.000000000E+00
 #


Exhibit 10.4.1d
Contents of file 024 that provides instructions for the command vary2:

 par2     4
  #


Exhibit 10.4.1e
Contents of file 027 that provides instructions for the command rset:

par1 6 par2 6
0
```

```
0
#


Exhibit 10.4.1f
Contents of file 025 that provides instructions for the command aim3:

 1    f(140) =      0.000000000E+00
 #


Exhibit 10.4.1g
Contents of file 026 that provides instructions for the command vary3:

 sex        2
  #


Exhibit 10.4.1h
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.4.1.
 This is a MARYLIE run for a sextupole corrected Solenoidal Spot-Forming
 System. The system consists of two solenoids, a sextupole, and suitably
 chosen drifts.  The beam parameters are those for 200 KeV electrons.

 This run does 8 things:

  a) The strength of the first solenoid is fit to produce a
     crossover (starting with parallel incoming rays) at the
     center of the location where the sextupole corrector is
     to be placed.  For simplicity, this fitting is carried
     out with the rotation part of the first solenoid map
     removed, and is equivalent to demanding that

     r(1,1)=0

     for the map corresponding to the line sect1f.

  b) The strength of the second solenoid is fit to image the
     crossover at a distance of .5 meter after the second
     solenoid. For simplicity, this fitting is carried
     out with the rotation part of the second solenoid map
     also removed, and is equivalent to demanding that

     r(1,2)=0
```

```
   for the map corresponding to the line sect2f.

c) After the strengths of the two solenoids have been set,
   the map for the combined system (sect1f sect2f) is
   computed to make sure that it is spot forming as it
   should be, i.e. it should satisfy

   r(1,1)=0.


d) As indicated, for ease of fitting the operations above
   were carried out with the rotation parts of the solenoidal
   maps removed.  After solenoidal fitting is complete, the
   6'th parameters in the parameter sets par1 and par2 are
   reset to IOPT=0 so that in subsequent use the full transfer
   maps for the solenoids will be computed.

e) The full transfer map is computed for the sextupole corrected
   spot-forming system with the sextupole turned off (replaced
   by an equivalent drift).  A ray trace is carried out using
   this map to show the detrimental effect of third-order
   spherical aberrations described by the generators

     f(140) = f(04 00 00) = -33.7677,
     f(149) = f(02 02 00) = -67.5354, and
     f(195) = f(00 04 00) = -33.7677.

   See the results below of printing out the generators for
   the full transfer map.  Note that, due to the rotational
   symmetry of solenoids, the generators appear in the
   combination

     (Px**2 + Py**2)**2.

   That is, there are the relations

     f(195) = f(140),
     f(149) = 2f(140).

f) The strength of the sextupole corrector corrector is fit
   to "zero out" the detrimental aberration generators.
   This is done by requiring that

     f(140) = 0.

   In so doing, the remaining aberration generators, f(149)
   and f(195),  are then also zeroed out automatically as
   a result of symmetry.

g) A ray trace is carried out using the map for the sextupole
   corrected system to indicate that the spot size has indeed
   diminished as a result of properly powering the corrector.
```

```
  h) An element by element ray trace is carried out to estimate
     the effect of residual fourth and higher order aberrations.

#beam
 1.649033824222788E-003
 0.391390152459380
  1.00000000000000
  1.00000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 sol1     sol
  0.000000000000000E+00  0.200000000000000       200.000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 sol2     sol
  0.000000000000000E+00  0.200000000000000       200.000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 par1     ps1
  5.000000000000000E-02  0.100000000000000      1.000000000000000E-02
  2.000000000000000E-02   3.00000000000000       1.00000000000000
 par2     ps1
  5.000000000000000E-02  0.100000000000000      1.000000000000000E-02
  2.000000000000000E-02   3.00000000000000       1.00000000000000
 dr1      drft
  0.000000000000000E+00
 dr2      drft
  0.100000000000000
 dr3      drft
  0.500000000000000
 dr4      drft
  0.500000000000000
 nosex    drft
  0.200000000000000
 nosex/2  drft
  0.100000000000000
 sex      sext
  0.200000000000000       100.000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 matout   ptm
   3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 bip      bip
   20.0000000000000
 tip      tip
  0.000000000000000E+00
 aim1     aim
   1.00000000000000       21.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 aim2     aim
   1.00000000000000       23.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
```

```
 aim3     aim
   1.00000000000000       25.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary1    vary
   1.00000000000000       22.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary2    vary
   1.00000000000000       24.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary3    vary
   1.00000000000000       26.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 rset     rset
   1.00000000000000       27.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fit      fit
   1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
  1.000000000000000E-03   1.00000000000000       1.00000000000000
 clear    iden
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 trace14  rt
  0.000000000000000E+00   14.0000000000000       4.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 trace16  rt
  0.000000000000000E+00   16.0000000000000       6.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 circ18   circ
  0.000000000000000E+00   18.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000       3.00000000000000
 end      end
#lines
 lens1
     1*par1        1*sol1
 lens2
     1*par2        1*sol1
 sect1
     1*dr1         1*llens1      1*dr2
 sect2
     1*dr3         1*llens2      1*dr4
 sect1f
     1*dr1         1*lens1       1*dr2         1*nosex/2
 sect2f
     1*nosex/2     1*dr3         1*lens2       1*dr4
 spot
     1*sect1       1*nosex       1*sect2
 cspot
     1*sect1       1*sex         1*sect2
#lumps
 llens1
     1*lens1
 llens2
```

```
     1*lens2
#loops
 onebyone
     1*cspot
#labor
    1*fileout
    1*sect1f
    1*aim1
    1*vary1
    1*bip
    1*clear
    1*sect1f
    1*fit
    1*tip
    1*matout
    1*clear
    1*sect2f
    1*aim2
    1*vary2
    1*bip
    1*clear
    1*sect2f
    1*fit
    1*tip
    1*matout
    1*clear
    1*sect1f
    1*sect2f
    1*matout
    1*rset
    1*clear
    1*spot
    1*mapout
    1*raysin
    1*trace14
    1*clear
    1*cspot
    1*mapout
    1*aim3
    1*vary3
    1*bip
    1*clear
    1*cspot
    1*fit
    1*tip
    1*mapout
    1*raysin
    1*trace16
    1*clear
    1*raysin
    1*onebyone
    1*circ18
    1*end
```

```
 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.000000000000000E-002
 iopt=           1

 symplectic violation =   1.554312234475219E-015

**********************
* Response to aim1:  *
**********************

accept  1:      r(1,1)

 Aims selected :
No.     item      present value
----------------------------------
 1    r(1,1) =      5.358248605E-02

**********************
* Response to vary1: *
**********************

No.  1 is par1     ps1     .  Parameter 4 out of 6 selected.
     par1(4) =  2.00000000000000E-02

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    par1       ps1          4      2.00000000000000E-02

****************************
* First fitting procedure: *
****************************

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.000000000000000E-002
 iopt=           1

 symplectic violation =   1.554312234475219E-015


 target           1 =  0.000000000000000E+000
Iter   1 Error= 5.3582E-02,  Step= 2.0000E-05,  SubErr= 5.3582E-02 @cut=    1
-------------------------------------------------------

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.002000000000000E-002
```

```
 iopt=
        1
 symplectic violation =   8.881784197001252E-016

*******************************************
* ETC. ETC. ETC.                          *
*******************************************

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.058806040588981E-002
 iopt=           1

 symplectic violation =   3.774758283725532E-015

Quit on iteration   6 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    r(1,1) =     -8.881784197E-16     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   par1       ps1        4      2.05880604058898E-02

 Maximum error is     8.881784E-16
 Maximum allowed was  1.000000E-10

*****************************************************************************
* Matrix part of map for sect1f showing that fitting has been accomplished: *
*****************************************************************************

matrix for map is :

-8.88178E-16  3.01323E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.31870E+00  6.63740E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -8.88178E-16  3.01323E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.31870E+00  6.63740E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.27366E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.000000000000000E-002
 iopt=           1

 symplectic violation =   1.554312234475219E-015
```

```
**********************
* Response to aim2:  *
**********************

accept  1:      r(1,2)

 Aims selected :
No.     item      present value
----------------------------------
 1    r(1,2) =     -2.210611452E-02

**********************
* Response to vary2: *
**********************

No.  1 is par2     ps1     .  Parameter 4 out of 6 selected.
     par2(4) =  2.00000000000000E-02

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    par2       ps1          4      2.00000000000000E-02

****************************
* Second fitting procedure: *
****************************

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.000000000000000E-002
 iopt=           1

 symplectic violation =   1.554312234475219E-015

 target           1 =  0.000000000000000E+000
Iter   1 Error= 2.2106E-02,  Step= 2.0000E-05,  SubErr= 2.2106E-02 @cut=    1
-------------------------------------------------------

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.002000000000000E-002
 iopt=           1

 symplectic violation =   8.881784197001252E-016

*********************************************
* ETC. ETC. ETC.                            *
*********************************************

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
```

```
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  1.982359647711195E-002
 iopt=           1

 symplectic violation =   1.776356839400250E-015
Quit on iteration   5 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
---------------------------------------------------
 1    r(1,2) =     -4.293787548E-12     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   par2       ps1        4      1.98235964771120E-02

 Maximum error is     4.293788E-12
 Maximum allowed was  1.000000E-10

*************************************************************************
* Matrix part of map for sect2f showing that fitting has been accomplished: *
*************************************************************************

matrix for map is :

-8.57392E-01 -4.29379E-12  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-3.08935E+00 -1.16633E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -8.57392E-01 -4.29379E-12  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -3.08935E+00 -1.16633E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.38894E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.058806040588981E-002
 iopt=           1

 symplectic violation =   3.774758283725532E-015

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  1.982359647711195E-002
 iopt=           1

 symplectic violation =   1.776356839400250E-015

*************************************************************************
* Matrix part of map for sect1f sect2f showing that is indeed spot forming; *
*************************************************************************
```

```
matrix for map is :

 1.42508E-11 -2.58352E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 3.87069E+00 -1.70503E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  1.42508E-11 -2.58352E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.87069E+00 -1.70503E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.81630E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

*********************
* Response to rset: *
*********************

MARYLIE #menu entries available to be varied:
-------------------------------------------------------------------------
aim1     circ18   dr3      fit      nosex/2  rset     tip      vary2
aim2     clear    dr4      mapout   par1     sex      trace14  vary3
aim3     dr1      end      matout   par2     sol1     trace16
bip      dr2      fileout  nosex    raysin   sol2     vary1

To RESET #menu items, enter name and (optional) parameter index.
Type * to relist, or a # sign when finished.

The following  2 #menu item(s) have been reset:
No.   Item       Type     Parameter   New value
-------------------------------------------------------------
 1    par1       ps1          6      0.00000000000000E+00
 2    par2       ps1          6      0.00000000000000E+00

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.058806040588981E-002
 iopt=           0

 symplectic violation =   1.457667320181599E-01

lump llens1   constructed and stored.( 1)

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  1.982359647711195E-002
 iopt=           0

 symplectic violation =   1.325245468919434E-012

lump llens2   constructed and stored.( 2)

*********************************************************
* Map for spot forming system with sextupole turned off: *
*********************************************************
```

```
matrix for map is :

 3.92752E-12 -8.74925E-02  1.36455E-11 -2.43086E-01  0.00000E+00  0.00000E+00
 1.31084E+00 -5.77420E-01  3.64197E+00 -1.60428E+00  0.00000E+00  0.00000E+00
-1.36451E-11  2.43086E-01  3.92752E-12 -8.74925E-02  0.00000E+00  0.00000E+00
-3.64197E+00  1.60428E+00  1.31084E+00 -5.77420E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.81630E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )= -19.354345343837
 f( 38)=f( 11 00 01 )=  8.5317869933313
 f( 42)=f( 10 10 01 )= 1.77635683940025E-15
 f( 45)=f( 10 01 01 )=  1.7622352560429
 f( 53)=f( 02 00 01 )= -1.8284384427366
 f( 57)=f( 01 10 01 )= -1.7622352560429
 f( 60)=f( 01 01 01 )= 3.19189119579733E-16
 f( 67)=f( 00 20 01 )= -19.354345343837
 f( 70)=f( 00 11 01 )=  8.5317869933313
 f( 76)=f( 00 02 01 )= -1.8284384427366
 f( 83)=f( 00 00 03 )= -1.3061027823091
 f( 84)=f( 40 00 00 )= -5368.1678868663
 f( 85)=f( 31 00 00 )=  3124.3698789490
 f( 86)=f( 30 10 00 )= 1.72228897810101E-11
 f( 87)=f( 30 01 00 )=  75.164310605427
 f( 90)=f( 22 00 00 )= -1010.3164650898
 f( 91)=f( 21 10 00 )= -75.164310605444
 f( 92)=f( 21 01 00 )= -30.445062048227
 f( 95)=f( 20 20 00 )= -10736.335773733
 f( 96)=f( 20 11 00 )=  3124.3698789490
 f( 99)=f( 20 02 00 )= -341.27794387998
 f(104)=f( 20 00 02 )= -88.367907544732
 f(105)=f( 13 00 00 )=  255.03547488004
 f(106)=f( 12 10 00 )=  30.445062048233
 f(107)=f( 12 01 00 )=  5.5596519517141
 f(110)=f( 11 20 00 )=  3124.3698789490
 f(111)=f( 11 11 00 )= -1338.0770424196
 f(114)=f( 11 02 00 )=  255.03547488004
 f(119)=f( 11 00 02 )=  62.904770756084
 f(120)=f( 10 30 00 )= 7.26174675946822E-12
 f(121)=f( 10 21 00 )=  75.164310605425
 f(124)=f( 10 12 00 )= -30.445062048228
 f(129)=f( 10 10 02 )= 5.49862591908003E-14
 f(130)=f( 10 03 00 )=  5.5596519517147
 f(135)=f( 10 01 02 )=  3.1890117279099
 f(140)=f( 04 00 00 )= -33.767743155594
 f(141)=f( 03 10 00 )= -5.5596519517151
 f(142)=f( 03 01 00 )= 3.83495318834193E-14
 f(145)=f( 02 20 00 )= -341.27794387998
 f(146)=f( 02 11 00 )=  255.03547488004
 f(149)=f( 02 02 00 )= -67.535486311188
```

```
 f(154)=f( 02 00 02 )= -10.294678284563
 f(155)=f( 01 30 00 )= -75.164310605433
 f(156)=f( 01 21 00 )=  30.445062048232
 f(159)=f( 01 12 00 )= -5.5596519517156
 f(164)=f( 01 10 02 )= -3.1890117279099
 f(165)=f( 01 03 00 )= 1.42049566553837E-13
 f(170)=f( 01 01 02 )= 9.90874049477952E-15
 f(175)=f( 00 40 00 )= -5368.1678868663
 f(176)=f( 00 31 00 )=  3124.3698789490
 f(179)=f( 00 22 00 )= -1010.3164650898
 f(184)=f( 00 20 02 )= -88.367907544732
 f(185)=f( 00 13 00 )=  255.03547488004
 f(190)=f( 00 11 02 )=  62.904770756084
 f(195)=f( 00 04 00 )= -33.767743155594
 f(200)=f( 00 02 02 )= -10.294678284563
 f(209)=f( 00 00 04 )= -2.1210054140684

*****************************************************************
* Ray trace for spot forming system with sextupole turned off: *
*****************************************************************

  200 ray(s) read in from file  13

******************************************************************
* Map for spot forming system with sextupole partially powered: *
******************************************************************

matrix for map is :

 3.92752E-12 -8.74925E-02  1.36455E-11 -2.43086E-01  0.00000E+00  0.00000E+00
 1.31084E+00 -5.77420E-01  3.64197E+00 -1.60428E+00  0.00000E+00  0.00000E+00
-1.36451E-11  2.43086E-01  3.92752E-12 -8.74925E-02  0.00000E+00  0.00000E+00
-3.64197E+00  1.60428E+00  1.31084E+00 -5.77420E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.81630E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= -1580.8951974218
 f( 29)=f( 21 00 00 )=  57.529446412786
 f( 30)=f( 20 10 00 )= -20039.007779464
 f( 31)=f( 20 01 00 )=  243.07599601673
 f( 33)=f( 20 00 01 )= -19.354345343837
 f( 34)=f( 12 00 00 )= -7.9831068587096
 f( 35)=f( 11 10 00 )=  486.15199203346
 f( 36)=f( 11 01 00 )= -67.461161965202
 f( 38)=f( 11 00 01 )=  8.5317869933313
 f( 39)=f( 10 20 00 )=  4742.6855922654
 f( 40)=f( 10 11 00 )= -115.05889282557
 f( 42)=f( 10 10 01 )= 1.77635683940025E-15
 f( 43)=f( 10 02 00 )=  7.9831068587118
 f( 45)=f( 10 01 01 )=  1.7622352560429
 f( 49)=f( 03 00 00 )= 3.52997631125618E-11
```

```
f( 50)=f( 02 10 00 )= -33.730580982600
f( 51)=f( 02 01 00 )= 3.67890606867149E-10
f( 53)=f( 02 00 01 )= -1.8284384427366
f( 54)=f( 01 20 00 )= -57.529446412787
f( 55)=f( 01 11 00 )=  15.966213717424
f( 57)=f( 01 10 01 )= -1.7622352560429
f( 58)=f( 01 02 00 )=-1.07888808997814E-10
f( 60)=f( 01 01 01 )= 3.19189119579733E-16
f( 64)=f( 00 30 00 )=  6679.6692598213
f( 65)=f( 00 21 00 )= -243.07599601673
f( 67)=f( 00 20 01 )= -19.354345343837
f( 68)=f( 00 12 00 )=  33.730580982599
f( 70)=f( 00 11 01 )=  8.5317869933313
f( 74)=f( 00 03 00 )=-1.22327037388459E-10
f( 76)=f( 00 02 01 )= -1.8284384427366
f( 83)=f( 00 00 03 )= -1.3061027823091
f( 84)=f( 40 00 00 )=  4534434.9355423
f( 85)=f( 31 00 00 )= -417.18711658195
f( 86)=f( 30 10 00 )=-1.58325412780158E-08
f( 87)=f( 30 01 00 )=  75.164314169984
f( 89)=f( 30 00 01 )= -4135.3649240237
f( 90)=f( 22 00 00 )=  464.02211364079
f( 91)=f( 21 10 00 )= -75.164314147981
f( 92)=f( 21 01 00 )= -30.445062007087
f( 94)=f( 21 00 01 )=  6096.0173475002
f( 95)=f( 20 20 00 )=  9068869.8710847
f( 96)=f( 20 11 00 )= -417.18711659685
f( 98)=f( 20 10 01 )= -56889.773039091
f( 99)=f( 20 02 00 )= -39075.431813770
f(101)=f( 20 01 01 )=  25811.399464434
f(104)=f( 20 00 02 )= -88.367907544732
f(105)=f( 13 00 00 )= -17.747784419917
f(106)=f( 12 10 00 )=  30.445061995271
f(107)=f( 12 01 00 )=  5.5596519387338
f(109)=f( 12 00 01 )= -155.63630755806
f(110)=f( 11 20 00 )= -417.18711658567
f(111)=f( 11 11 00 )=  79078.907854842
f(113)=f( 11 10 01 )=  51622.798928869
f(114)=f( 11 02 00 )= -17.747784425039
f(116)=f( 11 01 01 )= -1330.2545443094
f(119)=f( 11 00 02 )=  62.904770756084
f(120)=f( 10 30 00 )= 7.45056638606911E-09
f(121)=f( 10 21 00 )=  75.164314153104
f(123)=f( 10 20 01 )=  12406.094772071
f(124)=f( 10 12 00 )= -30.445061992942
f(126)=f( 10 11 01 )= -12192.034695000
f(129)=f( 10 10 02 )= 5.49862591908003E-14
f(130)=f( 10 03 00 )=  5.5596519354159
f(132)=f( 10 02 01 )=  155.63630755804
f(135)=f( 10 01 02 )=  3.1890117279099
f(140)=f( 04 00 00 )= -14.841279885615
f(141)=f( 03 10 00 )= -5.5596519367838
f(142)=f( 03 01 00 )= 9.82282501865717E-11
```

```
 f(144)=f( 03 00 01 )=  9.7710840812635
 f(145)=f( 02 20 00 )= -39075.431813776
 f(146)=f( 02 11 00 )= -17.747784422245
 f(148)=f( 02 10 01 )= -665.12727215468
 f(149)=f( 02 02 00 )= -29.682559770183
 f(151)=f( 02 01 01 )=  123.85566749550
 f(154)=f( 02 00 02 )= -10.294678284563
 f(155)=f( 01 30 00 )= -75.164314166142
  f(156)=f( 01 21 00 )=  30.445062003885
 f(158)=f( 01 20 01 )= -6096.0173475002
 f(159)=f( 01 12 00 )= -5.5596519411203
 f(161)=f( 01 11 01 )=  311.27261511610
 f(164)=f( 01 10 02 )= -3.1890117279099
 f(165)=f( 01 03 00 )= 1.28056848111568E-09
 f(167)=f( 01 02 01 )= -29.313252243794
 f(170)=f( 01 01 02 )= 9.90874049477952E-15
 f(175)=f( 00 40 00 )=  4534434.9355423
 f(176)=f( 00 31 00 )= -417.18711657636
 f(178)=f( 00 30 01 )=  18963.257679697
 f(179)=f( 00 22 00 )=  464.02211363707
 f(181)=f( 00 21 01 )= -25811.399464434
 f(184)=f( 00 20 02 )= -88.367907544732
 f(185)=f( 00 13 00 )= -17.747784420382
 f(187)=f( 00 12 01 )=  665.12727215469
 f(190)=f( 00 11 02 )=  62.904770756084
 f(195)=f( 00 04 00 )= -14.841279885324
 f(197)=f( 00 03 01 )= -41.285222498500
 f(200)=f( 00 02 02 )= -10.294678284563
 f(209)=f( 00 00 04 )= -2.1210054140684

*********************
* Response to aim3: *
*********************

accept  1:      f(140)

 Aims selected :
No.     item      present value
----------------------------------
 1    f(140) =      -14.8412799

*********************
* Response to vary3: *
*********************

No.  1 is sex      sext    .  Parameter 2 out of 2 selected.
      sex(2) =   100.00000000000

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
------------------------------------------------------------
 1    sex        sext         2       100.00000000000
```

```
*****************************************
* Beginning of third fitting procedure: *
*****************************************

 target           1 =  0.000000000000000E+000
Iter   1 Error= 1.4841E+01,  Step= 1.0000E-01,  SubErr= 1.4841E+01 @cut=    1
-------------------------------------------------------

*********************************************
* ETC. ETC. ETC.                            *
*********************************************

Quit on iteration   8 for reason 2: Best effort: Hit bottom at full reach
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f(140) =     -1.222360879E-09     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   sex        sext       2       133.57226325988

 Maximum error is     1.222361E-09
 Maximum allowed was  1.000000E-10

**********************************************
* Map for fully fitted and corrected system: *
**********************************************

matrix for map is :

 3.92752E-12 -8.74925E-02  1.36455E-11 -2.43086E-01  0.00000E+00  0.00000E+00
 1.31084E+00 -5.77420E-01  3.64197E+00 -1.60428E+00  0.00000E+00  0.00000E+00
-1.36451E-11  2.43086E-01  3.92752E-12 -8.74925E-02  0.00000E+00  0.00000E+00
-3.64197E+00  1.60428E+00  1.31084E+00 -5.77420E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.81630E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= -2111.6374949631
 f( 29)=f( 21 00 00 )=  76.843383614438
 f( 30)=f( 20 10 00 )= -26766.556225854
 f( 31)=f( 20 01 00 )=  324.68210932105
 f( 33)=f( 20 00 01 )= -19.354345343837
 f( 34)=f( 12 00 00 )= -10.663216509634
 f( 35)=f( 11 10 00 )=  649.36421864209
 f( 36)=f( 11 01 00 )= -90.109400858331
 f( 38)=f( 11 00 01 )=  8.5317869933313
 f( 39)=f( 10 20 00 )=  6334.9124848893
```

```
f( 40)=f( 10 11 00 )= -153.68676722888
f( 42)=f( 10 10 01 )= 1.77635683940025E-15
f( 43)=f( 10 02 00 )=  10.663216509637
f( 45)=f( 10 01 01 )=  1.7622352560429
f( 49)=f( 03 00 00 )= 4.74926764582051E-11
f( 50)=f( 02 10 00 )= -45.054700429167
f( 51)=f( 02 01 00 )= 4.90217644255608E-10
f( 53)=f( 02 00 01 )= -1.8284384427366
f( 54)=f( 01 20 00 )= -76.843383614437
f( 55)=f( 01 11 00 )=  21.326433019271
f( 57)=f( 01 10 01 )= -1.7622352560429
f( 58)=f( 01 02 00 )=-1.44609657581896E-10
f( 60)=f( 01 01 01 )= 3.19189119579733E-16
f( 64)=f( 00 30 00 )=  8922.1854086180
f( 65)=f( 00 21 00 )= -324.68210932103
f( 67)=f( 00 20 01 )= -19.354345343837
f( 68)=f( 00 12 00 )=  45.054700429161
f( 70)=f( 00 11 01 )=  8.5317869933313
f( 74)=f( 00 03 00 )=-1.62799551617354E-10
f( 76)=f( 00 02 01 )= -1.8284384427366
f( 83)=f( 00 00 03 )= -1.3061027823091
f( 84)=f( 40 00 00 )=  8094344.0167363
f( 85)=f( 31 00 00 )= -3194.3165697586
f( 86)=f( 30 10 00 )= 2.98022648781426E-08
f( 87)=f( 30 01 00 )=  75.164316937991
f( 89)=f( 30 00 01 )= -5523.7005230738
f( 90)=f( 22 00 00 )=  1620.1320099849
f( 91)=f( 21 10 00 )= -75.164316997363
f( 92)=f( 21 01 00 )= -30.445061940206
f( 94)=f( 21 00 01 )=  8142.5883397711
f( 95)=f( 20 20 00 )=  16188688.033473
f( 96)=f( 20 11 00 )= -3194.3165697176
f( 98)=f( 20 10 01 )= -75988.957411725
f( 99)=f( 20 02 00 )= -69449.010352819
f(101)=f( 20 01 01 )=  34476.870443694
f(104)=f( 20 00 02 )= -88.367907544732
f(105)=f( 13 00 00 )= -231.65212782705
f(106)=f( 12 10 00 )=  30.445061980602
f(107)=f( 12 01 00 )=  5.5596519154216
f(109)=f( 12 00 01 )= -207.88693845939
f(110)=f( 11 20 00 )= -3194.3165697195
f(111)=f( 11 11 00 )=  142138.28472555
f(113)=f( 11 10 01 )=  68953.740887388
f(114)=f( 11 02 00 )= -231.65212781727
f(116)=f( 11 01 01 )= -1776.8511019515
f(119)=f( 11 00 02 )=  62.904770756084
f(120)=f( 10 30 00 )=-2.98023365985500E-08
f(121)=f( 10 21 00 )=  75.164316997363
f(123)=f( 10 20 01 )=  16571.101569221
f(124)=f( 10 12 00 )= -30.445061979904
f(126)=f( 10 11 01 )= -16285.176679542
f(129)=f( 10 10 02 )= 5.49862591908003E-14
f(130)=f( 10 03 00 )=  5.5596519271504
```

```
 f(132)=f( 10 02 01 )=  207.88693845939
 f(135)=f( 10 01 02 )=  3.1890117279099
 f(140)=f( 04 00 00 )=-1.22236087918282E-09
 f(141)=f( 03 10 00 )= -5.5596519279799
 f(142)=f( 03 01 00 )= 1.46610828165306E-09
 f(144)=f( 03 00 01 )=  13.051458152368
 f(145)=f( 02 20 00 )= -69449.010352816
 f(146)=f( 02 11 00 )= -231.65212781914
 f(148)=f( 02 10 01 )= -888.42555097574
 f(149)=f( 02 02 00 )=-5.23868948221207E-09
 f(151)=f( 02 01 01 )=  165.43681824937
 f(154)=f( 02 00 02 )= -10.294678284563
 f(155)=f( 01 30 00 )= -75.164316935896
 f(156)=f( 01 21 00 )=  30.445061932406
 f(158)=f( 01 20 01 )= -8142.5883397711
 f(159)=f( 01 12 00 )= -5.5596519116672
 f(161)=f( 01 11 01 )=  415.77387691878
 f(164)=f( 01 10 02 )= -3.1890117279099
 f(165)=f( 01 03 00 )=-1.65891839501442E-09
 f(167)=f( 01 02 01 )= -39.154374457109
 f(170)=f( 01 01 02 )= 9.90874049477952E-15
 f(175)=f( 00 40 00 )=  8094344.0167363
 f(176)=f( 00 31 00 )= -3194.3165697567
 f(178)=f( 00 30 01 )=  25329.652470575
 f(179)=f( 00 22 00 )=  1620.1320099793
 f(181)=f( 00 21 01 )= -34476.870443694
 f(184)=f( 00 20 02 )= -88.367907544732
 f(185)=f( 00 13 00 )= -231.65212782333
 f(187)=f( 00 12 01 )=  888.42555097573
 f(190)=f( 00 11 02 )=  62.904770756084
 f(195)=f( 00 04 00 )=-2.38651409745216E-09
 f(197)=f( 00 03 01 )= -55.145606083124
 f(200)=f( 00 02 02 )= -10.294678284563
 f(209)=f( 00 00 04 )= -2.1210054140684

****************************************
* Ray trace for fully corrected system: *
****************************************

  200 ray(s) read in from file  13

************************************************************
* Element by element ray trace for fully corrected system: *
************************************************************

  200 ray(s) read in from file  13

circulating through onebyone :
 dr1      llens1   dr2      sex      dr3
 llens2   dr4

end of MARYLIE run
```

10.4.2 Scanning the Sextupole Strength

The results of Exhibit 10.4.1h show that the third-order spherical aberration vanishes when the sextupole corrector has the strength S=133.57 Tesla/(meter)^2. One might wonder how the spherical aberration, as measured say by the value of f(140), varies as the sextupole strength is varied. Questions of this sort can be addressed by use of the *scan* and *wsq* commands within a logical loop. See sections 9.14 and 8.27. The answer to our question in this case is given by the graph in figure 10.4.2.1. It shows that the spherical aberration is an even *quadratic* function of S that crosses zero at S = ± 133.57.

Figure 10.4.2.1 was prepared by plotting the file shown in Exhibit 10.4.2a. This file was written by the *wsq* command in the MaryLie run shown in the remaining parts of Exhibit 10.4.2. In particular, Exhibits 10.4.2b and 10.4.2c show the instruction files for the commands *svary* and *sq* that are issued prior to entering the logical loop. See sections 9.6 and 8.26. Finally, Exhibit 10.4.2d shows the MaryLie run itself. Scanning the parameter value for the sextupole strength and writing out the value of the parameter and the corresponding value of f(140) on file 10 are carried out in *#labor* by the commands in the logical loop listed below:

```
bip
scan
clear
cspot
wsq
tip
```

Figure 10.4.2.1: Spherical aberration of the Solenoidal Spot-Forming System as a function of the sextupole corrector strength.

```
Exhibit 10.4.2a
Contents of file 10 resulting from use of the scan and wsq commands.
The first column is the sextupole strength S, and the second is f(140)
for the total transfer map for the Solenoidal Spot-Forming System.

-1.50000E+02  8.81680E+00
-1.40000E+02  3.32812E+00
-1.30000E+02 -1.78202E+00
-1.20000E+02 -6.51364E+00
-1.10000E+02 -1.08667E+01
-1.00000E+02 -1.48413E+01
-9.00000E+01 -1.84373E+01
-8.00000E+01 -2.16548E+01
-7.00000E+01 -2.44938E+01
-6.00000E+01 -2.69542E+01
-5.00000E+01 -2.90361E+01
-4.00000E+01 -3.07395E+01
-3.00000E+01 -3.20644E+01
-2.00000E+01 -3.30107E+01
-1.00000E+01 -3.35785E+01
 0.00000E+00 -3.37677E+01
 1.00000E+01 -3.35785E+01
 2.00000E+01 -3.30107E+01
 3.00000E+01 -3.20644E+01
 4.00000E+01 -3.07395E+01
 5.00000E+01 -2.90361E+01
 6.00000E+01 -2.69542E+01
 7.00000E+01 -2.44938E+01
 8.00000E+01 -2.16548E+01
 9.00000E+01 -1.84373E+01
 1.00000E+02 -1.48413E+01
 1.10000E+02 -1.08667E+01
 1.20000E+02 -6.51364E+00
 1.30000E+02 -1.78202E+00
 1.40000E+02  3.32812E+00
 1.50000E+02  8.81680E+00


Exhibit 10.4.2b
Contents of file 19 that provides
instructions for the command svary:

 sex        2
  #


Exhibit 10.4.2c
Contents of file 21 that provides
instructions for the command sq:

 1    f(140)
 #
```

```
Exhibit 10.4.2d
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.4.2.
 This is a MARYLIE run for a sextupole corrected Solenoidal Spot-Forming
 System. The system consists of two solenoids, a sextupole, and suitably
 chosen drifts.  The beam parameters are those for 200 KeV electrons.

 This run illustrates, with the aid of scan and wsq commands, how
 the offensive spherical aberration generator f(140) depends on the
 strength of the sextupole corrector.  The fit solenoid strengths
 are those of Exhibit 10.4.1.  Again, due to the rotational
 symmetry of solenoids, the offensive generators appear in the
 combination

       (Px**2 + Py**2)**2.

 That is, there are the relations

       f(195) = f(140),
       f(149) = 2f(140),

 so that it is sufficient to examine only the f(140) generator.

#beam
 1.649033824222788E-003
 0.391390152459380
  1.00000000000000
  1.00000000000000
#menu
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 sol1     sol
  0.000000000000000E+00  0.200000000000000       200.000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 sol2     sol
  0.000000000000000E+00  0.200000000000000       200.000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 par1     ps1
  5.000000000000000E-02  0.100000000000000      1.000000000000000E-02
  2.058806040588981E-02   3.00000000000000      0.000000000000000E+00
 par2     ps1
  5.000000000000000E-02  0.100000000000000      1.000000000000000E-02
  1.982359647711195E-02   3.00000000000000      0.000000000000000E+00
 dr1      drft
  0.000000000000000E+00
 dr2      drft
```

```
  0.100000000000000
 dr3      drft
  0.500000000000000
 dr4      drft
  0.500000000000000
 sex      sext
  0.200000000000000      0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 clear    iden
 bip      bip
   200.000000000000
 tip      tip
  0.000000000000000E+00
 svary    vary
   3.00000000000000       19.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 sq       sq
   21.0000000000000      0.000000000000000E+00   1.00000000000000
   1.00000000000000
 scan     scan
   1.00000000000000      -15.0000000000000      0.000000000000000E+00
   10.0000000000000      0.000000000000000E+00   1.00000000000000
 wsq      wsq
   3.00000000000000       1.00000000000000       10.0000000000000
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 end      end
#lines
 lens1
     1*par1        1*sol1
 lens2
     1*par2        1*sol1
 sect1
     1*dr1         1*llens1      1*dr2
 sect2
     1*dr3         1*llens2      1*dr4
 cspot
     1*sect1       1*sex         1*sect2
#lumps
 llens1
     1*lens1
 llens2
     1*lens2
#loops
#labor
    1*fileout
    1*cspot
    1*svary
    1*sq
    1*bip
    1*scan
    1*clear
```

```
    1*cspot
    1*wsq
    1*tip
    1*end

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  2.058806040588981E-002
 iopt=           0

 symplectic violation =   1.457667320181599E-012
lump llens1   constructed and stored.( 1)

 zi=  0.000000000000000E+000 zf=  0.200000000000000
 ns=         200
 di=  5.000000000000000E-002 length=  0.100000000000000
 cl=  1.000000000000000E-002 B=  1.982359647711195E-002
 iopt=           0

 symplectic violation =   1.322331133479793E-012
lump llens2   constructed and stored.( 2)

**********************************
* Response to the command svary: *
**********************************

No.  1 is sex      sext    .  Parameter 2 out of 2 selected.
      sex(2) =  0.00000000000000E+00

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    sex        sext         2      0.00000000000000E+00


*******************************
* Response to the command sq: *
*******************************

 In subroutine sq
accept  1:      f(140)

 Aims selected :
No.     item      present value
-----------------------------------
 1    f(140) =      -33.7677432

*************************************************************
* Response to scan command in the logical loop in #labor: *
*************************************************************

** Scan Step   1 xx, yy =  -150.0000      0.0000000E+00
```

```
** Scan Step   2 xx, yy =  -140.0000      0.0000000E+00
** Scan Step   3 xx, yy =  -130.0000      0.0000000E+00
** Scan Step   4 xx, yy =  -120.0000      0.0000000E+00
** Scan Step   5 xx, yy =  -110.0000      0.0000000E+00
** Scan Step   6 xx, yy =  -100.0000      0.0000000E+00
** Scan Step   7 xx, yy =  -90.00000      0.0000000E+00
** Scan Step   8 xx, yy =  -80.00000      0.0000000E+00
** Scan Step   9 xx, yy =  -70.00000      0.0000000E+00
** Scan Step  10 xx, yy =  -60.00000      0.0000000E+00
** Scan Step  11 xx, yy =  -50.00000      0.0000000E+00
** Scan Step  12 xx, yy =  -40.00000      0.0000000E+00
** Scan Step  13 xx, yy =  -30.00000      0.0000000E+00
** Scan Step  14 xx, yy =  -20.00000      0.0000000E+00
** Scan Step  15 xx, yy =  -10.00000      0.0000000E+00
** Scan Step  16 xx, yy =  0.0000000E+00  0.0000000E+00
** Scan Step  17 xx, yy =   10.00000      0.0000000E+00
** Scan Step  18 xx, yy =   20.00000      0.0000000E+00
** Scan Step  19 xx, yy =   30.00000      0.0000000E+00
** Scan Step  20 xx, yy =   40.00000      0.0000000E+00
** Scan Step  21 xx, yy =   50.00000      0.0000000E+00
** Scan Step  22 xx, yy =   60.00000      0.0000000E+00
** Scan Step  23 xx, yy =   70.00000      0.0000000E+00
** Scan Step  24 xx, yy =   80.00000      0.0000000E+00
** Scan Step  25 xx, yy =   90.00000      0.0000000E+00
** Scan Step  26 xx, yy =   100.0000      0.0000000E+00
** Scan Step  27 xx, yy =   110.0000      0.0000000E+00
** Scan Step  28 xx, yy =   120.0000      0.0000000E+00
** Scan Step  29 xx, yy =   130.0000      0.0000000E+00
** Scan Step  30 xx, yy =   140.0000      0.0000000E+00
** Scan Step  31 xx, yy =   150.0000      0.0000000E+00
  Scan loop is finished

end of MARYLIE run
```

10.5 Fitting at Multiple Lattice Locations

The fitting/optimizing examples described so far in this chapter have involved quantities *all* calculated at the *same* location within a lattice. Such fitting/optimizing is relatively straight forward because essentially all the values of quantities computed by MaryLie in any "single instance" are automatically set aside for possible fitting/optimization or other possible study. However, fitting/optimization or other possible study of "multipole instances" of the same quantity requires a different approach. In this case, one must arrange to store the multiple values of the multiple instances of any given quantity until all the values required for any desired operation (fitting/optimization or other possible study) have been accumulated.

Consider, for example, the problem of fitting the horizontal beta functions (*plural*) at the centers of *both* the F and D quads in a simple FODO cell. In this problem we have two instances of the same quantity, namely the horizontal beta function, both of which must be computed and stored during the course of a fitting operation. Exhibit 10.5.1 shows how this may be done. Once this example has been understood, it will be evident how to fit/optimize,

or otherwise study, any number of occurrences of any number of quantities.

The general nature of the computation is explained in the *#comments* portion of the MaryLie run. The key feature is that the two values of the horizontal beta functions at the centers of the two quads are accumulated by putting them in locations 1 and 2 of the array *ucalc*. This is accomplished by use of the commands *sq*, *wsq1*, and *wsq2*. These two values are then looked at by *fit* thanks to the command *aim*.

Exhibit 10.5.1a shows the instructions for the *sq* command that selects the horizontal beta function as the quantity of interest. Note from the *#menu* component of the MaryLie run, see Exhibit 10.5.1d, that *sq* uses the *auxiliary* storage (ISTORE = 3). See section 8.26. Correspondingly *wsq1* and *wsq2*, which are meant to act in concert with *sq*, also use ISTORE = 3. The command *wsq1* has JOB=1 and IFILE=−1, and consequently writes the quantity selected by *sq* in the *first* location of the *ucalc* array. The command *wsq2* has JOB=1 and IFILE=−2. Therefore it writes the quantity selected by *sq* in the *second* location of the *ucalc* array. See section 8.27.

Exhibit 10.5.1b shows the instructions for the *aim* command. They specify that *fit* should try to achieve the conditions

u(1) = ucalc(1) = value of horizontal beta function in the center of the F quad = 11,

u(2) = ucalc(2) = value of horizontal beta function in the center of the D quad = 5.

Exhibit 10.5.1c shows the instructions for the *vary* command. They specify that *fit* should vary the strengths of the leading half-quads *inhfq* and *inhdq*. Also, the strengths of the trailing half-quads, *outhfq* and *outhdq*, should be tied to their leading counterparts. Note that both *aim* and *vary* use ISTORE=1 while (as described earlier) *sq*, *wsq1*, and *wsq2* use ISTORE=3. Indeed, it is necessary that *sq*, *wsq1*, and *wsq2* use a value of ISTORE different from ISTORE=1 since MaryLie treats the commands *aim* and *sq* as variants of each other. Therefore, they must use different storage (different values of ISTORE) in order not to conflict.

Exhibit 10.5.1d shows the MaryLie run itself along with explanatory comments. Note that, because MaryLie permits the use of nested lines that may contain both elements and commands, it is possible to carry out quite complicated operations while still keeping *#labor* quite succinct.

As a last comment, suppose one wanted instead to fit the horizontal beta function in the center of the F quad and the *vertical* beta function in the center of the D quad. In this case the *sq* command should be changed to select both the horizontal and vertical beta functions as illustrated below:

```
Contents of file 15 that provides instructions for the command sq
so that both the horizontal and vertical beta functions are selected.

1        bx
2        by
#
```

The *wsq1* command would be unchanged, but its effect would now be to place, for the center of the F quad, the value of the horizontal beta function in u(1) and the value of the vertical beta function in u(2). The command *wsq2*, which could now better be called *wsq3*, should be changed to have IFILE=−3. In this case its effect would be to place, for the center of the D quad, the value of the horizontal beta function in u(3) and the value of the vertical beta function in u(4). Correspondingly, in order to specify that *fit* should look at the value of the horizontal beta function in the center of the F quad and the value of the vertical beta function in the center of the D quad, *aim* should be altered to specify desired values for u(1) and u(4). Finally, if desired, the command *wuca* could be used to check that the proper values were indeed being placed in the *ucalc* array. See section 7.41.

```
Exhibit 10.5.1a
Contents of file 15 that provides instructions for the command sq:

 1        bx
 #


Exhibit 10.5.1b
Contents of file 17 that provides instructions for the command aim:

 1    u(  1) =       11.0000000
 2    u(  2) =       5.00000000
 #


Exhibit 10.5.1c
Contents of file 19 that provides instructions for the command vary:

 inhfq      2
 inhdq      2
 outhfq     2
          1   1.00000000000000
 outhdq     2
          2   1.00000000000000
  #



Exhibit 10.5.1d
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.5.1.
      This is an example of fitting quantities at two different locations
 in a lattice.  The lattice is a simple FODO cell, and the horizontal
 beta functions are fitted at the centers of the quads to achieve the aim
```

```
  horizontal beta function in center of F quad = 11,

  horizontal beta function in center of D quad = 5.

The drifts in the FODO cell are each 3 meters long, and the quads are
each .5 meters long.  The quads are split into "in" and "out" halves,

  F quad = hfq = inhfq outhfq,

  D quad = hdq = inhdq outhdq.

The beam parameters are those for 800 Mev protons.

     The computation of lattice functions at various locations in a
cell requires some explanation.  To compute lattice functions at the
center of the F quad it is necessary to compute the transfer map
from the center of the F quad back to itself,

  map from center of F quad and back =  outhfq dr3 inhdq outhdq dr3 inhfq.

Similarly, to compute lattice functions at the center of the D quad it
is necessary to compute the transfer map from the center of the D quad
back to itself,

  map from center of D quad and back =  outhdq dr3 inhfq outhfq dr3 inhdq.

In each case, once the required map has been computed, the linear lattice
functions can be found using a tasm command.

     The computation of these two maps can be organized as follows:
Let cell be the map for the FODO cell,

  cell = hfq dr3 hdq dr3 = inhfq outhfq dr3 inhdq outhdq dr3.

Evidently the maps from the centers of the F and D quads and back
can be written in the form

  map from center of F quad and back =  (inhfq)**-1 cell (inhfq),

  map from center of D quad and back =

      (inhfq outhfq dr3 inhdq)**-1 cell (inhfq outhfq dr3 inhdq).

Thus, in each case, it is only necessary to carry out the following steps:

  a) Accumulate (concatenate) the maps up to the observation point.

  b) Store the accumulated map.

  c) Invert the accumulated map.

  d) Concatenate the resulting map with the map for cell (which itself
     needs to be computed only once beforehand and stored for repeated use).
```

```
   e) Concatenate that result with the stored accumulated map.

     Examination of the #lines component below shows that the line "work"
 carries out the operations b through e above as well as calling tasm.
 Finally, it again makes the current map the accumulated map up to that point.
 The map for cell itself is computed and stored by invoking the line
 "setup".  The reader is now prepared to understand the contents of the line
 "wcell": it has the same contents as cell with the addition that (work wsq1)
 are inserted in the center of the F quad, and (work wsq2) are inserted in
 the center of the D quad.  The net result is that the required maps are
 computed, the corresponding lattice functions are computed, and the
 required horizontal beta functions are written into the first and second
 locations of the array ucalc for subsequent fitting.

#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 dr3      drft
   3.00000000000000
 inhfq    quad
  0.250000000000000       2.60000000000000       1.00000000000000
  0.000000000000000E+00
 outhfq   quad
  0.250000000000000       2.60000000000000      0.000000000000000E+00
   1.00000000000000
 inhdq    quad
  0.250000000000000      -2.40000000000000       1.00000000000000
  0.000000000000000E+00
 outhdq   quad
  0.250000000000000      -2.40000000000000      0.000000000000000E+00
   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 tasm     tasm
   1.00000000000000      1.000000000000000E-03   12.0000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 clear    iden
 inv      inv
 stm1     stm
   1.00000000000000
 gtm1     gtm
   1.00000000000000       1.00000000000000
 stm2     stm
   2.00000000000000
 gtm2     gtm
   1.00000000000000       2.00000000000000
```

```
 sq       sq
   15.0000000000000      0.000000000000000E+00   3.00000000000000
   1.00000000000000
 wsq1     wsq
   1.00000000000000       3.00000000000000      -1.00000000000000
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 wsq2     wsq
   1.00000000000000       3.00000000000000      -2.00000000000000
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 aim      aim
   2.00000000000000       17.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary     vary
  -1.00000000000000       19.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 bip      bip
   20.0000000000000
 tip      tip
  0.000000000000000E+00
 fit      fit
   1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
 fin      end
#lines
 cell
     1*hfq         1*dr3         1*hdq         1*dr3
 wcell
     1*inhfq       1*work        1*wsq1        1*outhfq      1*dr3      &
     1*inhdq       1*work        1*wsq2        1*outhdq      1*dr3
 hfq
     1*inhfq       1*outhfq
 hdq
     1*inhdq       1*outhdq
 setup
     1*clear       1*cell        1*stm1        1*clear
 work
     1*stm2        1*inv         1*gtm1        1*gtm2        1*tasm     &
     1*clear       1*gtm2
#lumps
#loops
#labor
    1*fileout
    1*setup
    1*sq
    1*wcell
    1*aim
    1*vary
    1*bip
    1*setup
    1*wcell
    1*fit
    1*tip
    1*fileout
```

```
    1*fin

*******************************
* Response to the command sq: *
*******************************

 In subroutine sq
accept  1:          bx

 Aims selected :
No.     item      present value
----------------------------------
 1        bx =      0.000000000E+00

********************************
* Response to the command aim: *
********************************

accept  1:      u(  1)
accept  2:      u(  2)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    u(  1) =       11.8093936          11.0000000
 2    u(  2) =       4.74820585          5.00000000

*********************************
* Response to the command vary: *
*********************************

No.  1 is inhfq    quad    .  Parameter 2 out of 4 selected.
    inhfq(2) =   2.6000000000000
No.  2 is inhdq    quad    .  Parameter 2 out of 4 selected.
    inhdq(2) =  -2.4000000000000

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    inhfq      quad         2       2.6000000000000
 2    inhdq      quad         2      -2.4000000000000
No.  1 is outhfq   quad    .  Parameter 2 out of 4 selected.
No.  2 is outhdq   quad    .  Parameter 2 out of 4 selected.

  Dependent #menu elements selected:
No.  Element    Type     Parameter   Present value       IDV  Slope
-----------------------------------------------------------------------
 1   outhfq     quad       2       2.6000000000000        1   1.00000
 2   outhdq     quad       2      -2.4000000000000        2   1.00000

**************************************
* Beginning of fitting logical loop. *
**************************************
```

```
 target           1 =   11.0000000000000
 target           2 =   5.00000000000000
Iter   1 Error= 8.0939E-01,  Step= 2.6000E-03,  SubErr= 8.0939E-01 @cut=    1
-------------------------------------------------------

*****************
* ETC. ETC. ETC. *
*****************

***************************
* End of fitting process. *
***************************

Iter   7 Error= 1.5515E-06,  Step= 3.0118E-07,  SubErr= 1.5515E-06 @cut=    1
-------------------------------------------------------

Quit on iteration   8 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    u(  1) =       11.0000000          11.0000000
 2    u(  2) =       5.00000000          5.00000000

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   inhfq      quad       2       2.3922257405147
 2   inhdq      quad       2      -1.8908550921822
 3   outhfq     quad       2       2.3922257405147        1    1.0000
 4   outhdq     quad       2      -1.8908550921822        2    1.0000

 Maximum error is     1.583444E-11
 Maximum allowed was  1.000000E-10

*****************************************************************
* Writing out new master input file with quad strenghts fitted. *
*****************************************************************

#comment
 Exhibit 10.5.1.
      This is an example of fitting quantities at two different locations
 in a lattice.  The lattice is a simple FODO cell, and the horizontal
 beta functions are fitted at the centers of the quads to achieve the aim

   horizontal beta function in center of F quad = 11,

   horizontal beta function in center of D quad = 5.

 The drifts in the FODO cell are each 3 meters long, and the quads are
 each .5 meters long.  The quads are split into "in" and "out" halves,
```

```
  F quad = hfq = inhfq outhfq,

  D quad = hdq = inhdq outhdq.
```

The beam parameters are those for 800 Mev protons.

The computation of lattice functions at various locations in a cell requires some explanation. To compute lattice functions at the center of the F quad it is necessary to compute the transfer map from the center of the F quad back to itself,

```
  map from center of F quad and back =  outhfq dr3 inhdq outhdq dr3 inhfq.
```

Similarly, to compute lattice functions at the center of the D quad it is necessary to compute the transfer map from the center of the D quad back to itself,

```
  map from center of D quad and back =  outhdq dr3 inhfq outhfq dr3 inhdq.
```

In each case, once the required map has been computed, the linear lattice functions can be found using a tasm command.

The computation of these two maps can be organized as follows: Let cell be the map for the FODO cell,

```
  cell = hfq dr3 hdq dr3 = inhfq outhfq dr3 inhdq outhdq dr3.
```

Evidently the maps from the centers of the F and D quads and back can be written in the form

```
  map from center of F quad and back =  (inhfq)**-1 cell (inhfq),

  map from center of D quad and back =

      (inhfq outhfq dr3 inhdq)**-1 cell (inhfq outhfq dr3 inhdq).
```

Thus, in each case, it is only necessary to carry out the following steps:

a) Accumulate (concatenate) the maps up to the observation point.

b) Store the accumulated map.

c) Invert the accumulated map.

d) Concatenate the resulting map with the map for cell (which itself needs to be computed only once beforehand and stored for repeated use).

e) Concatenate that result with the stored accumulated map.

Examination of the #lines component below shows that the line "work" carries out the operations b through e above as well as calling tasm. Finally, it again makes the current map the accumulated map up to that point.

```
 The map for cell itself is computed and stored by invoking the line
 "setup".  The reader is now prepared to understand the contents of the line
 "wcell": it has the same contents as cell with the addition that (work wsq1)
 are inserted in the center of the F quad, and (work wsq2) are inserted in
 the center of the D quad.  The net result is that the required maps are
 computed, the corresponding lattice functions are computed, and the
 required horizontal beta functions are written into the first and second
 locations of the array ucalc for subsequent fitting.

#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 dr3      drft
   3.00000000000000
 inhfq    quad
  0.250000000000000       2.39222574051469       1.00000000000000
  0.000000000000000E+00
 outhfq   quad
  0.250000000000000       2.39222574051469      0.000000000000000E+00
   1.00000000000000
 inhdq    quad
  0.250000000000000      -1.89085509218216       1.00000000000000
  0.000000000000000E+00
 outhdq   quad
  0.250000000000000      -1.89085509218216      0.000000000000000E+00
   1.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 tasm     tasm
   1.00000000000000      1.000000000000000E-03   12.0000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 clear    iden
 inv      inv
 stm1     stm
   1.00000000000000
 gtm1     gtm
   1.00000000000000       1.00000000000000
 stm2     stm
   2.00000000000000
 gtm2     gtm
   1.00000000000000       2.00000000000000
 sq       sq
   15.0000000000000      0.000000000000000E+00   3.00000000000000
   1.00000000000000
 wsq1     wsq
   1.00000000000000       3.00000000000000      -1.00000000000000
   1.00000000000000       2.00000000000000      0.000000000000000E+00
```

```
 wsq2     wsq
   1.00000000000000       3.00000000000000      -2.00000000000000
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 aim      aim
   2.00000000000000       17.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary     vary
  -1.00000000000000       19.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 bip      bip
   20.0000000000000
 tip      tip
  0.000000000000000E+00
 fit      fit
   1.00000000000000      0.000000000000000E+00  1.000000000000000E-10
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
 fin      end
#lines
 cell
     1*hfq         1*dr3         1*hdq         1*dr3
 wcell
     1*inhfq       1*work        1*wsq1        1*outhfq      1*dr3      &
     1*inhdq       1*work        1*wsq2        1*outhdq      1*dr3
 hfq
     1*inhfq       1*outhfq
 hdq
     1*inhdq       1*outhdq
 setup
     1*clear       1*cell        1*stm1        1*clear
 work
     1*stm2        1*inv         1*gtm1        1*gtm2        1*tasm     &
     1*clear       1*gtm2
#lumps
#loops
#labor
    1*fileout
    1*setup
    1*sq
    1*wcell
    1*aim
    1*vary
    1*bip
    1*setup
    1*wcell
    1*fit
    1*tip
    1*fileout
    1*fin

end of MARYLIE run
```

10.6 Construction of Third-Order Achromats

An achromat is a collection of beam-line elements that transports and bends a beam but otherwise, through some order, does as little to a beam as possible. What may be called an *ordinary* achromat is a system whose net transfer map (through some order) is the identity map except for energy dependent time-of-flight (temporal dispersion) terms. The construction of such achromats, at least through third order, is relatively easy and is described in subsection 10.6.1. A *complete* achromat is a system whose net transfer map is exactly the identity map through some order. The construction of complete achromats is somewhat harder. An example of such a construction is given in subsection 10.6.2. Again, in no case is it claimed that these constructions are optimal. They are only proofs of principle. Finally, all achromats constructed in this section make use of *repetitive* symmetry. Other kinds of symmetry (e.g. reversal symmetry, see section 7.11) may also be of value, but that is not explored here.

10.6.1 Construction of Ordinary Third-Order Achromat

The net transfer map $\mathcal{M}$ for an ordinary achromat is (through third order) of the form

$$\mathcal{M} = \exp(: aP_\tau^2 + bP_\tau^3 + cP_\tau^4 :). \tag{10.6.1}$$

One way to construct such a device is to use 5 identical cells (repetitive symmetry) with each cell having a tune of 1/5 in both transverse planes.

Let $\mathcal{M}_c$ be the map for a single cell, and let $\mathcal{N}_c$ be its normal form. That is, write $\mathcal{M}_c$ in the form

$$\mathcal{M}_c = \mathcal{A}^{-1}\mathcal{N}_c\mathcal{A}. \tag{10.6.2}$$

The normal form map $\mathcal{N}_c$ will generally be of the form

$$\mathcal{N}_c = \mathcal{R}_c \exp(: g_3 :) \exp(: g_4 :) \tag{10.6.3}$$

where (under the assumption of midplane symmetry)

$$g_3 = *(P_x^2 + X^2)P_\tau + (P_y^2 + Y^2)P_\tau + *P_\tau^3, \tag{10.6.4}$$

$$\begin{aligned} g_4 &= *(P_x^2 + X^2)^2 + *(P_x^2 + X^2)(P_y^2 + Y^2) + *(P_y^2 + Y^2)^2 \\ &+ *(P_x^2 + X^2)P_\tau^2 + *(P_y^2 + Y^2)P_\tau^2 + *P_\tau^4 \\ &+ *\mathrm{R20020} + *\mathrm{I20020}. \end{aligned} \tag{10.6.5}$$

Here the asterisks "*" denote possibly nonzero coefficients, and the symbols R20020 and I20020 are shorthand for the $(2\nu_x - 2\nu_y)$ resonance terms. See section 14.2. Because both transverse tunes are 1/5, the linear map $\mathcal{R}_c$ will satisfy the condition

$$(\mathcal{R}_c)^5 = \exp(: aP_\tau^2 :). \tag{10.6.6}$$

Suppose each cell contains a pair of sextupoles and these sextupoles are tuned to zero out the first-order chromaticity terms $(P_x^2 + X^2)P_\tau$and $(P_y^2 + Y^2)P_\tau$ in g_3. Suppose each cell

also contains 7 octupoles and that these octupoles are tuned to zero out all the terms in g_4 except $*P_\tau^4$. When this is done, the nonlinear part of $\mathcal{N}_c$ has only time-of-flight terms,

$$\mathcal{N}_c = \mathcal{R}_c \exp(: *P_\tau^3 + *P_\tau^4). \tag{10.6.7}$$

Since both $\mathcal{R}_c$ and $\mathcal{A}$ are static maps (have no τ dependent terms), they both commute with $: P_\tau :$. It follows that

$$\begin{aligned} \mathcal{M} &= (\mathcal{M}_c)^5 = \mathcal{A}^{-1}(\mathcal{N}_c)^5\mathcal{A} = \mathcal{A}^{-1}(\mathcal{R}_c)^5 \exp(: bP_\tau^3 + cP_\tau^4 :)\mathcal{A} \\ &= \mathcal{A}^{-1} \exp(: aP_\tau^2 + bP_\tau^3 + cP_\tau^4 :)\mathcal{A} = \exp(: aP_\tau^2 + bP_\tau^3 + cP_\tau^4 :). \end{aligned} \tag{10.6.8}$$

Thus, the system described by $\mathcal{M}$ will be an (ordinary) third-order achromat.

Exhibit 10.6.1 below shows an example of such a system. The horizontal and vertical tunes have already been set to 1/5 in an earlier run similar to that described in section 10.2. For fun all the offensive Lie generators are fit to zero at once. See Exhibits 10.6.1a and 10.6.1b. However, in principle it is better to first fit the offensive terms in g_3 to zero in one logical fitting loop, and then fit those in g_4 to zero in a subsequent loop. The nature of the fitting process itself is described in the *#comments* section of Exhibit 10.6.1c. The calculations at the end of the exhibit verify that, after fitting, the five-cell system does indeed comprise an ordinary achromat. That is, the matrix part of the map is the identity save for a 5,6 entry that corresponds to the factor $\exp(: aP_\tau^2 :)$, and all other Lie generators vanish save for bP_τ^3 and cP_τ^4.

```
Exhibit 10.6.1a
Contents of file 15 that provides instructions for the command aim:

 1    f( 28) =      0.000000000E+00
 2    f( 29) =      0.000000000E+00
 3    f( 84) =      0.000000000E+00
 4    f( 85) =      0.000000000E+00
 5    f( 87) =      0.000000000E+00
 6    f( 88) =      0.000000000E+00
 7    f( 89) =      0.000000000E+00
 8    f(152) =      0.000000000E+00
 9    f(153) =      0.000000000E+00
 #


Exhibit 10.6.1b
Contents of file 17 that provides instructions for the command vary:

 hcs        2
 vcs        2
 oct1       2
 oct2       2
 oct3       2
 oct4       2
 oct5       2
 oct6       2
 oct7       2
```

```
  #


Exhibit 10.6.1c
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.6.1.
      This is a study of a five-cell "ordinary" achromat based
 on a cell similar to that for the Los Alamos Proton Storage Ring.
 See section 2.5.  The tunes have already been fit to 1/5.  See
 the response below to the tasm command.  This run does 4 things:

  a) Computes the static normal form map for a cell and displays
     the f3 and f4 generators for the map in a static resonance
     basis.  See section 14.2.

  b) Fits the offensive terms in the generators by adjusting
     2 sextupoles and 7 octupoles so that

         f( 28)=f( R11001 )=0
         f( 29)=f( R00111 )=0
         f( 84)=f( R11002 )=0
         f( 85)=f( R00112 )=0
         f( 87)=f( R22000 )=0
         f( 88)=f( R00220 )=0
         f( 89)=f( R11110 )=0
         f(152)=f( R20020 )=0
         f(153)=f( I20020 )=0

  c) Displays that the map (in a Cartesian basis) for the
     fitted cell still has many nonlinear generators,
     as might be expected.

  d) Verifies that five such cells make an ordinary achromat.

#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 czer     zer
  0.000000000000000E+00  5.000000000000000E-09  0.000000000000000E+00
 dr       drft
  0.300000000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
```

```
hfq       quad
 0.500000000000000       1.72870070795707       1.00000000000000
  1.00000000000000
hdq       quad
 0.500000000000000     -0.885058164867516       1.00000000000000
  1.00000000000000
hcs       sext
 0.500000000000000      0.000000000000000E+00
vcs       sext
 0.500000000000000      0.000000000000000E+00
oct1      octm
 0.250000000000000      0.000000000000000E+00
oct2      octm
 0.500000000000000      0.000000000000000E+00
oct3      octm
 0.500000000000000      0.000000000000000E+00
oct4      octm
 0.500000000000000      0.000000000000000E+00
oct5      octm
 0.500000000000000      0.000000000000000E+00
oct6      octm
 0.500000000000000      0.000000000000000E+00
oct7      octm
 0.500000000000000      0.000000000000000E+00
fileout   pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout    ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
tasm      tasm
  2.00000000000000      1.000000000000000E-03   1.00000000000000
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
snor      snor
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00
gbuf2     gbuf
  2.00000000000000       2.00000000000000
ctosr     tbas
  1.00000000000000
rmapout   ptm
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   2.00000000000000
clear     iden
bip       bip
  20.0000000000000
tip       tip
 0.000000000000000E+00
aim       aim
  2.00000000000000       15.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
vary      vary
  1.00000000000000       17.0000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000       1.00000000000000
```

```
 fit      fit
   1.00000000000000      0.000000000000000E+00  5.000000000000000E-11
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
 fin      end
#lines
 cell
     1*oct1        1*dr          1*oct2        1*dr          1*hdq      &
     1*dr          1*oct3        1*dr          1*vcs         1*dr       &
     1*oct4        1*dr          1*bend        1*dr          1*oct5     &
     1*dr          1*hcs         1*dr          1*oct6        1*dr       &
     1*hfq         1*dr          1*oct7        1*dr          1*oct1
 achromat
     5*cell
#lumps
#loops
#labor
    1*fileout
    1*czer
    1*cell
    1*tasm
    1*snor
    1*gbuf2
    1*ctosr
    1*rmapout
    1*aim
    1*vary
    1*bip
    1*clear
    1*cell
    1*snor
    1*gbuf2
    1*ctosr
    1*fit
    1*tip
    1*rmapout
    1*clear
    1*cell
    1*mapout
    1*clear
    1*achromat
    1*mapout
    1*fin

****************************************
* Response to tasm command illustrating *
* that the tunes have been fit to 1/5.  *
****************************************

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.200000000000008
```

```
first order horizontal chromaticity = -0.105040682704069
second order horizontal chromaticity =   2.75215458263220
horizontal tune when delta =  1.000000000000000E-003
 0.199897711471887

vertical tune =  0.200000000000007
first order vertical chromaticity = -0.272925216767796
second order vertical chromaticity =   1.68583159998465
vertical tune when delta =  1.000000000000000E-003
 0.199728760614839

tune separation when delta=  1.000000000000000E-003
 1.689508570480169E-004
  5.14062615116981      -0.519186710495935       0.000000000000000E+000
 0.000000000000000E+000
  5.14062615116981      -0.519186710495935       0.000000000000000E+000
 0.000000000000000E+000

normalized anharmonicities
 hhn=  0.271979146437940
 vvn=  9.484809327375197E-002
 hvn=  0.303162388848243

****************************************
* Lie generators for cell normal form *
* map in the static resonance basis.  *
****************************************

nonzero elements in generating polynomial in
the static resonance basis are :

f( 28)=f( R11001 )=-0.39228609609582
f( 29)=f( R00111 )= -1.0192695349627
f( 30)=f( R00003 )= -3.3913541328042
f( 84)=f( R11002 )= -12.286552529786
f( 85)=f( R00112 )= -7.6614923782636
f( 86)=f( R00004 )= -18.214644014768
f( 87)=f( R22000 )=-0.42722384418953
f( 88)=f( R00220 )=-0.14898703651791
f( 89)=f( R11110 )=-0.95241273365037
f(152)=f( R20020 )=-0.35459141937509
f(153)=f( I20020 )=-8.76601451451970E-02

*******************************
* Response to the command aim: *
*******************************

accept  1:      f( 28)
accept  2:      f( 29)
accept  3:      f( 84)
accept  4:      f( 85)
accept  5:      f( 87)
accept  6:      f( 88)
```

```
accept  7:      f( 89)
accept  8:      f(152)
accept  9:      f(153)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f( 28) =     -0.392286096         0.000000000E+00
 2    f( 29) =      -1.01926953         0.000000000E+00
 3    f( 84) =      -12.2865525         0.000000000E+00
 4    f( 85) =      -7.66149238         0.000000000E+00
 5    f( 87) =     -0.427223844         0.000000000E+00
 6    f( 88) =     -0.148987037         0.000000000E+00
 7    f( 89) =     -0.952412734         0.000000000E+00
 8    f(152) =     -0.354591419         0.000000000E+00
 9    f(153) =     -8.766014515E-02     0.000000000E+00

**********************************
* Response to the command vary: *
**********************************

No.  1 is hcs      sext    .  Parameter 2 out of 2 selected.
      hcs(2) =  0.00000000000000E+00
No.  2 is vcs      sext    .  Parameter 2 out of 2 selected.
      vcs(2) =  0.00000000000000E+00
No.  3 is oct1     octm    .  Parameter 2 out of 2 selected.
     oct1(2) =  0.00000000000000E+00
No.  4 is oct2     octm    .  Parameter 2 out of 2 selected.
     oct2(2) =  0.00000000000000E+00
No.  5 is oct3     octm    .  Parameter 2 out of 2 selected.
     oct3(2) =  0.00000000000000E+00
No.  6 is oct4     octm    .  Parameter 2 out of 2 selected.
     oct4(2) =  0.00000000000000E+00
No.  7 is oct5     octm    .  Parameter 2 out of 2 selected.
     oct5(2) =  0.00000000000000E+00
No.  8 is oct6     octm    .  Parameter 2 out of 2 selected.
     oct6(2) =  0.00000000000000E+00
No.  9 is oct7     octm    .  Parameter 2 out of 2 selected.
     oct7(2) =  0.00000000000000E+00

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    hcs        sext         2      0.00000000000000E+00
 2    vcs        sext         2      0.00000000000000E+00
 3    oct1       octm         2      0.00000000000000E+00
 4    oct2       octm         2      0.00000000000000E+00
 5    oct3       octm         2      0.00000000000000E+00
 6    oct4       octm         2      0.00000000000000E+00
 7    oct5       octm         2      0.00000000000000E+00
 8    oct6       octm         2      0.00000000000000E+00
 9    oct7       octm         2      0.00000000000000E+00
```

```
****************************
* Start of fitting process: *
****************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
 target           4 =  0.000000000000000E+000
 target           5 =  0.000000000000000E+000
 target           6 =  0.000000000000000E+000
 target           7 =  0.000000000000000E+000
 target           8 =  0.000000000000000E+000
 target           9 =  0.000000000000000E+000

Iter   1 Error= 1.2287E+01,  Step= 1.0000E-05,  SubErr= 1.2287E+01 @cut=    1
-------------------------------------------------------

*****************
* ETC. ETC. ETC. *
*****************

Iter  15 Error= 7.1680E-11,  Step= 2.5141E-09,  SubErr= 7.1680E-11 @cut=    1
-------------------------------------------------------

***************************
* End of fitting process: *
***************************

Quit on iteration  16 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f( 28) =      3.073983097E-16     0.000000000E+00
 2    f( 29) =     -1.735744426E-15     0.000000000E+00
 3    f( 84) =     -2.103206498E-11     0.000000000E+00
 4    f( 85) =     -5.513811629E-12     0.000000000E+00
 5    f( 87) =     -6.439293543E-13     0.000000000E+00
 6    f( 88) =     -8.200107260E-13     0.000000000E+00
 7    f( 89) =     -1.000088901E-12     0.000000000E+00
 8    f(152) =      6.528111385E-13     0.000000000E+00
 9    f(153) =     -1.088018564E-13     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   hcs        sext       2      0.45738388996717
 2   vcs        sext       2     -0.71686155644278
 3   oct1       octm       2      -49.658510427750
 4   oct2       octm       2      -39.836629522527
 5   oct3       octm       2       68.022002709280
 6   oct4       octm       2      -49.277842476621
```

```
 7   oct5       octm       2       48.388287465694
 8   oct6       octm       2      -72.003461762391
 9   oct7       octm       2       94.871363144729

 Maximum error is     2.103206E-11
 Maximum allowed was  5.000000E-11

*********************************************************
* Confirmation that all nonlinear generators have been  *
* removed except energy dependent time-of-flight terms: *
*********************************************************

nonzero elements in generating polynomial in
the static resonance basis are :

f( 30)=f( R00003 )= -2.7017339551965
f( 86)=f( R00004 )= -16.577513712914

************************
* Map for fitted cell: *
************************

matrix for map is :

 1.41271E+00  9.82535E+00  0.00000E+00  0.00000E+00  0.00000E+00 -3.54203E+00
-2.16038E-01 -7.94679E-01  0.00000E+00  0.00000E+00  0.00000E+00 -2.12550E-01
 0.00000E+00  0.00000E+00 -5.00626E-01  7.51094E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -2.07701E-01  1.11866E+00  0.00000E+00  0.00000E+00
 1.06549E+00  4.90315E+00  0.00000E+00  0.00000E+00  1.00000E+00  4.58067E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=-8.72327683853229E-03
 f( 29)=f( 21 00 00 )= 7.60724728692393E-02
 f( 33)=f( 20 00 01 )=-0.20062065357347
 f( 34)=f( 12 00 00 )=-0.44374011377906
 f( 38)=f( 11 00 01 )= -1.9135093070850
 f( 39)=f( 10 20 00 )= 7.30532670999164E-02
 f( 40)=f( 10 11 00 )=-0.38053025668887
 f( 43)=f( 10 02 00 )= 0.42873435511120
 f( 48)=f( 10 00 02 )= -1.1734603194920
 f( 49)=f( 03 00 00 )= -5.3829856680555
 f( 53)=f( 02 00 01 )= -10.353119823222
 f( 54)=f( 01 20 00 )= 0.93364238881311
 f( 55)=f( 01 11 00 )= -9.9153819701391
 f( 58)=f( 01 02 00 )=  26.121323755701
 f( 63)=f( 01 00 02 )= -6.1813341273347
 f( 67)=f( 00 20 01 )=-4.84106691153551E-02
 f( 70)=f( 00 11 01 )= 0.16319300504732
 f( 76)=f( 00 02 01 )= -2.2910002594037
 f( 83)=f( 00 00 03 )= -5.3504737382645
 f( 84)=f( 40 00 00 )=-0.57077243308329
```

```
 f( 85)=f( 31 00 00 )=  18.798412194369
 f( 89)=f( 30 00 01 )=  8.9481929564508
 f( 90)=f( 22 00 00 )=  342.53269801100
 f( 94)=f( 21 00 01 )=  339.33355073316
 f( 95)=f( 20 20 00 )= 0.58831468012863
 f( 96)=f( 20 11 00 )=  56.061390927939
 f( 99)=f( 20 02 00 )= -204.23714100640
 f(104)=f( 20 00 02 )=  83.337734413186
 f(105)=f( 13 00 00 )=  2775.2632369420
 f(109)=f( 12 00 01 )=  4158.0133168998
 f(110)=f( 11 20 00 )= -113.76908354845
 f(111)=f( 11 11 00 )=  1457.4068490211
 f(114)=f( 11 02 00 )= -4935.5704153071
 f(119)=f( 11 00 02 )=  2083.8585681415
 f(123)=f( 10 20 01 )= -50.944992792765
 f(126)=f( 10 11 01 )=  717.98692352551
 f(132)=f( 10 02 01 )= -2476.1870284595
 f(139)=f( 10 00 03 )=  343.67889208102
 f(140)=f( 04 00 00 )=  8570.7839447688
 f(144)=f( 03 00 01 )=  17044.964493973
 f(145)=f( 02 20 00 )= -643.17214600877
 f(146)=f( 02 11 00 )=  8765.5286681778
 f(149)=f( 02 02 00 )= -29732.376175673
 f(154)=f( 02 00 02 )=  12792.301076641
 f(158)=f( 01 20 01 )= -627.87468256384
 f(161)=f( 01 11 01 )=  8614.7475493499
 f(167)=f( 01 02 01 )= -29453.240311761
 f(174)=f( 01 00 03 )=  4270.6396991783
 f(175)=f( 00 40 00 )=-9.50239478347628E-02
 f(176)=f( 00 31 00 )= -25.119373689691
 f(179)=f( 00 22 00 )=  305.05043014208
 f(184)=f( 00 20 02 )= -157.78902945287
 f(185)=f( 00 13 00 )= -1439.6754113905
 f(190)=f( 00 11 02 )=  2194.9822397543
 f(195)=f( 00 04 00 )=  2502.4896243607
 f(200)=f( 00 02 02 )= -7536.4831205450
 f(209)=f( 00 00 04 )=  525.71263685475

*************************************************************
* Map for resulting "ordinary" achromat.  Note that        *
* the matrix part is the identity except for the 5,6       *
* energy dependent time-of-flight entry that corresponds   *
* to the quadratic generator Ptau**2; and all cubic and    *
* quartic generators vanish except for the energy          *
* dependent time-of-flight generators Ptau**3 and Ptau**4. *
*************************************************************

matrix for map is :

 1.00000E+00  2.64003E-12  0.00000E+00  0.00000E+00  0.00000E+00  1.88369E-13
-5.81757E-14  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.71932E-13
 0.00000E+00  0.00000E+00  1.00000E+00  1.62059E-12  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -4.46310E-14  1.00000E+00  0.00000E+00  0.00000E+00
```

```
 1.71280E-13  1.74971E-13  0.00000E+00  0.00000E+00  1.00000E+00  5.47836E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 83)=f( 00 00 03 )= -13.508669775982
 f(209)=f( 00 00 04 )= -82.887568564841

end of MARYLIE run
```

10.6.2 Construction of Complete Third-Order Achromat

The construction of a complete achromat is similar to that of an ordinary achromat except for a difficult first step. For a complete achromat the coefficients a, b, c in (10.6.1) must also vanish. What is required initially is that the cell be isochronous in the sense that the matrix part for the cell map normal form must have a zero 5,6 entry. If this can be accomplished (again assuming transverse tunes of 1/5), then $\mathcal{R}_c$ has (through fourth-order generators) the property

$$(\mathcal{R}_c)^5 = \mathcal{I} \tag{10.6.9}$$

where $\mathcal{I}$ denotes the identity map. That is, the coefficient a in (10.6.1) has been made to vanish. In this example, a is made to vanish by suitably powering a third quadrupole in each cell.

As before, $\mathcal{N}_c$ has the form (10.6.3). Next, 3 sextupoles are fit to remove *all* the terms in g_3 as given by (10.6.4). Finally, 8 octupoles are fit to remove *all* the terms in g_4 as given by (10.6.5). The net result is that $\mathcal{N}_c$ takes the form

$$\mathcal{N}_c = \mathcal{R}_c. \tag{10.6.10}$$

It follows (through fourth-order generators) that

$$\mathcal{M} = (\mathcal{M}_c)^5 = \mathcal{A}^{-1}(\mathcal{N}_c)^5\mathcal{A} = \mathcal{A}^{-1}\mathcal{I}\mathcal{A} = \mathcal{I}. \tag{10.6.11}$$

Consequently, 5 such cells form a complete third-order achromat.

Exhibit 10.6.2 shows an example of such a complete achromat. Fitting is done in 3 consecutive logical loops. Exhibits 10.6.2a and 10.6.2b show *aim1* and *vary1* for the first logical loop, and Exhibits 10.6.2c through 10.6.2f show their counterparts for the other two logical loops. Evidently quadrupole strengths are fit in the first loop, sextupole strengths in the second, and octupole strengths in the third. Exhibit 10.6.2g shows the MaryLie run itself. Observe, by examining the end of the run, that the five-cell system is indeed a complete achromat (the matrix part of the map is the identity, and all Lie generators vanish) as desired.

```
Exhibit 10.6.2a
Contents of file 15 that provides instructions for the command aim1:

 1    r(5,6) =      0.000000000E+00
 2        tx =      0.200000000
 3        ty =      0.200000000
```

```
 #
```

Exhibit 10.6.2b
Contents of file 16 that provides instructions for the command vary1:

```
 hfq        2
 hdq        2
 cq         2
  #
```

Exhibit 10.6.2c
Contents of file 17 that provides instructions for the command aim2:

```
 1    f( 28) =      0.000000000E+00
 2    f( 29) =      0.000000000E+00
 3    f( 30) =      0.000000000E+00
 #
```

Exhibit 10.6.2d
Contents of file 18 that provides instructions for the command vary2:

```
 sex1       2
 sex2       2
 sex3       2
  #
```

Exhibit 10.6.2e
Contents of file 19 that provides instructions for the command aim3:

```
 1    f( 84) =      0.000000000E+00
 2    f( 85) =      0.000000000E+00
 3    f( 86) =      0.000000000E+00
 4    f( 87) =      0.000000000E+00
 5    f( 88) =      0.000000000E+00
 6    f( 89) =      0.000000000E+00
 7    f(152) =      0.000000000E+00
 8    f(153) =      0.000000000E+00
 #
```

Exhibit 10.6.2f
Contents of file 20 that provides instructions for the command vary3:

```
 oct1       2
 oct2       2
 oct3       2
 oct4       2
 oct5       2
 oct6       2
```

```
 oct7       2
 oct8       2
  #


Exhibit 10.6.2g
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.6.2.
      This is an example of a complete third-order achromat.  The
 beam parameters are those for 1.5 Mev electrons.  This run consists
 of three successive logical fitting loops followed by two subsequent
 calculations:

  a) It fits 3 quads to achieve the condition that the cell have
     transverse tunes of 1/5 and that the cell normal form map
     5,6 matrix entry vanish.

  b) It fits 3 sextupoles to achieve the condition that in a static
     resonance basis the 3 offensive f3 generators (for a complete
     achromat) vanish.

  c) It fits 8 octupoles to achieve the condition that in a static
     resonance basis the 8 offensive f4 generators (for a complete
     achromat) vanish.

  d) It displays the map for the fitted cell (in a cartesian basis)
     to show that it still has many nonlinear generators.

  e) It verifies that five such cells make a complete achromat.

#beam
 6.487797796153699E-003
  2.93542614344535
  1.00000000000000
  1.00000000000000
#menu
 zer      zer
  0.000000000000000E+00  2.000000000000000E-08  0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 next     pmif
   1.00000000000000       30.0000000000000       3.00000000000000
 sex1     sext
  0.100000000000000      0.100000000000000
 sex2     sext
  0.100000000000000      0.200000000000000
 sex3     sext
```

```
 0.100000000000000      0.300000000000000
oct1     octm
 0.100000000000000      0.100000000000000
oct2     octm
 0.100000000000000      0.200000000000000
oct3     octm
 0.100000000000000      0.300000000000000
oct4     octm
 0.100000000000000      0.400000000000000
oct5     octm
 0.100000000000000      0.500000000000000
oct6     octm
 0.100000000000000      0.600000000000000
oct7     octm
 0.100000000000000      0.700000000000000
oct8     octm
 0.100000000000000      0.800000000000000
dr.1     drft
 0.100000000000000
dr.2     drft
 0.200000000000000
dr.3     drft
 0.300000000000000
dr.6     drft
 0.600000000000000
dr.8     drft
 0.800000000000000
dr.9     drft
 0.900000000000000
hbend    nbnd
  9.00000000000000      0.000000000000000E+00  0.500000000000000
 1.300000000000000E-03   1.00000000000000       1.00000000000000
hfq      quad
 0.300000000000000      5.200000000000000E-03   1.00000000000000
  1.00000000000000
hdq      quad
 0.300000000000000     -5.200000000000000E-03   1.00000000000000
  1.00000000000000
cq       quad
 0.300000000000000     -6.300000000000000E-04   1.00000000000000
  1.00000000000000
cmapout  ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
tasm     tasm
  2.00000000000000      1.000000000000000E-03   1.00000000000000
 0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
snor     snor
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00
gbuf2    gbuf
  2.00000000000000       2.00000000000000
ctosr    tbas
```

```
   1.00000000000000
 rmapout  ptm
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   2.00000000000000
 clear    iden
 bip      bip
   20.0000000000000
 tip      tip
  0.000000000000000E+00
 aim1     aim
   2.00000000000000       15.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 aim2     aim
   2.00000000000000       17.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 aim3     aim
   2.00000000000000       19.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary1    vary
   1.00000000000000       16.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary2    vary
   1.00000000000000       18.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 vary3    vary
   1.00000000000000       20.0000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000       1.00000000000000
 fit      fit
   1.00000000000000      0.000000000000000E+00  2.000000000000000E-12
  1.000000000000000E-03  0.000000000000000E+00   1.00000000000000
 end      end
#lines
 cell
     1*dr.9        1*oct1        1*dr.8        1*oct2        1*dr.1     &
     1*hfq         1*dr.1        1*sex1        1*dr.1        1*oct3     &
     1*dr.2        1*sex2        1*dr.1        1*oct4        1*dr.1     &
     1*hbend       1*dr.3        1*cq          1*dr.3        1*hbend    &
     1*dr.6        1*sex3        1*dr.1        1*oct5        1*dr.1     &
     1*hdq         1*dr.1        1*oct6        1*dr.8        1*oct7     &
     1*dr.8        1*oct8
 cachro
     5*lcell
#lumps
 lcell
     1*cell
#loops
#labor
    1*fileout
    1*zer
    1*cell
    1*tasm
    1*snor
    1*gbuf2
```

```
    1*cmapout
    1*aim1
    1*vary1
    1*bip
    1*clear
    1*cell
    1*tasm
    1*snor
    1*gbuf2
    1*fit
    1*tip
    1*tasm
    1*cmapout
    1*clear
    1*cell
    1*snor
    1*gbuf2
    1*ctosr
    1*rmapout
    1*aim2
    1*vary2
    1*bip
    1*clear
    1*cell
    1*snor
    1*gbuf2
    1*ctosr
    1*fit
    1*tip
    1*rmapout
    1*aim3
    1*vary3
    1*bip
    1*clear
    1*cell
    1*snor
    1*gbuf2
    1*ctosr
    1*fit
    1*tip
    1*rmapout
    1*clear
    1*cell
    1*cmapout
    1*clear
    1*cachro
    1*cmapout
    1*end

***************************
* Tunes prior to fitting: *
***************************
```

```
twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.201835940492592
first order horizontal chromaticity =   25.9927865117704
second order horizontal chromaticity =  -3952.62913129948
horizontal tune when delta =  1.000000000000000E-003
 0.223876097873063

vertical tune =  0.198008838192001
first order vertical chromaticity =  -22.6817278094467
second order vertical chromaticity =   1321.08383112491
vertical tune when delta =  1.000000000000000E-003
 0.176648194213679

tune separation when delta=  1.000000000000000E-003
 4.722790365938462E-002

normalized anharmonicities
 hhn=  -2454.25146548007
 vvn=  -1666.13809887793
 hvn=   1120.96589120575

*****************************************************************
* Normal form map for cell in cartesian basis prior to fitting: *
*****************************************************************

matrix for map is :

 2.98026E-01  9.54558E-01  0.00000E+00  0.00000E+00  0.00000E+00 -1.60944E-16
-9.54558E-01  2.98026E-01  0.00000E+00  0.00000E+00  0.00000E+00  1.87381E-16
 0.00000E+00  0.00000E+00  3.20891E-01  9.47116E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -9.47116E-01  3.20891E-01  0.00000E+00  0.00000E+00
 2.21591E-16  3.21888E-16  0.00000E+00  0.00000E+00  1.00000E+00  1.21309E-02
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=  84.429957624290
 f( 53)=f( 02 00 01 )=  84.429957624290
 f( 67)=f( 00 20 01 )= -73.674952738525
 f( 76)=f( 00 02 01 )= -73.674952738525
 f( 83)=f( 00 00 03 )=  26.990573886779
 f( 84)=f( 40 00 00 )=  3855.1291870071
 f( 90)=f( 22 00 00 )=  7710.2583740142
 f( 95)=f( 20 20 00 )= -3521.6182087416
 f( 99)=f( 20 02 00 )= -3521.6182087316
 f(104)=f( 20 00 02 )=  13277.485905428
 f(111)=f( 11 11 00 )=-2.01980583369732E-08
 f(140)=f( 04 00 00 )=  3855.1291870071
 f(145)=f( 02 20 00 )= -3521.6182087318
 f(149)=f( 02 02 00 )= -3521.6182087420
```

```
 f(154)=f( 02 00 02 )=  13277.485905428
 f(175)=f( 00 40 00 )=  2617.1636056505
 f(179)=f( 00 22 00 )=  5234.3272113010
 f(184)=f( 00 20 02 )= -4439.2399818658
 f(195)=f( 00 04 00 )=  2617.1636056505
 f(200)=f( 00 02 02 )= -4439.2399818658
 f(209)=f( 00 00 04 )=  4884.3139306985

**********************************
* Response to the command aim1: *
**********************************

accept  1:      r(5,6)
accept  2:          tx
accept  3:          ty

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    r(5,6) =      1.213086180E-02     0.000000000E+00
 2        tx =      0.201835940         0.200000000
 3        ty =      0.198008838         0.200000000

**********************************
* Response to the command vary1: *
**********************************

No.  1 is hfq      quad    .  Parameter 2 out of 4 selected.
      hfq(2) =  5.20000000000000E-03
No.  2 is hdq      quad    .  Parameter 2 out of 4 selected.
      hdq(2) = -5.20000000000000E-03
No.  3 is cq       quad    .  Parameter 2 out of 4 selected.
       cq(2) = -6.30000000000000E-04

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    hfq        quad         2      5.20000000000000E-03
 2    hdq        quad         2     -5.20000000000000E-03
 3    cq         quad         2     -6.30000000000000E-04

*******************************************
* Beginning of first logical fitting loop: *
*******************************************

twiss analysis of static map

tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.201835940492592
first order horizontal chromaticity =   25.9927865117704
second order horizontal chromaticity =  -3952.62913129948
horizontal tune when delta =  1.000000000000000E-003
```

```
 0.223876097873063

vertical tune =  0.198008838192001
first order vertical chromaticity =  -22.6817278094467
second order vertical chromaticity =   1321.08383112491
vertical tune when delta =  1.000000000000000E-003
 0.176648194213679

tune separation when delta=  1.000000000000000E-003
 4.722790365938462E-002

normalized anharmonicities
 hhn=  -2454.25146548007
 vvn=  -1666.13809887793
 hvn=   1120.96589120575

 target           1 =  0.000000000000000E+000
 target           2 =  0.200000000000000
 target           3 =  0.200000000000000

Iter   1 Error= 1.2131E-02,  Step= 1.0000E-05,  SubErr= 1.2131E-02 @cut=    1
-------------------------------------------------------

******************
* ETC. ETC. ETC. *
******************

Quit on iteration  10 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
---------------------------------------------------
 1    r(5,6) =     -2.220446049E-16     0.000000000E+00
 2        tx =      0.200000000         0.200000000
 3        ty =      0.200000000         0.200000000

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-------------------------------------------------------------------------
 1   hfq        quad       2      5.17149206779275E-03
 2   hdq        quad       2     -5.23280935347573E-03
 3   cq         quad       2     -6.26171204808990E-04

 Maximum error is     3.885781E-16
 Maximum allowed was  2.000000E-12

***********************************************
* Verify that tunes for fitted cell are 1/5: *
***********************************************

twiss analysis of static map
```

```
tunes and chromaticities for delta defined in terms of momentum deviation:

horizontal tune =  0.200000000000000
first order horizontal chromaticity =   26.6669650947606
second order horizontal chromaticity =  -4220.68383992105
horizontal tune when delta =  1.000000000000000E-003
 0.222446281254839

vertical tune =  0.200000000000000
first order vertical chromaticity =  -22.9030466022963
second order vertical chromaticity =   1435.06078760167
vertical tune when delta =  1.000000000000000E-003
 0.178532014185305

tune separation when delta=  1.000000000000000E-003
 4.391426706953436E-002

normalized anharmonicities
 hhn=  -2546.92498409510
 vvn=  -1697.62525984766
 hvn=   1229.38912398208

*******************************************************
* Normal form map (in cartesian basis) for cell after *
* quadrupole fit.  Note that the 5,6 matrix element   *
* vanishes as desired.                                *
*******************************************************

matrix for map is :

 3.09017E-01  9.51057E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-9.51057E-01  3.09017E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.09017E-01  9.51057E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -9.51057E-01  3.09017E-01  0.00000E+00  0.00000E+00
-1.87328E-16 -1.60254E-16  0.00000E+00  0.00000E+00  1.00000E+00 -2.22045E-16
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 33)=f( 20 00 01 )=  86.619829386107
 f( 53)=f( 02 00 01 )=  86.619829386107
 f( 67)=f( 00 20 01 )= -74.393842046268
 f( 76)=f( 00 02 01 )= -74.393842046268
 f( 83)=f( 00 00 03 )=  28.517013718012
 f( 84)=f( 40 00 00 )=  4000.7004096387
 f( 90)=f( 22 00 00 )=  8001.4008192775
 f( 95)=f( 20 20 00 )=  4947.4129136183
 f( 96)=f( 20 11 00 )=  10437.908206357
 f( 99)=f( 20 02 00 )= -12671.892594229
 f(104)=f( 20 00 02 )=  14177.804650681
 f(110)=f( 11 20 00 )= -10437.908206357
 f(111)=f( 11 11 00 )=  35238.611015694
 f(114)=f( 11 02 00 )=  10437.908206357
```

```
 f(140)=f( 04 00 00 )=  4000.7004096387
 f(145)=f( 02 20 00 )= -12671.892594229
 f(146)=f( 02 11 00 )= -10437.908206357
 f(149)=f( 02 02 00 )=  4947.4129136183
 f(154)=f( 02 00 02 )=  14177.804650681
 f(175)=f( 00 40 00 )=  2666.6235224429
 f(179)=f( 00 22 00 )=  5333.2470448859
 f(184)=f( 00 20 02 )= -4822.0487446741
 f(195)=f( 00 04 00 )=  2666.6235224429
 f(200)=f( 00 02 02 )= -4822.0487446741
 f(209)=f( 00 00 04 )=  5351.3456237838

***************************************
* Nonlinear part of cell normal form *
* map in static resonance basis:     *
***************************************

nonzero elements in generating polynomial in
the static resonance basis are :

f( 28)=f( R11001 )=  86.619829386107
f( 29)=f( R00111 )= -74.393842046268
f( 30)=f( R00003 )=  28.517013718012
f( 84)=f( R11002 )=  14177.804650681
f( 85)=f( R00112 )= -4822.0487446741
f( 86)=f( R00004 )=  5351.3456237838
f( 87)=f( R22000 )=  4000.7004096387
f( 88)=f( R00220 )=  2666.6235224429
f( 89)=f( R11110 )= -3862.2398403053
f(152)=f( R20020 )=  8809.6527539236
f(153)=f( I20020 )= -5218.9541031786

**********************************
* Response to the command aim2: *
**********************************

accept  1:      f( 28)
accept  2:      f( 29)
accept  3:      f( 30)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f( 28) =       86.6198294         0.000000000E+00
 2    f( 29) =      -74.3938420         0.000000000E+00
 3    f( 30) =       28.5170137         0.000000000E+00

***********************************
* Response to the command vary2: *
***********************************

No.  1 is sex1     sext    .  Parameter 2 out of 2 selected.
     sex1(2) =  0.10000000000000
```

```
No.  2 is sex2     sext    .  Parameter 2 out of 2 selected.
     sex2(2) =  0.20000000000000
No.  3 is sex3     sext    .  Parameter 2 out of 2 selected.
     sex3(2) =  0.30000000000000

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    sex1       sext         2      0.10000000000000
 2    sex2       sext         2      0.20000000000000
 3    sex3       sext         2      0.30000000000000

**********************************************
* Beginning of second logical fitting loop: *
**********************************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000

Iter   1 Error= 8.6620E+01,  Step= 1.0000E-04,  SubErr= 8.6620E+01 @cut=    1
-------------------------------------------------------------

******************
* ETC. ETC. ETC. *
******************

Quit on iteration   6 for reason 1: Converged: error < tolerance

Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-------------------------------------------------------
 1    f( 28) =     -5.043190184E-15     0.000000000E+00
 2    f( 29) =      1.788693421E-15     0.000000000E+00
 3    f( 30) =     -1.520836376E-15     0.000000000E+00

 New values for parameters:
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   sex1       sext       2      0.11389741911436
 2   sex2       sext       2     -0.13802079399748
 3   sex3       sext       2      1.44223295333892E-02

 Maximum error is     5.043190E-15
 Maximum allowed was  2.000000E-12

***********************************************
* Nonlinear part of cell normal form map in   *
* static resonance basis after sextupole fit: *
***********************************************
```

```
nonzero elements in generating polynomial in
the static resonance basis are :

f( 84)=f( R11002 )= -1637.6733025941
f( 85)=f( R00112 )=  1333.0662102413
f( 86)=f( R00004 )= -288.28675838792
f( 87)=f( R22000 )= -390.70283844555
f( 88)=f( R00220 )= -494.77753816978
f( 89)=f( R11110 )=  1183.8890238353
f(152)=f( R20020 )=  562.05343052423
f(153)=f( I20020 )= -145.31894585022

**********************************
* Response to the command aim3: *
**********************************

accept  1:      f( 84)
accept  2:      f( 85)
accept  3:      f( 86)
accept  4:      f( 87)
accept  5:      f( 88)
accept  6:      f( 89)
accept  7:      f(152)
accept  8:      f(153)

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f( 84) =      -1637.67330         0.000000000E+00
 2    f( 85) =       1333.06621         0.000000000E+00
 3    f( 86) =      -288.286758         0.000000000E+00
 4    f( 87) =      -390.702838         0.000000000E+00
 5    f( 88) =      -494.777538         0.000000000E+00
 6    f( 89) =       1183.88902         0.000000000E+00
 7    f(152) =       562.053431         0.000000000E+00
 8    f(153) =      -145.318946         0.000000000E+00\

***********************************
* Response to the command vary3: *
***********************************

No.  1 is oct1     octm    .  Parameter 2 out of 2 selected.
     oct1(2) =  0.10000000000000
No.  2 is oct2     octm    .  Parameter 2 out of 2 selected.
     oct2(2) =  0.20000000000000
No.  3 is oct3     octm    .  Parameter 2 out of 2 selected.
     oct3(2) =  0.30000000000000
No.  4 is oct4     octm    .  Parameter 2 out of 2 selected.
     oct4(2) =  0.40000000000000
No.  5 is oct5     octm    .  Parameter 2 out of 2 selected.
     oct5(2) =  0.50000000000000
No.  6 is oct6     octm    .  Parameter 2 out of 2 selected.
     oct6(2) =  0.60000000000000
```

```
No.  7 is oct7     octm    .  Parameter 2 out of 2 selected.
     oct7(2) =  0.70000000000000
No.  8 is oct8     octm    .  Parameter 2 out of 2 selected.
     oct8(2) =  0.80000000000000

  Variable #menu elements selected:
No.  Element    Type     Parameter   Present value
-------------------------------------------------------------
 1    oct1       octm         2      0.10000000000000
 2    oct2       octm         2      0.20000000000000
 3    oct3       octm         2      0.30000000000000
 4    oct4       octm         2      0.40000000000000
 5    oct5       octm         2      0.50000000000000
 6    oct6       octm         2      0.60000000000000
 7    oct7       octm         2      0.70000000000000
 8    oct8       octm         2      0.80000000000000

**********************************************
* Beginning of third logical fitting loop: *
**********************************************

 target           1 =  0.000000000000000E+000
 target           2 =  0.000000000000000E+000
 target           3 =  0.000000000000000E+000
 target           4 =  0.000000000000000E+000
 target           5 =  0.000000000000000E+000
 target           6 =  0.000000000000000E+000
 target           7 =  0.000000000000000E+000
 target           8 =  0.000000000000000E+000

Iter   1 Error= 1.6377E+03,  Step= 1.0000E-04,  SubErr= 1.6377E+03 @cut=    1
-------------------------------------------------------

*****************
* ETC. ETC. ETC. *
*****************

Quit on iteration  12 for reason 1: Converged: error < tolerance
Final values with reach =   1 are:

 Aims selected :
No.     item        present value        target value
-----------------------------------------------------
 1    f( 84) =     -4.263256415E-13     0.000000000E+00
 2    f( 85) =     -1.449507181E-12     0.000000000E+00
 3    f( 86) =     -2.317461646E-13     0.000000000E+00
 4    f( 87) =      2.531308496E-14     0.000000000E+00
 5    f( 88) =      4.041211810E-13     0.000000000E+00
 6    f( 89) =     -1.225686219E-13     0.000000000E+00
 7    f(152) =      4.325428904E-13     0.000000000E+00
 8    f(153) =     -5.719869023E-13     0.000000000E+00

 New values for parameters:
```

```
No.  Element    Type   Parameter   Present value         IDV  Slope
-----------------------------------------------------------------------
 1   oct1       octm       2       2.1166369696817
 2   oct2       octm       2      -1.5547815137167
 3   oct3       octm       2      0.86232989354332
 4   oct4       octm       2     -0.46517533926577
 5   oct5       octm       2       1.3164862148093
 6   oct6       octm       2      -2.2053747299133
 7   oct7       octm       2       2.1494886803769
 8   oct8       octm       2      -2.1371658664381

 Maximum error is     1.449507E-12
 Maximum allowed was  2.000000E-12

*****************************************************************
* Nonlinear part of cell normal form map in  static resonance   *
* basis after octupole fit showing that all entries now vanish: *
*****************************************************************

nonzero elements in generating polynomial in
the static resonance basis are :

*************************************************************
* Map for cell (in cartesian basis) after complete fitting: *
*************************************************************

matrix for map is :

-8.02720E-01  7.30350E+00  0.00000E+00  0.00000E+00  0.00000E+00 -1.85649E+00
-2.93074E-01  1.42075E+00  0.00000E+00  0.00000E+00  0.00000E+00 -5.06744E-01
 0.00000E+00  0.00000E+00  1.41606E+00  7.28257E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -2.92485E-01 -7.98022E-01  0.00000E+00  0.00000E+00
 1.37317E-01  1.06338E+00  0.00000E+00  0.00000E+00  1.00000E+00  5.74393E-01
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )= 9.36185945825059E-02
 f( 29)=f( 21 00 00 )= 0.20559950958805
 f( 33)=f( 20 00 01 )= 0.57159603176407
 f( 34)=f( 12 00 00 )= -7.7584995254453
 f( 38)=f( 11 00 01 )= -5.7937990194291
 f( 39)=f( 10 20 00 )= 0.19624372473537
 f( 40)=f( 10 11 00 )=  2.6811419005490
 f( 43)=f( 10 02 00 )= -1.4226165471601
 f( 48)=f( 10 00 02 )=-0.47893075325664
 f( 49)=f( 03 00 00 )=  21.563580204125
 f( 53)=f( 02 00 01 )=  12.352514359375
 f( 54)=f( 01 20 00 )=-0.51044283883564
 f( 55)=f( 01 11 00 )= -13.408922077615
 f( 58)=f( 01 02 00 )= 9.19161839553424E-02
 f( 63)=f( 01 00 02 )=  1.6665969855111
 f( 67)=f( 00 20 01 )=-0.25316850839476
```

```
 f( 70)=f( 00 11 01 )= -2.5880088921489
 f( 76)=f( 00 02 01 )= -7.4206772414169
 f( 83)=f( 00 00 03 )=-0.35952752890917
 f( 84)=f( 40 00 00 )= -17.012750333984
 f( 85)=f( 31 00 00 )=  420.02835844805
 f( 89)=f( 30 00 01 )=  56.910511199077
 f( 90)=f( 22 00 00 )= -3096.8881849064
 f( 94)=f( 21 00 01 )= -860.11150207998
 f( 95)=f( 20 20 00 )=  7.7142605993477
 f( 96)=f( 20 11 00 )=  539.59931059322
 f( 99)=f( 20 02 00 )=  2606.1168945382
 f(104)=f( 20 00 02 )= -58.274080316506
 f(105)=f( 13 00 00 )=  10370.088710489
 f(109)=f( 12 00 01 )=  4309.0406946965
 f(110)=f( 11 20 00 )= -322.16396422896
 f(111)=f( 11 11 00 )= -5520.8848283479
 f(114)=f( 11 02 00 )= -25794.203636156
 f(119)=f( 11 00 02 )=  577.18813818450
 f(123)=f( 10 20 01 )= -33.407671953576
 f(126)=f( 10 11 01 )= -706.44208706967
 f(132)=f( 10 02 01 )= -3421.1525139268
 f(139)=f( 10 00 03 )=  25.252689691496
 f(140)=f( 04 00 00 )= -13044.755726123
 f(144)=f( 03 00 01 )= -7161.9106618381
 f(145)=f( 02 20 00 )=  714.56769388657
 f(146)=f( 02 11 00 )=  13569.312668952
 f(149)=f( 02 02 00 )=  63698.496023237
 f(154)=f( 02 00 02 )= -1426.3323759890
 f(158)=f( 01 20 01 )=  162.09433365999
 f(161)=f( 01 11 01 )=  3413.6941029111
 f(167)=f( 01 02 01 )=  16731.326540328
 f(174)=f( 01 00 03 )= -123.88951464508
 f(175)=f( 00 40 00 )=  4.0960294742211
 f(176)=f( 00 31 00 )= -20.501161329250
 f(179)=f( 00 22 00 )= -427.69731282908
 f(184)=f( 00 20 02 )=  9.5433188776337
 f(185)=f( 00 13 00 )= -2965.8704736907
 f(190)=f( 00 11 02 )=  207.61350536025
 f(195)=f( 00 04 00 )= -8082.5727030495
 f(200)=f( 00 02 02 )=  1059.4562341778
 f(209)=f( 00 00 04 )= -4.4177803986659

lump lcell    constructed and stored.( 1)

*************************************************************
* Map for resulting "complete" achromat (in cartesian basis). *
* Note that the matrix part is the identity, and all          *
* cubic and quartic generators vanish as desired.             *
*************************************************************

matrix for map is :

 1.00000E+00 -2.04281E-14  0.00000E+00  0.00000E+00  0.00000E+00  2.88658E-15
```

```
 7.77156E-16  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.11022E-15
 0.00000E+00  0.00000E+00  1.00000E+00 -7.81597E-14  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  3.10862E-15  1.00000E+00  0.00000E+00  0.00000E+00
-5.55112E-16 -1.55431E-15  0.00000E+00  0.00000E+00  1.00000E+00 -2.66454E-15
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :


end of MARYLIE run
```

10.7 Study of Final Focus Test Beam Facility

The Stanford Linear Accelerator Center has designed, built, and performed experiments with a Final Focus Test Beam (FFTB) Facility in order to study various issues of interest for the design, construction, and operation of the proposed Next Linear Collider. The MaryLie run of this section displays the nominal behavior of the FFTB. It does little more than track an initial particle distribution through the FFTB to demonstrate that the FFTB does indeed produce a final spot of the desired shape and size. However, it could serve as the starting point for runs that would study the effects of multipole, misalignment, misplacement, and mispowering errors. It could also be used as a starting point for aberration studies and further optimization. Finally, it gives some feeling for the complexity of a large beam handling system.

The generation of the initial conditions for the initial particle distribution itself deserves comment. This is accomplished by the commands in the line *makeic*. See Exhibit 10.7a. Suppose the beam at the end of the linac that feeds the FFTB Facility is described (as it enters the line *ltff*) in transverse phase space by the distribution

$$h(X, P_x, Y, P_y) = \text{cnst} \ \times \exp[(\beta_x P_x^2 + \gamma_x X^2)/(2\sigma_x^2) + (\beta_y P_y^2 + \gamma_y Y^2)/(2\sigma_y^2)]. \tag{10.7.1}$$

Here the *beam* twiss parameters are taken to have the values

$$\beta_x = 49.288 \text{ m},$$

$$\beta_y = 22.753 \text{ m}. \tag{10.7.2}$$

(Note that in this case "twiss parameters" are used to describe the incoming beam ellipsoid, and not the beam transport system.) With regard to energy, it is assumed that the distribution is uniform within a $\pm 0.4\%$ interval, and zero outside this interval. Finally the multiplicative normalization term (cnst) is selected to normalize the integral of h to unity.

For this distribution one has the relations

$$\begin{aligned} \langle X^2 \rangle &= \beta_x \sigma_x^2, \\ \langle P_x^2 \rangle &= \gamma_x \sigma_x^2, \\ \langle Y^2 \rangle &= \beta_y \sigma_y^2, \\ \langle P_y^2 \rangle &= \gamma_y \sigma_y^2. \end{aligned} \tag{10.7.3}$$

Correspondingly, the mean square emittances defined in terms of moments are given by the relations

$$\epsilon_x^2 = \langle X^2 \rangle \langle P_x^2 \rangle = \sigma_x^4,$$
$$\epsilon_y^2 = \langle Y^2 \rangle \langle P_y^2 \rangle = \sigma_y^4. \tag{10.7.4}$$

The emittances are taken to have the values

$$\epsilon_x = \sigma_x^2 = 3 \times 10^{-10} \text{ m rad},$$
$$\epsilon_y = \sigma_y^2 = 3 \times 10^{-11} \text{ m rad}, \tag{10.7.5}$$

Correspondingly X and Y have the rms values

$$X_{\text{rms}} = \sigma_x \sqrt{\beta_x} = 1.2 \times 10^{-4} \text{ m},$$
$$Y_{\text{rms}} = \sigma_y \sqrt{\beta_y} = 2.6 \times 10^{-5} \text{ m}. \tag{10.7.6}$$

The distribution just described could be simulated using the command *bgen* with JOB=5. See section 8.34. However, to visualize system performance, it is also useful to employ a distribution in which particles are placed uniformly on *matched* ellipses in such a way that $X_{\max}$ and $Y_{\max}$ take on the values X_{rms} and Y_{rms}, respectively. This can be accomplished by using the command *bgen* with JOB=11. With regard to energy, it is convenient to give all particles the P_τ values 0 or $\pm.004$.

The distribution described above is constructed in several steps (look again at the line *makeic* in Exhibit 10.7a):

1. Use *bgen* with JOB=11 and IPSET=1, and give P1 and P2 the values

$$P1 = (1/2)(3 \times 10^{-10}),$$
$$P2 = (1/2)(3 \times 10^{-11}). \tag{10.7.7}$$

So doing produces a particle set in the *zblock* array that is uniform on a 2-torus and has the properties

$$\langle X^2 \rangle = \langle P_x^2 \rangle = 1.5 \times 10^{-10},$$
$$\langle Y^2 \rangle = \langle P_y^2 \rangle = 1.5 \times 10^{-11}. \tag{10.7.8}$$

(See the response to the line *dist* in Exhibit 10.7.) Correspondingly, because this distribution is uniform on (the surface of) a 2-torus, it has the property

$$(X_{\max})^2 = 2\langle X^2 \rangle = 3 \times 10^{-10},$$
$$(Y_{\max})^2 = 2\langle Y^2 \rangle = 3 \times 10^{-11}. \tag{10.7.9}$$

2. Construct a 6×6 matrix having the requisite twiss parameters using the command *twsm*. See the line *tws* in Exhibit 10.7a and section 6.15. The choice of phase advances in the X and Y planes is immaterial except that they should be different and not add to 180° (to avoid MaryLie complaining about resonances) and not equal to 0° or 180°.

3. Use the static normal form routine *snor* and a *gbuf* command to get the normalizing map $\mathcal{A}$ for the map produced by *tws*. See sections 8.8 and 8.14. The map $\mathcal{A}$ converts a perfect torus into a matched ellipse. In this case (since $\alpha_x = \alpha_y = 0$) its matrix part is diagonal with entries $\sqrt{\beta_x}$, $\sqrt{\gamma_x}$, $\sqrt{\beta_y}$, $\sqrt{\gamma_y}$, 1, 1. See Exhibit 10.7a where $\mathcal{A}$ is displayed as a result of the *mapout* command in the line *makeic*.

4. Use the ray-trace command *xform* to apply the map $\mathcal{A}$ to the contents of *zblock* and write the results to an external file (in this case file 14). See section 7.2.

5. Use the *tic* command *pt+* to shift the P_τ entries in *zblock* by .004. See section 8.35. Again use *xform* to apply $\mathcal{A}$ to the contents of *zblock* and append the results to file 14.

6. Use the *tic* command *pt−* to shift the values of P_τ in *zblock* by −.008 so that they now have the value −.004. Again use *xform* to apply $\mathcal{A}$ to the contents of *zblock* and append the results to file 14. Figures 10.7.1 and 10.7.2 show the X, P_x, Y, P_y contents of file 14 as a result of all these operations. Note that the distributions are indeed ellipsoidal (matched) and have the proper values for $X_{\max}$, $Y_{\max}$ as desired.

7. Use the ray trace command *raysin* to read the contents of file 14 into the *zblock* array. As a result, *zblock* now contains the desired initial conditions for a matched input beam.

As a side comment at this point, we remark that the procedure just described for producing a matched beam is convenient if there is no coupling between planes, and has the advantage that all parameters used are accessible to *vary* or *scan* commands should one want to do fitting/optimization, etc. However, an alternate procedure involving a *moma* command (see section 8.37) is also available, and can be used in the general case. This procedure begins with a knowledge of the desired quadratic moments of the particle distribution, and replaces steps 1 through 4 above by the following:

1′. Place the desired second moments in an external file, say file 9, and analyze these moments using a *moma* command. As a result of this analysis, the matrix part of buffer 1 contains a matrix that, as a map, sends eigenmoments to the original moments; and the array part contains the eigenmoments.

2′. Use a *gbuf* command to replace the current map (if any) with the contents of buffer 1.

3′. Use a *bgen* command with P1 < 0 for the parameter set IPSET. See section 8.34. With P1 < 0, *bgen* will use the eigenmoments produced by *moma* to generate a particle set.

4′. Use a *mask* command that keeps the matrix part of the map, but removes the array part. Call the resulting map $\mathcal{A}$. Use the ray-trace command *xform* to apply $\mathcal{A}$ to the contents of *zblock* and write the results to an external file (in this case file 14). See section 7.2.

Exhibits 10.7b and 10.7c show a sample file 9 and a sample MaryLie run that generates a matched beam in this fashion. Note that the moments in file 9 are half those given by (10.14) in order to achieve (10.19) as before.

To continue the main discussion, the rest of the MaryLie run in Exhibit 10.7a is devoted to transporting the incoming beam to the end of the FFTB to verify that a suitably small and chromatic-aberration-corrected spot is indeed produced. First the net transfer for the entire system is produced through third order and exhibited. Next, this map is applied to the initial conditions of figures 10.7.1 and 10.7.2 to produce the final conditions shown in figures 10.7.3 and 10.7.4. Chromatic effects are apparent since there are three distinct patterns corresponding to the three different values of P_τ. However, the chromatic effects are not major (that is, chromatic aberrations are well corrected, especially the vertical ones that are potentially more detrimental) because the horizontal and vertical spot sizes are 3.5 μm and 100 nm, respectively. Finally, as an estimate of the effect of still higher order aberrations, the initial conditions are traced element-by-element to yield the results shown in figures 10.7.5 and 10.7.6. Higher-order aberration effects are noticeable, particularly in the vertical projection. However, again they do not seriously enlarge the final spot size.

Figure 10.7.1: Horizontal projection of phase-space initial conditions data (taken from file 14) as a result of invoking the line *makeic* in *#labor*.

Figure 10.7.2: Vertical projection of phase-space initial conditions data (taken from file 14) as a result of invoking the line *makeic* in *#labor*.

Figure 10.7.3: Horizontal projection of phase-space final conditions data computed using the net transfer map for the entire system.

Figure 10.7.4: Vertical projection of phase-space final conditions data computed using the net transfer map for the entire system.

Figure 10.7.5: Horizontal projection of phase-space final conditions data computed using element-by-element tracking.

Figure 10.7.6: Vertical projection of phase-space final conditions data computed using element-by-element tracking.

```
Exhibit 10.7a
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
Exhibit 10.7.1.
     This is a study of the SLAC FFTB facility.  This run does 4
things:

 a) It generates the initial conditions for an input beam.
    This is done by invoking the line makeic in #labor.

 b) It computes the transfer map for the whole system.

 c) It applies this map to the input beam to generate an
    output beam.

 d) To provide an extimate of the effect of higher-order
    aberrations, it also carries out an element-by-element
    ray trace.

The #beam parameters are those for 50 GeV electrons.

#beam
  166.782047599076
  97845.7070864890
  1.00000000000000
  1.00000000000000
#menu
 bgen     bgen
   8.00000000000000       123.000000000000       2304.00000000000
   123.000000000000       1.00000000000000       1.00000000000000
 mom      ps1
  1.500000000000000E-10  1.500000000000000E-11  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 pt+      tic
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  4.000000000000000E-03
 pt-      tic
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00 -8.000000000000000E-03
 twx      twsm
   1.00000000000000       290.000000000000      0.000000000000000E+00
   49.2880000000000
 twy      twsm
   2.00000000000000       31.0000000000000      0.000000000000000E+00
   22.7530000000000
 snor     snor
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00
```

```
gbuf1    gbuf
  2.00000000000000       1.00000000000000
xform    rt
 0.000000000000000E+00   14.0000000000000       3.00000000000000
 0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
raysin   rt
  14.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
clear    iden
trace16  rt
 0.000000000000000E+00   16.0000000000000       3.00000000000000
 0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
circ18   circ
 0.000000000000000E+00   18.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000       3.00000000000000
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
q81      quad
 7.000000000000001E-02   76.4285271317830       1.00000000000000
  1.00000000000000
dr01     drft
  13.2483800000000
q1       quad
 9.367000000000000E-02  -62.0893946731235       1.00000000000000
  1.00000000000000
dr02     drft
 0.140050000000000
bpm1     drft
 0.000000000000000E+00
dr03     drft
  1.62400000000000
s100     drft
 0.000000000000000E+00
dr04     drft
  2.68800000000000
a1x      drft
 0.000000000000000E+00
dr05     drft
 0.351000000000000
a1y      drft
 0.000000000000000E+00
dr06     drft
  4.33553000000000
b40b1    drft
  1.98120000000000
dr07     drft
  2.09207000000000
a2x      drft
 0.000000000000000E+00
dr08     drft
```

```
 0.351000000000000
a2y      drft
 0.000000000000000E+00
dr09     drft
 0.382760000000000
q2       quad
 0.162155000000000       62.0893946731235       1.00000000000000
  1.00000000000000
dr10     drft
 0.127930000000000
bpm2     drft
 0.000000000000000E+00
dr11     drft
  3.21300000000000
a3x      drft
 0.000000000000000E+00
dr12     drft
 0.351000000000000
a3y      drft
 0.000000000000000E+00
dr13     drft
 0.487100000000000
q3       quad
 0.143255000000000      -62.0893946731235       1.00000000000000
  1.00000000000000
dr14     drft
 0.204390000000000
bpm3     drft
 0.000000000000000E+00
dr15     drft
 0.580977000000000
b1       drft
  1.13202700000000
dr16     drft
 0.761970000000000
pr2      drft
 0.000000000000000E+00
dr17     drft
  5.08621300000000
dr18     drft
  21.8169800000000
qa01     drft
 5.075000000000000E-02
dr19     drft
 0.406300000000000
qa02     drft
 5.075000000000000E-02
dr20     drft
 0.562626000000000
pmv      drft
 0.000000000000000E+00
dr21     drft
  1.09849920000000
```

```
pm1      drft
 0.000000000000000E+00
dr22     drft
  4.90240320000000
pm3      drft
 0.000000000000000E+00
dr23     drft
  4.90198160000000
b2       drft
 0.000000000000000E+00
dr24     drft
  65.8979512800000
d10      drft
  4.26085000000000
lx0      drft
 0.200000000000000
lx1      drft
 0.300000000000000
lx5      drft
 0.500000000000000
dm0      drft
  7.70439912000000
dm1      drft
  3.53897240000000
dm2      drft
 0.561408100000000
dm3      drft
 0.152400000000000
dm4      drft
  9.21146240000000
dm5      drft
 0.561378900000000
dm6      drft
 0.567588400000000
swall    drft
  16.7640000000000
dm7      drft
 0.203200000000000
dm8      drft
  1.53908000000000
d2       drft
 0.609600000000000
st60     drft
 0.609600000000000
st61     drft
 0.609600000000000
dmoni    drft
 9.525000000000000E-03
q5       quad
 0.230460000000000      -19.2771304347826       1.00000000000000
  1.00000000000000
q6       quad
 0.230460000000000       35.4012173913043       1.00000000000000
```

```
  1.00000000000000
qa0      quad
 0.230460000000000      -24.8659130434783       1.00000000000000
  1.00000000000000
qa1      quad
 0.230460000000000      -86.9565217391304       1.00000000000000
  1.00000000000000
qa2      quad
 0.230460000000000       31.2483478260870       1.00000000000000
  1.00000000000000
dws1     drft
 0.152400000000000
dws2     drft
 0.304800000000000
ws       drft
 0.104775000000000
ws1      mark
mccsx    mark
b01      pbnd
 0.216215560584730      0.000000000000000E+00  0.500000000000000
 0.239947000000000
la1      drft
  1.00000000000000
la2      drft
  4.95822000000000
ln1      drft
  5.69811000000000
ln2      drft
 0.439340000000000
qn1      quad
 0.230460000000000       75.9993043478261       1.00000000000000
  1.00000000000000
qn2      quad
 0.230460000000000      -77.6042608695652       1.00000000000000
  1.00000000000000
qn3      quad
 0.230460000000000       40.9167826086957       1.00000000000000
  1.00000000000000
sf1      sext
 6.250000000000000E-02   5703.26378000000
lt1      drft
  3.00000000000000
lt2      drft
  7.41409000000000
lt3      drft
  7.54059000000000
qt1      quad
 0.230460000000000       74.5672173913044       1.00000000000000
  1.00000000000000
qt2      quad
 0.230460000000000      -62.5448695652174       1.00000000000000
  1.00000000000000
qt3      quad
```

```
 0.230460000000000       80.5043478260870       1.00000000000000
  1.00000000000000
qt4      quad
 0.230460000000000      -36.8060869565217       1.00000000000000
  1.00000000000000
b02      pbnd
-0.216215560584730      0.000000000000000E+00  0.500000000000000
-0.239947000000000
qm1      quad
 0.230460000000000       77.6042608695652       1.00000000000000
  1.00000000000000
qm2      quad
 0.230460000000000      -75.9993043478261       1.00000000000000
  1.00000000000000
qm3      quad
 0.230460000000000      -40.9167826086957       1.00000000000000
  1.00000000000000
sd1      sext
 6.250000000000000E-02   6069.09800000000
lb2      drft
  1.26077000000000
vcor     drft
 5.000000000000000E-02
lb3      drft
  8.63923000000000
lb1      drft
  2.50612000000000
mff      mark
pmip     drft
 0.000000000000000E+00
b03      pbnd
-2.758937360080000E-02  0.000000000000000E+00  0.500000000000000
-3.047810000000000E-02
lb0      drft
  4.96078000000000
hcor     drft
 5.000000000000000E-02
lb00     drft
 0.168610000000000
lc4      drft
  10.3637400000000
lc3      drft
  4.87095000000000
lc2      drft
 0.889270000000000
lxc      drft
 0.150000000000000
lstar    drft
 0.400000000000000
qc5      quad
 0.230460000000000      -65.8373913043478       1.00000000000000
  1.00000000000000
qc4      quad
```

```
 0.230460000000000       4.94034782608696       1.00000000000000
  1.00000000000000
qc3      quad
 0.233500000000000      -44.7427428571429       1.00000000000000
  1.00000000000000
qc2      quad
  1.00000000000000       40.4418148148148       1.00000000000000
  1.00000000000000
qx1      quad
 0.150000000000000      -140.000000000000       1.00000000000000
  1.00000000000000
qc1      quad
 0.550000000000000      -175.384615384615       1.00000000000000
  1.00000000000000
qs1      quad
 0.230460000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
qs2      quad
 0.230460000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
ftp      mark
end      end
#lines
dist
    1*mom         1*bgen
tws
    1*clear       1*twx         1*twy
makeic
    1*dist        1*tws         1*snor        1*gbuf1       1*mapout   &
    1*xform       1*pt+         1*xform       1*pt-         1*xform    &
    1*raysin      1*clear
ltff
    1*q81         1*dr01        2*q1          1*dr02        1*bpm1     &
    1*dr03        1*s100        1*dr04        1*a1x         1*dr05     &
    1*a1y         1*dr06        2*b40b1       1*dr07        1*a2x      &
    1*dr08        1*a2y         1*dr09        2*q2          1*dr10     &
    1*bpm2        1*dr11        1*a3x         1*dr12        1*a3y      &
    1*dr13        2*q3          1*dr14        1*bpm3        1*dr15     &
    2*b1          1*dr16        1*pr2         1*dr17        1*dr18     &
    2*qa01        1*dr19        2*qa02        1*dr20        1*pmv      &
    1*dr21        1*pm1         1*dr22        1*pm3         1*dr23     &
    1*b2          1*dr24        1*d10
pre
    1*dm0         2*q5          1*dm1         2*q6          1*dm2      &
    2*d2          1*dm3         2*st60        1*dm4         2*st61     &
    1*dm5         2*qa0         1*dm6         2*dmoni       1*swall    &
    1*dm7         2*qa1         1*dm8         2*qa2         1*dws1     &
    1*ws          1*ws1         1*ws          1*dws2
b0b
    1*b02         1*lx1         1*b02
b0a
    1*b01         1*lx1         1*b01
itrm
```

```
     2*qt1         1*lt1         2*qt2         1*lx0         2*qt2      &
     1*lt2         2*qt3         1*lt3         2*qt4
 xcc
     1*lx0         4*sf1         1*lx0         2*qn3         1*ln1      &
     2*qn2         1*ln2         1*b0a         1*lx5
 ccsx
     1*b0a         1*la1         2*qn2         1*la2         2*qn3      &
     1*xcc         2*qn1        -1*xcc
 ycc2
     2*sd1         1*lx0         2*qm3         1*ln1         2*qm1      &
     1*ln2         1*b0b         1*lx5
 ccsy
     1*lx0         2*sd1         1*ycc2        2*qm2        -1*ycc2     &
     2*sd1         1*lx0         2*qm3         1*lb3         2*vcor     &
     1*lb2         2*qm1         1*lb1
 sb
     1*b03         1*lx1         1*b03
 ft
     2*qs1         1*lx0         2*qc5         1*lb0         2*hcor     &
     1*lb00        1*b0b         1*lx1         1*sb          1*lx1      &
     2*qc4         1*lc4         2*qs2         1*lx0         2*qc3      &
     1*lc3         1*qc2         1*qc2         1*lc2         2*qx1      &
     1*lxc         2*qc1         1*lstar       1*pmip
 fftb
     1*pre         1*ccsx        1*itrm        1*ccsy        1*ft
 whole
     1*ltff        1*fftb
#lumps
#loops
 onebyone
     1*whole
#labor
    1*fileout
    1*makeic
    1*whole
    1*mapout
    1*trace16
    1*clear
    1*onebyone
    1*circ18
    1*end

****************************************************
* Response to the command bgen in the line makeic: *
****************************************************

       2304 rays generated

 numerically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 1.500000000000011E-010 -2.664596835413195E-026  1.500000000000018E-010
 values of <y*y>, <y*py>, <py*py>:
 1.499999999999989E-011  5.402820688647018E-028  1.499999999999990E-011
```

```
 values of <t*t>, <t*pt>, <pt*pt>:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

 analytically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 1.500000000000000E-010  0.000000000000000E+000  1.500000000000000E-010
 values of <y*y>, <y*py>, <py*py>:
 1.500000000000000E-011  0.000000000000000E+000  1.500000000000000E-011
 values of <t*t>, <t*pt>, <pt*pt>:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

******************************
* The matching map script A: *
******************************

matrix for map is :

 7.02054E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.42439E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.77001E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  2.09643E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

****************************
* Reading in matched beam: *
****************************

 6912 ray(s) read in from file  14

***********************************
* Transfer map for whole system: *
***********************************

matrix for map is :

 1.16042E-03  3.80255E-01  0.00000E+00  0.00000E+00  0.00000E+00 -6.52656E-07
-2.57205E+00  1.89289E+01  0.00000E+00  0.00000E+00  0.00000E+00  5.22049E-07
 0.00000E+00  0.00000E+00 -7.92903E-04  4.41560E-02  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.94231E+01 -1.79537E+02  0.00000E+00  0.00000E+00
 1.67806E-06 -1.25526E-05  0.00000E+00  0.00000E+00  1.00000E+00 -1.17060E-03
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=-3.89227825816134E-02
 f( 29)=f( 21 00 00 )= 8.82503434730550E-02
 f( 33)=f( 20 00 01 )= -913.33367251839
 f( 34)=f( 12 00 00 )=-6.28096901842339E-02
 f( 38)=f( 11 00 01 )=  13.817825821166
 f( 39)=f( 10 20 00 )= 5.30573201601003E-02
```

```
 f( 40)=f( 10 11 00 )= 0.11077558702035
 f( 43)=f( 10 02 00 )= 3.51466442021233E-03
 f( 48)=f( 10 00 02 )= 0.45425928722257
 f( 49)=f( 03 00 00 )=-1.15123554435159E-04
 f( 53)=f( 02 00 01 )= 0.28123701550681
 f( 54)=f( 01 20 00 )=-0.54577103156849
 f( 55)=f( 01 11 00 )=-7.11881814520230E-02
 f( 58)=f( 01 02 00 )= 4.16651091578457E-05
 f( 63)=f( 01 00 02 )= 2.10706864800540E-02
 f( 67)=f( 00 20 01 )= -7607.4908656048
 f( 70)=f( 00 11 01 )=  4.0703495464613
 f( 76)=f( 00 02 01 )= 3.78505599207130E-03
 f( 83)=f( 00 00 03 )= 7.16372329713475E-04
 f( 84)=f( 40 00 00 )= -1509577.7880829
 f( 85)=f( 31 00 00 )=  31525.584758041
 f( 89)=f( 30 00 01 )=  1051.3749913900
 f( 90)=f( 22 00 00 )= -102.70479343622
 f( 94)=f( 21 00 01 )= -3117.6407533003
 f( 95)=f( 20 20 00 )= -41749615.002048
 f( 96)=f( 20 11 00 )= -823.45497555658
 f( 99)=f( 20 02 00 )= -12.024976325221
 f(104)=f( 20 00 02 )= -8215.8059409657
 f(105)=f( 13 00 00 )= -548.24879728467
 f(109)=f( 12 00 01 )=  2317.1974918641
 f(110)=f( 11 20 00 )=  166247.09590690
 f(111)=f( 11 11 00 )=  67.126217696816
 f(114)=f( 11 02 00 )= -7.8306638896465
 f(119)=f( 11 00 02 )=  593.63080145352
 f(123)=f( 10 20 01 )=  3977.3635330146
 f(126)=f( 10 11 01 )= -612.91427242861
 f(132)=f( 10 02 01 )= -173.87987518916
 f(139)=f( 10 00 03 )= -8.5049112006986
 f(140)=f( 04 00 00 )=  200.25538481841
 f(144)=f( 03 00 01 )= 0.89850794460257
 f(145)=f( 02 20 00 )= -249.70141545497
 f(146)=f( 02 11 00 )=  139.41365645640
 f(149)=f( 02 02 00 )=  5.9698255205061
 f(154)=f( 02 00 02 )=  5.2855306317948
 f(158)=f( 01 20 01 )= -18326.626444905
 f(161)=f( 01 11 01 )= -2655.4851662894
 f(167)=f( 01 02 01 )=-6.39318206821628E-02
 f(174)=f( 01 00 03 )= -10.589966794531
 f(175)=f( 00 40 00 )= -95434657.276104
 f(176)=f( 00 31 00 )=  30603.630699873
 f(179)=f( 00 22 00 )=  1516.9195509404
 f(184)=f( 00 20 02 )= -41485.878055000
 f(185)=f( 00 13 00 )=  147.03874397650
 f(190)=f( 00 11 02 )= -34.314062493911
 f(195)=f( 00 04 00 )=  5.0709745286033
 f(200)=f( 00 02 02 )= 1.72432221019481E-05
 f(209)=f( 00 00 04 )=-2.45235185866578E-02

*****************************************************
```

```
* Elements traversed in element-by-element ray trace: *
*******************************************************

circulating through onebyone :
 q81      dr01     q1       q1       dr02
 bpm1     dr03     s100     dr04     a1x
 dr05     a1y      dr06     b40b1    b40b1
 dr07     a2x      dr08     a2y      dr09
 q2       q2       dr10     bpm2     dr11
 a3x      dr12     a3y      dr13     q3
 q3       dr14     bpm3     dr15     b1
 b1       dr16     pr2      dr17     dr18
 qa01     qa01     dr19     qa02     qa02
 dr20     pmv      dr21     pm1      dr22
 pm3      dr23     b2       dr24     d10
 dm0      q5       q5       dm1      q6
 q6       dm2      d2       d2       dm3
 st60     st60     dm4      st61     st61
 dm5      qa0      qa0      dm6      dmoni
 dmoni    swall    dm7      qa1      qa1
 dm8      qa2      qa2      dws1     ws
 ws1      ws       dws2     b01      lx1
 b01      la1      qn2      qn2      la2
 qn3      qn3      lx0      sf1      sf1
 sf1      sf1      lx0      qn3      qn3
 ln1      qn2      qn2      ln2      b01
 lx1      b01      lx5      qn1      qn1
 lx5      b01      lx1      b01      ln2
 qn2      qn2      ln1      qn3      qn3
 lx0      sf1      sf1      sf1      sf1
 lx0      qt1      qt1      lt1      qt2
 qt2      lx0      qt2      qt2      lt2
 qt3      qt3      lt3      qt4      qt4
 lx0      sd1      sd1      sd1      sd1
 lx0      qm3      qm3      ln1      qm1
 qm1      ln2      b02      lx1      b02
 lx5      qm2      qm2      lx5      b02
 lx1      b02      ln2      qm1      qm1
 ln1      qm3      qm3      lx0      sd1
 sd1      sd1      sd1      lx0      qm3
 qm3      lb3      vcor     vcor     lb2
 qm1      qm1      lb1      qs1      qs1
 lx0      qc5      qc5      lb0      hcor
 hcor     lb00     b02      lx1      b02
 lx1      b03      lx1      b03      lx1
 qc4      qc4      lc4      qs2      qs2
 lx0      qc3      qc3      lc3      qc2
 qc2      lc2      qx1      qx1      lxc
 qc1      qc1      lstar    pmip

end of MARYLIE run
```

```
Exhibit 10.7b
Contents of file 9 that provides desired second
moments for the command moma:

6 6  0
7     .73932e-8
13   3.04333712059731e-12
18   3.41295e-10
22    .659253724783545e-12
209  0


Exhibit 10.7c
***MARYLIE 3.0***
Prerelease Development Version 1/12/00
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
Exhibit 10.7c
     This MaryLie run exhibits the use of the moma command
to generate a matched beam given a set of moments.

Although immaterial for this purpose, the #beam parameters are
those for 50 GeV electrons.

#beam
  166.782047599076
  97845.7070864890
  1.00000000000000
  1.00000000000000
#menu
 moma     moma
   21.0000000000000       1.00000000000000       9.00000000000000
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
 bgen     bgen
   8.00000000000000       123.000000000000       2304.00000000000
   123.000000000000       1.00000000000000       1.00000000000000
 mom      ps1
  -1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 pt+      tic
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  4.000000000000000E-03
 pt-      tic
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00 -8.000000000000000E-03
 gbuf1    gbuf
   2.00000000000000       1.00000000000000
 mask     mask
   1.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
```

```
 xform    rt
  0.000000000000000E+00   14.0000000000000       3.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 raysin   rt
   14.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 clear    iden
 trace16  rt
  0.000000000000000E+00   16.0000000000000       3.00000000000000
  0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 circ18   circ
  0.000000000000000E+00   18.0000000000000       5.00000000000000
   1.00000000000000       1.00000000000000       3.00000000000000
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 end      end
#lines
 dist
     1*mom         1*bgen
 makeic
     1*moma        1*gbuf1       1*mapout      1*dist        1*mask     &
     1*mapout      1*xform       1*pt+         1*xform       1*pt-      &
     1*xform       1*raysin      1*clear
#lumps
#loops
#labor
    1*fileout
    1*makeic
    1*end

****************************************************
* Response to the moma command in the line makeic: *
****************************************************

file unit  9 rewound
map read in from file   9;   0 record(s) skipped
xee2=  2.250000000000006E-020
yee2=  2.250000000000000E-022
tee2=  0.000000000000000E+000

*******************************************
* Matrix part of the map produced by moma: *
*******************************************

matrix for map is :

 7.02054E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.42439E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.77001E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  2.09643E-01  0.00000E+00  0.00000E+00
```

```
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

*******************************
* Eigenmoments found by moma: *
*******************************

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 1.50000000000000E-10
 f( 13)=f( 02 00 00 )= 1.50000000000000E-10
 f( 18)=f( 00 20 00 )= 1.50000000000000E-11
 f( 22)=f( 00 02 00 )= 1.50000000000000E-11

***************************************************
* Response to the command bgen in the line dist: *
***************************************************

       2304 rays generated

 numerically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 1.500000000000022E-010 -3.136370717220037E-026  1.500000000000018E-010
 values of <y*y>, <y*py>, <py*py>:
 1.499999999999989E-011  5.402820688647018E-028  1.499999999999990E-011
 values of <t*t>, <t*pt>, <pt*pt>:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

 analytically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 1.500000000000002E-010  0.000000000000000E+000  1.500000000000002E-010
 values of <y*y>, <y*py>, <py*py>:
 1.500000000000000E-011  0.000000000000000E+000  1.500000000000000E-011
 values of <t*t>, <t*pt>, <pt*pt>:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

*******************************************************
* Map (script A) used to transform the distribution: *
*******************************************************

matrix for map is :

 7.02054E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.42439E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.77001E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  2.09643E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

*********************************************
* Reading in the transformed distribution: *
```

```
*******************************************

 6912 ray(s) read in from file  14

end of MARYLIE run
```

10.8 Calculation of Dynamic Aperture for a Storage Ring

Accelerator codes that do not use maps typically determine the dynamic aperture using element-by-element tracking. Elements such as drifts, bends, and quads are treated in linear approximation using matrices, and nonlinear elements are treated as integrated multipole kicks. Sometimes even the linear effects of bends and quads are treated as kicks. High-order multipole errors in quads and dipoles are treated by slicing these elements and inserting nonlinear kicks between the slices. Sometimes many slices are used. Often, to speed up computation (but presumably at the expense of accuracy), only a few slices are used. Sometimes kicks which properly should be made along the body of dipoles (and frequently) are made instead only in quads.

Simulations of the type described above can also be done with MaryLie . One can easily write user routines (see section 6.20) that kick particles in any desired manner. Any element can be sliced at will, and a user routine can be inserted between any two slices. For convenience, all the slices and user routines can be put in a tracking loop, and this loop can be tracked repeatedly as often as desired with the use of a circ command. See sections 5.10 and 7.3.

However, wherever possible, it is desirable to make use of maps. So doing could speed up dynamic aperture calculations (which are often very lengthy) by orders of magnitude. Without any increase in computer time, one could also use maps that are more accurate than kick approximations. They could describe thick elements and could also include fringe-field effects. Finally, with maps one might hope for more insight: perhaps various lattices could be compared without the need for long-term tracking, detrimental terms in maps could be identified, and correction schemes could be devised to remove them.

It appears that the dynamic aperture for various SSC designs and the CERN Large Hadron Collider can be determined by symplectic tracking with a one-turn map. The same is believed to be true for the Stanford B-Factory rings. In the case of synchrotron light rings, whose one-turn maps may be more nonlinear, it may be necessary to use a modest number of lumps. All these applications of map methods are still somewhere in the development stage, and require maps somewhat beyond third order.

Subsection 10.8.1 describes the calculation of a point on the edge of the dynamic aperture of the Tevatron in the case that several families of sextupoles are purposely strongly powered to reduce the dynamic aperture. (Experiments of this type were done on the Tevatron to answer various questions about magnet quality requirements for the SSC.) In this circumstance the dominant nonlinearities are those produced by the sextupoles, and all other sources of nonlinearity such as multipole errors in dipoles and quadrupoles become negligible by comparison. Correspondingly, it is sufficient to use third-order maps providing the effect of the one-turn map is approximated by a sufficient number of lumps. The same is true for any ring

whose dynamic aperture is controlled primarily by sextupole and octupole nonlinearities.

Subsection 10.8.2 studies the history of a particle that is just outside the dynamic aperture, and consequently makes several turns before it is lost.

10.8.1 Dynamic Aperture of Tevatron with Strong Distortion Sextupoles

The Tevatron lattice has 6 sectors labelled A through F as one circles around the ring clockwise when seen from above. See figure 10.8.1.1. The strong "distortion" sextupoles described above are located in sectors C and F. Eight of them are placed in each sector. The beam is "observed" (a Poincare surface of section is situated) at the location of a horizontal position monitor in sector E (by marker e24).

Exhibit 10.8.1 shows the results of a MaryLie run designed to find a point on the edge of the dynamic aperture. A "linearly" matched "surface" beam (consisting of 30 particles) is generated randomly and tracked for 10,000 turns. It is found that one particle is lost in this process when the horizontal and vertical mean-square emittances have the values

$$\epsilon_x^2 = \langle X^2 \rangle \langle P_x^2 \rangle = (5.31 \times 10^{-8})^2,$$

$$\epsilon_y^2 = \langle Y^2 \rangle \langle P_y^2 \rangle = (5.31 \times 10^{-8})^2.$$

By contrast, when $\epsilon_x^2 = \epsilon_y^2 = (5.30 \times 10^{-8})^2$, no particles are lost. Thus, a point on or near the boundary of the (10,000 turn) dynamic aperture has been determined. Unlike many tracking studies, the use of a randomly generated matched surface beam explores the dynamic aperture for many different initial betatron and synchrotron phases. Figures 10.8.1.2 through 10.8.1.4 show this initial beam. Note that the 2-dimensional projections of the beam lie on ellipses in the vertical and temporal planes. However, the horizontal distribution is spread out because of dispersion at the location of the horizontal position monitor.

Exhibits 10.8.1a and 10.8.1b show the contents of the *istat* and *ihist* arrays at the end of the tracking run. The *istat* array shows that only particle number 6 is lost, and that loss occurred on turn 427. The *ihist* array shows that the first (and only) particle to be lost was lost on turn 427, and that particle is the 6th one in the initial distribution. Figures 10.8.1.5 through 10.8.1.7 show 2-dimensional projections of the beam during the course of 10,000 turns with results written out every 43rd turn. The effect of dispersion in the horizontal projection is again evident. Also, the temporal projection exhibits synchro-betatron coupling. Finally, Exhibit 10.8.1c shows the MaryLie run itself.

Figure 10.8.1.1: Schematic layout of the Tevatron lattice.

Figure 10.8.1.2: Horizontal projection of phase-space initial conditions (file 16).

Figure 10.8.1.3: Vertical projection of phase-space initial conditions (file 16).

Figure 10.8.1.4: Temporal projection of phase-space initial conditions (file 16).

Figure 10.8.1.5: Horizontal projection of phase-space tracking data (file 14).

Figure 10.8.1.6: Vertical projection of phase-space tracking data (file 14).

Figure 10.8.1.7: Temporal projection of phase-space tracking data (file 14).

```
Exhibit 10.8.1a
Contents of file 20, the array istat:

         1         0
         2         0
         3         0
         4         0
         5         0
         6       427
         7         0
         8         0
         9         0
        10         0
        11         0
        12         0
        13         0
        14         0
        15         0
        16         0
        17         0
        18         0
        19         0
        20         0
        21         0
        22         0
        23         0
        24         0
        25         0
        26         0
        27         0
        28         0
        29         0
        30         0


Exhibit 10.8.1b
Contents of file 22, the array ihist:

         1       427         6


Exhibit 10.8.1c
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
Exhibit 10.8.1.
    This is a MARYLIE run to study the dynamic aperture of the Tevatron when
16 distortion sextupoles are powered.  The 'quads' q0c1 and q0c2 are set to
achieve horizontal and vertical tunes of .39 and .46, respectively, and the
```

```
'sextupoles' s0c1 and s0c2 are set to achieve zero first-order chromaticities.
Both these fits were made with the RF cavity (rfca) turned off.  In this run
the RF cavity is powered.  As one can see from the response to the tadm
command, the transverse tunes shift slightly when the cavity is turned on due
both to synchro-betatron coupling and the defocussing effects of the cavity.
The distortion sextupoles dsex+ and dsex- are set to strengths corresponding
to 30 amperes  excitation current.  Fringe-field effects are included.
Finally, the beam parameters are those for 150 GeV protons.

This MaryLie run does 10 things:

 a) It computes maps for the lines cxfd and cxdf, which occur
    repeatedly in the Tevatron lattice, and stores them as lumps
    in order to speed up subsequent computations.

 b) It computes 17 transfer maps and writes them out sequentially on file 10.
    The 1st transfer map describes the elements stretching from the location
    of a  horizontal beam position monitor in sector E (location e24) to the
    trailing end of the 1st distortion sextupole.  The 2nd transfer map
    describes the elements from the trailing end of the 1st distortion
    sextupole location to the trailing end of the 2nd distortion
    sextupole. ... The 16th transfer map describes the elements from the
    trailing end of the 15th distortion sextupole location to the trailing
    end of the 16th distortion sextupole.  Finally, the 17th transfer map
    describes the elements from the trailing end of the 16th distortion
    sextupole location back to the horizontal position monitor location.
    Thus, when combined, these 17 transfer maps give a full one-turn map.

 c) It reads these maps back in and stores them as lumps.

 d) These 17 maps are concatenated to give a one-turn map.

 e) A tadm command is used to compute script A and the tunes
    for the one-turn map.

 f) A random uniform distribution of rays is generated on the
    3-dimensional surface of a perfect torus in 6 dimensions.

 g) The "linear" part of script A is gotten from buffer1 and applied
    to these rays to make a "linearly" matched beam.

 h) This matched beam is written in "ordinary" precision on files
    14 and 16, and in full precision on file 18.

 i) This matched beam is tracked lump-by-lump for 10,000 turns
    with results written to file 14 every 43rd turn.

 j) The istat and ihist arrays are written on files 20 and 22,
    respectively.

#beam
  503.466178765432
  159.867058817009
```

```
  1.00000000000000
  1.00000000000000
#menu
 bh1      nbnd
  0.465116280536985      0.000000000000000E+00  0.500000000000000
  0.667664755182949       1.00000000000000       1.00000000000000
 bbs      nbnd
  0.222087819188675      0.000000000000000E+00  0.500000000000000
  0.667656760387843       1.00000000000000       1.00000000000000
 bl       nbnd
  0.105114881640705      0.000000000000000E+00  0.500000000000000
  0.163191579283132       1.00000000000000       1.00000000000000
 bc       nbnd
  8.535352727179631E-02  0.000000000000000E+00  0.500000000000000
  0.198175132463878       1.00000000000000       1.00000000000000
 bbpl     nbnd
  0.180481821924914      0.000000000000000E+00  0.500000000000000
  0.261245920741005       1.00000000000000       1.00000000000000
 bbmi     nbnd
  0.180481821924914      0.000000000000000E+00  0.500000000000000
  0.261245920741005       1.00000000000000       1.00000000000000
 bbmi002z arot
   180.000000000000
 bbmi003z arot
  -180.000000000000
 qf1      quad
  0.839470000000000       11.4651702288463       1.00000000000000
   1.00000000000000
 qd1      quad
  0.839470000000000      -11.4651702288463       1.00000000000000
   1.00000000000000
 qfab1    quad
  0.839470000000000       5.57362233202272       1.00000000000000
   1.00000000000000
 qfab2    quad
   2.28600000000000      0.000000000000000E+00   1.00000000000000
   1.00000000000000
 qfab3    quad
   1.82880000000000      0.000000000000000E+00   1.00000000000000
   1.00000000000000
 qfab4    quad
   1.82880000000000      0.000000000000000E+00   1.00000000000000
   1.00000000000000
 qfab5    quad
   1.05054400000000       11.4651702288463       1.00000000000000
   1.00000000000000
 qfab6    quad
   1.26238000000000       11.4651702288463       1.00000000000000
   1.00000000000000
 qdab1    quad
  0.839470000000000      -5.57362233202272       1.00000000000000
   1.00000000000000
 qdab2    quad
```

```
  2.28600000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
qdab3    quad
  1.82880000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
qdab4    quad
  1.82880000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
qdab5    quad
  1.05054400000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdab6    quad
  1.26238000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qfbc1    quad
 0.407289000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qfbc2    quad
  1.05054400000000       11.4651702288463       1.00000000000000
  1.00000000000000
qfbc3    quad
  1.26238000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qdbc1    quad
 0.407289000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdbc2    quad
  1.05054400000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdbc3    quad
  1.26238000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qfcd1    quad
 0.323850000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qfcd2    quad
  1.14541300000000       11.4651702288463       1.00000000000000
  1.00000000000000
qfcd3    quad
  1.26238000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qdcd1    quad
 0.323850000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdcd2    quad
  1.14541300000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdcd3    quad
  1.26238000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qfde1    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
```

```
qfde2    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
qfde3    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
qdde1    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
qdde2    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
qdde3    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
qfef1    quad
 0.407289000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qfef2    quad
  1.05054400000000       11.4651702288463       1.00000000000000
  1.00000000000000
qfef3    quad
  1.26238000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qdef1    quad
 0.407289000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdef2    quad
  1.05054400000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdef3    quad
  1.26238000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qffa1    quad
 0.323850000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qffa2    quad
  1.14541300000000       11.4651702288463       1.00000000000000
  1.00000000000000
qffa3    quad
  1.26238000000000       11.4651702288463       1.00000000000000
  1.00000000000000
qdfa1    quad
 0.323850000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdfa2    quad
  1.14541300000000      -11.4651702288463       1.00000000000000
  1.00000000000000
qdfa3    quad
  1.26238000000000      -11.4651702288463       1.00000000000000
  1.00000000000000
dr1      drft
 0.279400000000000
```

```
dr2      drft
 0.254000000000000
dr3      drft
 0.381000000000000
dr4      drft
  1.12433100000000
dr5      drft
 0.600379800000000
dr6      drft
  13.5057642000000
dr6bke17 drft
  6.75288210000000
dr6ake17 drft
  6.75288210000000
dr7      drft
  1.01945440000000
dr8      drft
 0.362331000000000
dr9      drft
  7.97994340000000
dr10     drft
  1.92595500000000
dr11     drft
  26.5964670000000
dr12     drft
  1.57919420000000
dr13     drft
  8.03554400000000
dr14     drft
  1.90957200000000
dr15     drft
  26.5967853000000
dr16     drft
  1.63482020000000
dr17     drft
 0.320980000000000
dr18     drft
  7.45060000000000
dr19     drft
  1.29540000000000
dr20     drft
 0.876300000000000
dr21     drft
 0.381000000000000
dr22     drft
  7.62300000000000
dr23     drft
  1.63480000000000
dr24     drft
  11.2340090000000
dr25     drft
 0.600379800000000
dr26     drft
```

```
  4.24680000000000
dr27     drft
  4.81182680000000
dr28     drft
  9.36772300000000
dr29     drft
  9.38100700000000
dr30     drft
  1.63480000000000
dr31     drft
  3.47786500000000
dr32     drft
  1.00363000000000
dr33     drft
  10.7251500000000
dr34     drft
  2.72647000000000
dr35     drft
 0.913330000000000
dr36     drft
  2.81678000000000
dr37     drft
  1.57920000000000
dr38     drft
  2.14028000000000
dr39     drft
  1.55537000000000
s0c1     thlm
 0.000000000000000E+00  0.000000000000000E+00   3.68808805114383
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
s0c2     thlm
 0.000000000000000E+00  0.000000000000000E+00  -6.02362277156882
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
dsex+    thlm
 0.000000000000000E+00  0.000000000000000E+00   105.555729800000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
dsex-    thlm
 0.000000000000000E+00  0.000000000000000E+00  -105.555729800000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd12    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd14    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd16    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd18    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd23    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
```

```
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd27    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd37    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ssd43    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
q0c1     thlm
-1.873584106403250E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
q0c2     thlm
-4.420029539156910E-02  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
sq0c1    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
sq0c2    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
sq0c3    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
sq0c4    thlm
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
rfca     srfc
 -720000.000000000       53000000.0000000
dummywrt mark
a0       mark
a11      mark
a12      mark
a13      mark
a14      mark
a15      mark
a16      mark
a17      mark
a18      mark
a19      mark
a21      mark
a22      mark
a23      mark
a24      mark
a25      mark
a26      mark
a27      mark
a28      mark
a29      mark
a32      mark
a33      mark
a34      mark
```

```
a35      mark
a36      mark
a37      mark
a38      mark
a39      mark
a42      mark
a43      mark
a44      mark
a45      mark
a46      mark
a47      mark
a48      mark
a49      mark
b0       mark
b11      mark
b12      mark
b13      mark
b14      mark
b15      mark
b16      mark
b17      mark
b18      mark
b19      mark
b21      mark
b22      mark
b23      mark
b24      mark
b25      mark
b26      mark
b27      mark
b28      mark
b29      mark
b32      mark
b33      mark
b34      mark
b35      mark
b36      mark
b37      mark
b38      mark
b39      mark
b42      mark
b43      mark
b44      mark
b45      mark
b46      mark
b47      mark
b48      mark
b49      mark
c0       mark
c11      mark
c12      mark
c13      mark
c14      mark
```

```
c15      mark
c16      mark
c17      mark
c18      mark
c19      mark
c21      mark
c22      mark
c23      mark
c24      mark
c25      mark
c26      mark
c27      mark
c28      mark
c29      mark
c32      mark
c33      mark
c34      mark
c35      mark
c36      mark
c37      mark
c38      mark
c39      mark
c42      mark
c43      mark
c44      mark
c45      mark
c46      mark
c47      mark
c48      mark
c49      mark
d0       mark
d11      mark
d12      mark
d13      mark
d14      mark
d15      mark
d16      mark
d17      mark
d18      mark
d19      mark
d21      mark
d22      mark
d23      mark
d24      mark
d25      mark
d26      mark
d27      mark
d28      mark
d29      mark
d32      mark
d33      mark
d34      mark
d35      mark
```

```
d36      mark
d37      mark
d38      mark
d39      mark
d42      mark
d43      mark
d44      mark
d45      mark
d46      mark
d47      mark
d48      mark
d49      mark
e0       mark
e11      mark
e12      mark
e13      mark
e14      mark
e15      mark
e16      mark
e17      mark
kicke17m mark
e18      mark
e19      mark
e21      mark
e22      mark
e23      mark
hpme24   mark
e24      mark
vpme25   mark
e25      mark
hpme26m  mark
e26      mark
vpme27   mark
e27      mark
e28      mark
e29      mark
e32      mark
e33      mark
e34      mark
e35      mark
e36      mark
e37      mark
e38      mark
e39      mark
e42      mark
e43      mark
e44      mark
e45      mark
e46      mark
e47      mark
e48      mark
e49      mark
f0       mark
```

```
f11      mark
f12      mark
f13      mark
f14      mark
f15      mark
f16      mark
f17      mark
f18      mark
f19      mark
f21      mark
f22      mark
f23      mark
f24      mark
f25      mark
f26      mark
f27      mark
f28      mark
f29      mark
f32      mark
f33      mark
f34      mark
f35      mark
f36      mark
f37      mark
f38      mark
f39      mark
f42      mark
f43      mark
f44      mark
f45      mark
f46      mark
f47      mark
f48      mark
f49      mark
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
clear    iden
tadm     tadm
  1.00000000000000      0.000000000000000E+00   3.00000000000000
 0.000000000000000E+00
tmo      tmo
  10.0000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
inv      inv
gbuf1    gbuf
  2.00000000000000       1.00000000000000
gbuf2    gbuf
  2.00000000000000       2.00000000000000
icin     rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
```

```
raysin   rt
  14.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
track14  rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  2.00000000000000       1.00000000000000      0.000000000000000E+00
rmap1    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
 0.000000000000000E+00
rmap2    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  1.00000000000000
rmap3    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  2.00000000000000
rmap4    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  3.00000000000000
rmap5    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  4.00000000000000
rmap6    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  5.00000000000000
rmap7    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  6.00000000000000
rmap8    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  7.00000000000000
rmap9    tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  8.00000000000000
rmap10   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  9.00000000000000
rmap11   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  10.0000000000000
rmap12   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  11.0000000000000
rmap13   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  12.0000000000000
rmap14   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  13.0000000000000
rmap15   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  14.0000000000000
rmap16   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
```

```
  15.0000000000000
 rmap17   tmi
  1.00000000000000       10.0000000000000       1.00000000000000
  16.0000000000000
 circ14   circ
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  10000.0000000000       43.0000000000000       1.00000000000000
 circ14e  circ
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  2000.00000000000       1.00000000000000       1.00000000000000
 stat     whst
  20.0000000000000       1.00000000000000
 hist     whst
  22.0000000000000       2.00000000000000
 zer      zer
 0.000000000000000E+00  1.000000000000000E-10  0.000000000000000E+00
 bgen     bgen
  12.0000000000000       123.000000000000       30.0000000000000
  123.000000000000       1.00000000000000       1.00000000000000
 mom      ps1
 5.310000000000000E-08  5.310000000000000E-08  5.999999999999999E-04
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 xform14  rt
 0.000000000000000E+00   14.0000000000000       3.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
 xform24  rt
 0.000000000000000E+00   24.0000000000000       3.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
 rout16   rt
 0.000000000000000E+00   16.0000000000000       3.00000000000000
 0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 rout18f  rt
 0.000000000000000E+00  -18.0000000000000       3.00000000000000
 0.000000000000000E+00   1.00000000000000      0.000000000000000E+00
 end      end
#lines
 kicke17w
     1*wrtmap
 hpme26w
     1*wrtmap
 wrtmap
     1*tmo         1*clear
 wsc22
     1*dsex+       1*wrtmap
 wsc24
     1*dsex-       1*wrtmap
 wsc26
     1*dsex+       1*wrtmap
 wsc28
     1*dsex-       1*wrtmap
 wsc32
     1*dsex+       1*wrtmap
 wsc34
```

```
    1*dsex-       1*wrtmap
wsc36
    1*dsex+       1*wrtmap
wsc38
    1*dsex-       1*wrtmap
wsf22
    1*dsex+       1*wrtmap
wsf24
    1*dsex-       1*wrtmap
wsf26
    1*dsex+       1*wrtmap
wsf28
    1*dsex-       1*wrtmap
wsf32
    1*dsex+       1*wrtmap
wsf34
    1*dsex-       1*wrtmap
wsf36
    1*dsex+       1*wrtmap
wsf38
    1*dsex-       1*wrtmap
spnormfd
    1*dr3         1*sq0c1       1*s0c1        1*q0c1        1*dr3
spnormdf
    1*dr3         1*sq0c2       1*s0c2        1*q0c2        1*dr3
bnd4
    1*bh1         1*dr1         1*bh1         1*dr1         1*bh1      &
    1*dr1         1*bh1
bnd3
    1*bh1         1*dr1         1*bh1         1*dr1         1*bh1
bnd2
    1*bh1         1*dr1         1*bh1
setcxfd
    1*clear       1*qf1         1*dr2         1*spnormfd    1*dr4      &
    1*bnd4        1*dr5         1*qd1
cxfd
    1*lcxfd
setcxdf
    1*clear       1*qd1         1*dr2         1*spnormdf    1*dr4      &
    1*bnd4        1*dr5         1*qf1
cxdf
    1*lcxdf
eeef
    1*qd1         1*dr2         1*spnormdf    1*dr4         1*bnd4     &
    1*dr5         1*qf1
cxbeg
    1*dr38        1*bnd4        1*dr5         1*qf1
cdbeg
    1*dr2         1*dr3         1*dr3         1*dr3         1*ssd12    &
    1*dr3         1*dr8         1*bnd4        1*dr5         1*qf1
cxend
    1*qd1         1*dr2         1*spnormdf    1*dr4         1*bnd4     &
    1*dr5
```

```
ca47
    1*qd1          1*dr2          1*spnormdf     1*dr4          1*bnd4      &
    1*dr17
cbbeg
    1*dr39         1*bnd4         1*dr5          1*qf1
wcc22
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc22     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc24
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc24     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc26
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc26     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc28
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc28     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc32
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc32     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc34
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc34     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc36
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc36     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcc38
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsc38     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf22
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf22     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf24
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf24     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf26
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf26     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf28
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf28     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf32
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf32     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf34
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf34     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf36
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf36     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
wcf38
    1*qf1          1*dr2          1*spnormfd     1*dr3          1*wsf38     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qd1
```

```
cd14
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd14     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cd16
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd16     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cd18
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd18     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cd23
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd23     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cd27
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd27     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cd37
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd37     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cd43
    1*qd1          1*dr2          1*spnormfd     1*dr3          1*ssd43     &
    1*dr3          1*dr8          1*bnd4         1*dr5          1*qf1
cx17
    1*qf1          1*dr2          1*spnormfd     1*dr6          1*bnd2      &
    1*dr7          1*qd1
ex17
    1*qf1          1*dr2          1*spnormfd     1*dr6bke17     1*kicke17m &
    1*dr6ake17     1*bnd2         1*dr7          1*qd1
wex17
    1*qf1          1*dr2          1*spnormfd     1*dr6bke17     1*kicke17w &
    1*dr6ake17     1*bnd2         1*dr7          1*qd1
fx17
    1*qf1          1*dr2          1*spnormfd     1*dr6          1*bnd2      &
    1*dr7          1*qd1
axarc
    1*cxbeg        1*a13          1*cxfd         1*a14          1*cxdf      &
    1*a15          1*cxfd         1*a16          1*cxdf         1*a17       &
    1*cx17         1*a18          1*cxdf         1*a19          1*cxfd      &
    1*a21          1*cxdf         1*a22          1*cxfd         1*a23       &
    1*cxdf         1*a24          1*cxfd         1*a25          1*cxdf      &
    1*a26          1*cxfd         1*a27          1*cxdf         1*a28       &
    1*cxfd         1*a29          1*cxdf         1*a32          1*cxfd      &
    1*a33          1*cxdf         1*a34          1*cxfd         1*a35       &
    1*cxdf         1*a36          1*cxfd         1*a37          1*cxdf      &
    1*a38          1*cxfd         1*a39          1*cxdf         1*a42       &
    1*cxfd         1*a43          1*cxdf         1*a44          1*cxfd      &
    1*a45          1*cxdf         1*a46          1*cxfd         1*a47       &
    1*ca47
bxarc
    1*cbbeg        1*b13          1*cxfd         1*b14          1*cxdf      &
    1*b15          1*cxfd         1*b16          1*cxdf         1*b17       &
    1*cx17         1*b18          1*cxdf         1*b19          1*cxfd      &
    1*b21          1*cxdf         1*b22          1*cxfd         1*b23       &
    1*cxdf         1*b24          1*cxfd         1*b25          1*cxdf      &
```

```
    1*b26         1*cxfd        1*b27         1*cxdf        1*b28      &
    1*cxfd        1*b29         1*cxdf        1*b32         1*cxfd     &
    1*b33         1*cxdf        1*b34         1*cxfd        1*b35      &
    1*cxdf        1*b36         1*cxfd        1*b37         1*cxdf     &
    1*b38         1*cxfd        1*b39         1*cxdf        1*b42      &
    1*cxfd        1*b43         1*cxdf        1*b44         1*cxfd     &
    1*b45         1*cxdf        1*b46         1*cxfd        1*b47      &
    1*cxend
wcxarc
    1*cxbeg       1*c13         1*cxfd        1*c14         1*cxdf     &
    1*c15         1*cxfd        1*c16         1*cxdf        1*c17      &
    1*cx17        1*c18         1*cxdf        1*c19         1*cxfd     &
    1*c21         1*cxdf        1*c22         1*wcc22       1*c23      &
    1*cxdf        1*c24         1*wcc24       1*c25         1*cxdf     &
    1*c26         1*wcc26       1*c27         1*cxdf        1*c28      &
    1*wcc28       1*c29         1*cxdf        1*c32         1*wcc32    &
    1*c33         1*cxdf        1*c34         1*wcc34       1*c35      &
    1*cxdf        1*c36         1*wcc36       1*c37         1*cxdf     &
    1*c38         1*wcc38       1*c39         1*cxdf        1*c42      &
    1*cxfd        1*c43         1*cxdf        1*c44         1*cxfd     &
    1*c45         1*cxdf        1*c46         1*cxfd        1*c47      &
    1*cxend
dxarc
    1*cdbeg       1*d13         1*cxfd        1*d14         1*cd14     &
    1*d15         1*cxfd        1*d16         1*cd16        1*d17      &
    1*cx17        1*d18         1*cd18        1*d19         1*cxfd     &
    1*d21         1*cxdf        1*d22         1*cxfd        1*d23      &
    1*cd23        1*d24         1*cxfd        1*d25         1*cxdf     &
    1*d26         1*cxfd        1*d27         1*cd27        1*d28      &
    1*cxfd        1*d29         1*cxdf        1*d32         1*cxfd     &
    1*d33         1*cxdf        1*d34         1*cxfd        1*d35      &
    1*cxdf        1*d36         1*cxfd        1*d37         1*cd37     &
    1*d38         1*cxfd        1*d39         1*cxdf        1*d42      &
    1*cxfd        1*d43         1*cd43        1*d44         1*cxfd     &
    1*d45         1*cxdf        1*d46         1*cxfd        1*d47      &
    1*cxend
wexarc
    1*cxbeg       1*e13         1*cxfd        1*e14         1*cxdf     &
    1*e15         1*cxfd        1*e16         1*cxdf        1*e17      &
    1*wex17       1*e18         1*cxdf        1*e19         1*cxfd     &
    1*e21         1*cxdf        1*e22         1*cxfd        1*e23      &
    1*cxdf        1*e24         1*cxfd        1*e25         1*cxdf     &
    1*e26         1*cxfd        1*e27         1*cxdf        1*e28      &
    1*cxfd        1*e29         1*cxdf        1*e32         1*cxfd     &
    1*e33         1*cxdf        1*e34         1*cxfd        1*e35      &
    1*cxdf        1*e36         1*cxfd        1*e37         1*cxdf     &
    1*e38         1*cxfd        1*e39         1*cxdf        1*e42      &
    1*cxfd        1*e43         1*cxdf        1*e44         1*cxfd     &
    1*e45         1*cxdf        1*e46         1*cxfd        1*e47      &
    1*cxend
wfxarc
    1*cxbeg       1*f13         1*cxfd        1*f14         1*cxdf     &
    1*f15         1*cxfd        1*f16         1*cxdf        1*f17      &
```

```
    1*fx17        1*f18         1*cxdf        1*f19         1*cxfd     &
    1*f21         1*cxdf        1*f22         1*wcf22       1*f23      &
    1*cxdf        1*f24         1*wcf24       1*f25         1*cxdf     &
    1*f26         1*wcf26       1*f27         1*cxdf        1*f28      &
    1*wcf28       1*f29         1*cxdf        1*f32         1*wcf32    &
    1*f33         1*cxdf        1*f34         1*wcf34       1*f35      &
    1*cxdf        1*f36         1*wcf36       1*f37         1*cxdf     &
    1*f38         1*wcf38       1*f39         1*cxdf        1*f42      &
    1*cxfd        1*f43         1*cxdf        1*f44         1*cxfd     &
    1*f45         1*cxdf        1*f46         1*cxfd        1*f47      &
    1*cxend
fxst
    1*qffa1       1*f48         1*qffa1       1*dr9         1*bnd3     &
    1*dr5         1*qdfa2       1*qdfa2       1*dr10        1*f49      &
    1*dr10        1*qffa3       1*qffa3       1*dr11
axst
    1*a0          1*dr11        1*qdfa3       1*qdfa3       1*dr10     &
    1*a11         1*dr10        1*qffa2       1*qffa2       1*dr12     &
    1*bnd4        1*dr5         1*qdfa1       1*a12         1*qdfa1
efxst
    1*qfef1       1*e48         1*qfef1       1*dr13        1*bnd3     &
    1*dr5         1*qfef2       1*qfef2       1*dr14        1*e49      &
    1*dr14        1*qdef3       1*qdef3       1*dr15        1*rfca     &
    1*f0          1*dr15        1*qfef3       1*qfef3       1*dr14     &
    1*f11         1*dr14        1*qdef2       1*qdef2       1*dr16     &
    1*bnd4        1*dr5         1*qdef1       1*f12         1*qdef1
abxst
    1*qfab1       1*a48         1*qfab1       1*dr18        1*bnd3     &
    1*dr5         1*qfab5       1*qfab5       1*dr14        1*a49      &
    1*dr14        1*qdab6       1*qdab6       1*dr19        1*qfab2    &
    1*qfab2       1*dr20        1*qdab3       1*qdab3       1*dr21     &
    1*qdab3       1*qdab3       1*dr20        1*qfab4       1*qfab4    &
    1*dr22        1*b0          1*dr22        1*qdab4       1*qdab4    &
    1*dr20        1*qfab3       1*qfab3       1*dr21        1*qfab3    &
    1*qfab3       1*dr20        1*qdab2       1*qdab2       1*dr19     &
    1*qfab6       1*qfab6       1*dr14        1*b11         1*dr14     &
    1*qdab5       1*qdab5       1*dr23        1*bnd4        1*dr17     &
    1*qdab1       1*b12         1*qdab1
bcxst
    1*qfbc1       1*b48         1*qfbc1       1*dr24        1*bbs      &
    1*dr1         1*bnd2        1*dr25        1*qfbc2       1*qfbc2    &
    1*dr14        1*b49         1*dr14        1*qdbc3       1*qdbc3    &
    1*dr26        1*bl          1*dr1         1*bl          1*dr1      &
    1*bl          1*dr27        1*c0          1*dr28        1*bc       &
    1*dr1         1*bc          1*dr29        1*qfbc3       1*qfbc3    &
    1*dr14        1*c11         1*dr14        1*qdbc2       1*qdbc2    &
    1*dr30        1*bh1         1*dr31        1*bbs         1*dr1      &
    1*bnd2        1*dr5         1*qdbc1       1*c12         1*qdbc1
cdxst
    1*qfcd1       1*c48         1*qfcd1       1*dr9         1*bnd3     &
    1*dr5         1*qdcd2       1*qdcd2       1*dr10        1*c49      &
    1*dr10        1*qfcd3       1*qfcd3       1*dr32        1*bbmi002z &
    1*bbmi        1*bbmi003z    1*dr33        1*bbpl        1*dr34     &
```

```
     1*d0          1*dr35        1*bbpl        1*dr33        1*bbmi002z &
     1*bbmi        1*bbmi003z    1*dr36        1*qdcd3       1*qdcd3    &
     1*dr10        1*d11         1*dr10        1*qfcd2       1*qfcd2    &
     1*dr37        1*bnd4        1*dr5         1*qdcd1       1*d12      &
     1*qdcd1
 dexst
     1*qfef1       1*d48         1*qfef1       1*dr13        1*bnd3     &
     1*dr5         1*qfef2       1*qfef2       1*dr14        1*d49      &
     1*dr14        1*qdef3       1*qdef3       1*dr15        1*e0       &
     1*dr15        1*qfef3       1*qfef3       1*dr14        1*e11      &
     1*dr14        1*qdef2       1*qdef2       1*dr16        1*bnd4     &
     1*dr5         1*qdef1       1*e12         1*qdef1
 wring
     1*hpme24      1*e24         1*cxfd        1*vpme25      1*e25      &
     1*cxdf        1*hpme26m     1*e26         1*cxfd        1*vpme27   &
     1*e27         1*cxdf        1*e28         1*cxfd        1*e29      &
     1*cxdf        1*e32         1*cxfd        1*e33         1*cxdf     &
     1*e34         1*cxfd        1*e35         1*cxdf        1*e36      &
     1*cxfd        1*e37         1*cxdf        1*e38         1*cxfd     &
     1*e39         1*cxdf        1*e42         1*cxfd        1*e43      &
     1*cxdf        1*e44         1*cxfd        1*e45         1*cxdf     &
     1*e46         1*cxfd        1*e47         1*cxend       1*efxst    &
     1*wfxarc      1*fxst        1*axst        1*axarc       1*abxst    &
     1*bxarc       1*bcxst       1*wcxarc      1*cdxst       1*dxarc    &
     1*dexst       1*cxbeg       1*e13         1*cxfd        1*e14      &
     1*cxdf        1*e15         1*cxfd        1*e16         1*cxdf     &
     1*e17         1*ex17        1*e18         1*cxdf        1*e19      &
     1*cxfd        1*e21         1*qd1         1*dr2         1*spnormdf &
     1*dr4         1*bnd4        1*dr5         1*qf1         1*e22      &
     1*cxfd        1*e23         1*cxdf        1*wrtmap
 totmap
     1*lm1         1*lm2         1*lm3         1*lm4         1*lm5      &
     1*lm6         1*lm7         1*lm8         1*lm9         1*lm10     &
     1*lm11        1*lm12        1*lm13        1*lm14        1*lm15     &
     1*lm16        1*lm17
#lumps
 lcxfd
     1*setcxfd
 lcxdf
     1*setcxdf
 lm1
     1*rmap1
 lm2
     1*rmap2
 lm3
     1*rmap3
 lm4
     1*rmap4
 lm5
     1*rmap5
 lm6
     1*rmap6
 lm7
```

```
     1*rmap7
 lm8
     1*rmap8
 lm9
     1*rmap9
 lm10
     1*rmap10
 lm11
     1*rmap11
 lm12
     1*rmap12
 lm13
     1*rmap13
 lm14
     1*rmap14
 lm15
     1*rmap15
 lm16
     1*rmap16
 lm17
     1*rmap17
#loops
 lmpbylmp
     1*lm1         1*lm2         1*lm3         1*lm4         1*lm5      &
     1*lm6         1*lm7         1*lm8         1*lm9         1*lm10     &
     1*lm11        1*lm12        1*lm13        1*lm14        1*lm15     &
     1*lm16        1*lm17
#labor
    1*fileout
    1*zer
    1*lcxfd
    1*lcxdf
    1*clear
    1*wring
    1*totmap
    1*tadm
    1*clear
    1*gbuf1
    1*mapout
    1*mom
    1*bgen
    1*xform14
    1*clear
    1*rout16
    1*rout18f
    1*clear
    1*lmpbylmp
    1*circ14
    1*stat
    1*hist
    1*end

*******************************************************
```

```
* Response requests to make lump maps for cxfd and cxdf: *
*********************************************************

lump lcxfd    constructed and stored.( 1)
lump lcxdf    constructed and stored.( 2)

********************************
* Writing out maps on file 10: *
********************************

starting new map file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10
adding current map to file on unit 10

*****************************************
* Reading maps back in and making lumps: *
*****************************************

file unit 10 rewound
map read in from file  10;   0 record(s) skipped
lump lm1      constructed and stored.( 3)
file unit 10 rewound
map read in from file  10;   1 record(s) skipped
lump lm2      constructed and stored.( 4)
file unit 10 rewound
map read in from file  10;   2 record(s) skipped
lump lm3      constructed and stored.( 5)
file unit 10 rewound
map read in from file  10;   3 record(s) skipped
lump lm4      constructed and stored.( 6)
file unit 10 rewound
map read in from file  10;   4 record(s) skipped
lump lm5      constructed and stored.( 7)
file unit 10 rewound
map read in from file  10;   5 record(s) skipped
lump lm6      constructed and stored.( 8)
file unit 10 rewound
map read in from file  10;   6 record(s) skipped
lump lm7      constructed and stored.( 9)
```

```
file unit 10 rewound
map read in from file  10;   7 record(s) skipped
lump lm8      constructed and stored.(10)
file unit 10 rewound
map read in from file  10;   8 record(s) skipped
lump lm9      constructed and stored.(11)
file unit 10 rewound
map read in from file  10;   9 record(s) skipped
lump lm10     constructed and stored.(12)
file unit 10 rewound
map read in from file  10;  10 record(s) skipped
lump lm11     constructed and stored.(13)
file unit 10 rewound
map read in from file  10;  11 record(s) skipped
lump lm12     constructed and stored.(14)
file unit 10 rewound
map read in from file  10;  12 record(s) skipped
lump lm13     constructed and stored.(15)
file unit 10 rewound
map read in from file  10;  13 record(s) skipped
lump lm14     constructed and stored.(16)
file unit 10 rewound
map read in from file  10;  14 record(s) skipped
lump lm15     constructed and stored.(17)
file unit 10 rewound
map read in from file  10;  15 record(s) skipped
lump lm16     constructed and stored.(18)
file unit 10 rewound
map read in from file  10;  16 record(s) skipped
lump lm17     constructed and stored.(19)

*******************************************
* Analysis of 1-turn map made from lumps: *
*******************************************

twiss analysis of dynamic map

horizontal tune =  0.389999973016777
vertical tune =  0.460000000580470
temporal tune = -1.540349178325981E-003

normalized anharmonicities
 hhn=  -77326.8639811444
 vvn=  -4818.30454004036
 ttn=  0.217365676500829
 hvn=  -18022.7819055046
 htn=   6.39304607228565
 vtn=  5.567733042679678E-003

******************************
* "Linear" part of script A: *
******************************
```

```
matrix for map is :

 1.00358E+01  0.00000E+00 -6.80430E-14 -6.17887E-14 -8.11623E-05 -5.09857E-02
 1.40221E-04  9.96432E-02  2.87319E-17 -7.24095E-17 -6.08413E-07 -2.33048E-04
 4.36877E-20  6.00495E-20  5.35004E+00  0.00000E+00  1.86323E-17 -4.19422E-18
-5.66633E-22 -3.85387E-21 -6.08481E-03  1.86914E-01 -2.48534E-19 -3.28674E-18
 9.96505E-02 -2.17124E-01 -5.46098E-16  2.19142E-16  4.27377E+01  0.00000E+00
-3.45704E-07  6.31758E-07 -1.76267E-20  3.03092E-17 -8.71051E-05  2.33986E-02

nonzero elements in generating polynomial are :

******************************
* Response to bgen command: *
******************************

         30 rays generated

 numerically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 4.783083287136278E-008 -1.292970142712755E-008  5.836916712863723E-008
 values of <y*y>, <y*py>, <py*py>:
 4.467706053833843E-008  5.881425116420553E-009  6.152293946166155E-008
 values of <t*t>, <t*pt>, <pt*pt>:
 7.082762262866653E-004 -1.121407686789513E-004  4.917237737133348E-004

 analytically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 5.309999999999999E-008  0.000000000000000E+000  5.309999999999999E-008
 values of <y*y>, <y*py>, <py*py>:
 5.309999999999999E-008  0.000000000000000E+000  5.309999999999999E-008
 values of <t*t>, <t*pt>, <pt*pt>:
 6.000000000000001E-004  0.000000000000000E+000  6.000000000000001E-004

******************************
* response to circ command: *
******************************

circulating through lmpbylmp :
 lm1      lm2      lm3      lm4      lm5
 lm6      lm7      lm8      lm9      lm10
 lm11     lm12     lm13     lm14     lm15
 lm16     lm17

************************************************
* Message from symplectic raytracer indicating *
* large amplitude resulting in particle loss:  *
************************************************

 Search did not converge; root=  0.70332E-01

***********************************
* Responses to the whst commands: *
***********************************
```

```
istat written on file           20
nlost =           1
ihist written on file           22

end of MARYLIE run
```

10.8.2 History of Lost Particle

Exhibit 10.8.2a shows the first 10 lines of the 30 lines in the "ordinary" precision initial conditions file (file 16) produced by the MaryLie run of subsection 10.8.1. The initial condition values for the lost particle, particle 6, are on line 6:

```
 1.10064E-03 -3.00936E-05 -1.24754E-03 -4.11328E-05  1.45749E+00 -1.45480E-04
```

This particle can be identified in figures 10.8.1.2 through 10.8.1.4. Evidently it does not appear to have any "distinguished" betatron and synchrotron phase. Correspondingly, it is an initial condition that might be missed in tracking studies that use only a more restricted set of initial conditions.

Exhibit 10.8.2b displays the corresponding initial condition values for the lost particle in full precision as extracted from file 18. The turn-by-turn behavior of this particle can be studied by a special MaryLie run as illustrated in Exhibit 10.8.2c. The initial condition values are placed in an input file (file 13 in this case) and the particle is tracked (with results written out on file 14 every turn) until it is lost.

Figures 10.8.2.1 through 10.8.2.3 show this tracking data. Evidently the motion remains bounded for a large number of turns, and then the particle is abruptly lost within a few turns. To study this tracking data in more detail, it is read back in, transformed by the full nonlinear normalizing map $\mathcal{A}^{-1}$, and again read out (onto file 24). Figures 10.8.2.4 through 10.8.2.6 show this transformed tracking data. Evidently the vertical and temporal projections of the transformed data are more regular than their counterparts in figures 10.8.2.2 and 10.8.2.3, and lie very nearly on tori. However, the horizontal projection (figure 10.8.2.4) is at best only marginally more regular than its counterpart in figure 10.8.2.1. Consequently, either effects beyond third order are important for normalization calculations, or the motion is chaotic, does not lie on a topological torus, and is irregular no matter how it is viewed.

Figure 10.8.2.1: Horizontal projection of turn-by-turn phase-space tracking data (file 14) for lost particle.

Figure 10.8.2.2: Vertical projection of turn-by-turn phase-space tracking data (file 14) for lost particle.

Figure 10.8.2.3: Temporal projection of turn-by-turn phase-space tracking data (file 14) for lost particle.

Figure 10.8.2.4: Horizontal projection of transformed phase-space data (file 24).

Figure 10.8.2.5: Vertical projection of transformed phase-space data (file 24).

Figure 10.8.2.6: Temporal projection of transformed phase-space data (file 24).

```
Exhibit 10.8.2a
First 10 lines from "ordinary" precision initial conditions file
(file 16) produced by the MaryLie run of subsection 10.8.1:

-2.72488E-03  2.24632E-05 -1.12649E-03 -4.52098E-05 -1.33566E+00  3.52445E-04
-2.21313E-03  1.37555E-06 -9.51780E-04  5.21177E-05  1.19509E+00 -4.80832E-04
 1.52671E-03 -1.46330E-05 -1.04655E-03 -4.75277E-05 -4.90794E-01  7.65706E-04
-1.42592E-03  1.59351E-05 -1.26283E-03  4.34335E-05 -2.38677E-01 -7.99465E-04
-1.62590E-03 -3.65853E-05 -2.60833E-04 -5.99301E-05  1.21025E+00  4.64431E-04
 1.10064E-03 -3.00936E-05 -1.24754E-03 -4.11328E-05  1.45749E+00 -1.45480E-04
 4.02458E-03 -9.72345E-06 -1.21551E-03 -4.22860E-05  1.14555E+00 -5.15828E-04
 6.26568E-04  3.84789E-05  1.42061E-03 -3.69284E-05  1.94167E-01 -8.03939E-04
 3.24966E-03 -6.99122E-06  1.70503E-03 -1.46624E-05  1.47949E+00 -3.33364E-05
 3.27788E-03  3.22586E-05 -5.60920E-05  6.09446E-05 -9.70909E-01 -6.09945E-04


Exhibit 10.8.2b
Full precision initial conditions for the lost particle, corresponding to
line 6 above, as extracted from file 18:
```

```
 1.100643642212187E-003
-3.009364428172349E-005
-1.247535309762509E-003
-4.113280353188271E-005
  1.45749101315727
-1.454801593473802E-004


Exhibit 10.8.2c
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
Exhibit 10.8.2.
    This is a MARYLIE run to study the dynamic aperture of the Tevatron when
16 distortion sextupoles are powered.  The 'quads' q0c1 and q0c2 are set to
achieve horizontal and vertical tunes of .39 and .46, respectively, and the
'sextupoles' s0c1 and s0c2 are set to achieve zero first-order chromaticities.
Both these fits were made with the RF cavity (rfca) turned off.  In this run
the RF cavity is powered.  As one can see from the response to the tadm
command, the transverse tunes shift slightly when the cavity is turned on due
both to synchro-betatron coupling and the defocussing effects of the cavity.
The distortion sextupoles dsex+ and dsex- are set to strengths corresponding
to 30 amperes  excitation current.  Fringe-field effects are included.
Finally, the beam parameters are those for 150 GeV protons.

This MaryLie run does 10 things:

 a) It computes maps for the lines cxfd and cxdf, which occur
    repeatedly in the Tevatron lattice, and stores them as lumps
    in order to speed up subsequent computations.

 b) It computes 17 transfer maps and writes them out sequentially on file 10.
    The 1st transfer map describes the elements stretching from the location
    of a  horizontal beam position monitor in sector E (location e24) to the
    trailing end of the 1st distortion sextupole.  The 2nd transfer map
    describes the elements from the trailing end of the 1st distortion
    sextupole location to the trailing end of the 2nd distortion
    sextupole. ... The 16th transfer map describes the elements from the
    trailing end of the 15th distortion sextupole location to the trailing
    end of the 16th distortion sextupole.  Finally, the 17th transfer map
    describes the elements from the trailing end of the 16th distortion
    sextupole location back to the horizontal position monitor location.
    Thus, when combined, these 17 transfer maps give a full one-turn map.

 c) It reads these maps back in and stores them as lumps.

 d) These 17 maps are concatenated to give a one-turn map.

 e) A tadm command is used to compute script A and the tunes
```

```
    for the one-turn map.

 f) A "lost" linearly matched ray is read in from file 13.

 g) This ray is tracked lump-by-lump, with results written
    on file 14 every turn, until it is lost.

 h) The istat and ihist arrays are written on files 20 and 22,
    respectively.

 i) File 14 is read back in.

 j) The full script A for the one-turn map is gotten from
    buffer 2, inverted, and applied to the tracked rays with
    the results written on file 24.


#beam

*********************************************************
* The #beam, #menu, #lines, #lumps, and #loops components *
* of this file are the same as those in Exhibit 10.8.1c.  *
*********************************************************

#labor
    1*fileout
    1*zer
    1*lcxfd
    1*lcxdf
    1*clear
    1*wring
    1*totmap
    1*tadm
    1*clear
    1*icin
    1*xform14
    1*lmpbylmp
    1*circ14e
    1*stat
    1*hist
    1*clear
    1*raysin
    1*gbuf2
    1*inv
    1*mapout
    1*xform24
    1*end

*******************************************************
* The responses to the #labor entries 1*fileout, 1*zer, *
* 1*lcxfd, 1*lcxdf, 1*clear, 1*wring, 1*totmap, 1*tadm, *
* and 1*clear are the same as in Exhibit 10.8.1c.       *
*******************************************************
```

```
**********************************
* Response to the icin command: *
**********************************

    1 ray(s) read in from file  13

***************************************
* Response to the circ command, etc.: *
***************************************

circulating through lmpbylmp :
 lm1      lm2      lm3      lm4      lm5
 lm6      lm7      lm8      lm9      lm10
 lm11     lm12     lm13     lm14     lm15
 lm16     lm17

 Search did not converge; root=  0.70331E-01

all particles lost

istat written on file           20
nlost =           1
ihist written on file           22

***********************************
* Response to the raysin command: *
***********************************

  427 ray(s) read in from file  14

*****************************************
* The transforming map script A inverse: *
*****************************************

matrix for map is :

 9.96432E-02  0.00000E+00 -3.85387E-21 -6.00495E-20  6.31758E-07  2.17124E-01
-1.40221E-04  1.00358E+01  5.66633E-22  4.36877E-20  3.45704E-07  9.96505E-02
-7.24095E-17  6.17887E-14  1.86914E-01  0.00000E+00  3.03092E-17 -2.19142E-16
-2.87319E-17 -6.80430E-14  6.08481E-03  5.35004E+00  1.76267E-20 -5.46098E-16
-2.33048E-04  5.09857E-02 -3.28674E-18  4.19422E-18  2.33986E-02  0.00000E+00
 6.08413E-07 -8.11623E-05  2.48534E-19  1.86323E-17  8.71051E-05  4.27377E+01

nonzero elements in generating polynomial are :

 f( 28)=f( 30 00 00 )=  110.33115619351
 f( 29)=f( 21 00 00 )=  230.08011452766
 f( 32)=f( 20 00 10 )=-7.64236363671544E-04
 f( 33)=f( 20 00 01 )= 0.21061201497236
 f( 34)=f( 12 00 00 )= -409.33709895966
 f( 37)=f( 11 00 10 )=-1.98308774624551E-02
 f( 38)=f( 11 00 01 )= -1.5346294736954
```

```
f( 39)=f( 10 20 00 )= -50.321815269185
f( 40)=f( 10 11 00 )= -112.64954194020
f( 43)=f( 10 02 00 )=  67.674674230258
f( 46)=f( 10 00 20 )= 5.49264263214689E-07
f( 47)=f( 10 00 11 )=-4.22537781016221E-05
f( 48)=f( 10 00 02 )=-1.16100284894868E-02
f( 49)=f( 03 00 00 )= -41.301620516121
f( 52)=f( 02 00 10 )=-6.25872941601144E-04
f( 53)=f( 02 00 01 )= 0.28339015039882
f( 54)=f( 01 20 00 )= -53.913467515215
f( 55)=f( 01 11 00 )=  145.98463472258
f( 58)=f( 01 02 00 )=  18.416597618544
f( 61)=f( 01 00 20 )=-2.43635493527592E-07
f( 62)=f( 01 00 11 )=-2.47356004995205E-05
f( 63)=f( 01 00 02 )= 1.05531720326059E-02
f( 66)=f( 00 20 10 )=-1.44291715055299E-03
f( 67)=f( 00 20 01 )=-0.14964403783604
f( 69)=f( 00 11 10 )= 1.95582556852843E-02
f( 70)=f( 00 11 01 )= 0.33591713531964
f( 75)=f( 00 02 10 )= 1.38441203994220E-03
f( 76)=f( 00 02 01 )= 0.18024669588979
f( 80)=f( 00 00 30 )=-7.22824267157899E-03
f( 81)=f( 00 00 21 )=-9.73630888091078E-10
f( 82)=f( 00 00 12 )=-1.08424609271605E-02
f( 83)=f( 00 00 03 )= 3.30917571378989E-05
f( 84)=f( 40 00 00 )= -3699.1440602575
f( 85)=f( 31 00 00 )=  42321.968671425
f( 86)=f( 30 10 00 )= 2.66770915391513E-09
f( 87)=f( 30 01 00 )=-2.33112222830107E-09
f( 88)=f( 30 00 10 )=  4.5693950614251
f( 89)=f( 30 00 01 )=  482.31125364226
f( 90)=f( 22 00 00 )= -28181.836095754
f( 91)=f( 21 10 00 )= 4.69209955695986E-09
f( 92)=f( 21 01 00 )= 2.86314136454196E-09
f( 93)=f( 21 00 10 )=  2.8759571796469
f( 94)=f( 21 00 01 )= -429.63407177763
f( 95)=f( 20 20 00 )=  12568.221040625
f( 96)=f( 20 11 00 )= -209898.00228013
f( 99)=f( 20 02 00 )= -7762.0900789243
f(100)=f( 20 01 10 )=-2.38388163473635E-10
f(102)=f( 20 00 20 )= -2.0075663524607
f(103)=f( 20 00 11 )= -2075.1974330970
f(104)=f( 20 00 02 )=-0.84614616589268
f(105)=f( 13 00 00 )= -39644.566479127
f(106)=f( 12 10 00 )= 5.32303330736889E-09
f(107)=f( 12 01 00 )= 2.99482383271274E-10
f(108)=f( 12 00 10 )=  5.1084562336192
f(109)=f( 12 00 01 )=  236.07859432082
f(110)=f( 11 20 00 )=  195669.82408561
f(111)=f( 11 11 00 )=  51210.427258493
f(112)=f( 11 10 10 )= 4.56259809595747E-10
f(114)=f( 11 02 00 )= -194877.67131442
f(117)=f( 11 00 20 )=-1.72811708277018E-02
```

```
f(118)=f( 11 00 11 )= 7.12792983006422E-02
f(119)=f( 11 00 02 )=  2.1815181618326
f(120)=f( 10 30 00 )= 8.35080084371653E-09
f(121)=f( 10 21 00 )=-2.27815722867953E-08
f(122)=f( 10 20 10 )= -1.0536861517615
f(123)=f( 10 20 01 )= -104.83855687539
f(124)=f( 10 12 00 )=-1.45097237919057E-08
f(125)=f( 10 11 10 )=-0.30587522080300
f(126)=f( 10 11 01 )= -31.943377023477
f(130)=f( 10 03 00 )= 1.01799936033082E-08
f(131)=f( 10 02 10 )=  2.1028938077905
f(132)=f( 10 02 01 )=  272.62188535886
f(136)=f( 10 00 30 )=-5.55035395974697E-03
f(137)=f( 10 00 21 )=-2.54193944373808E-04
f(138)=f( 10 00 12 )=-1.43680084374256E-03
f(139)=f( 10 00 03 )=-0.10479626465914
f(140)=f( 04 00 00 )=  13093.089425509
f(141)=f( 03 10 00 )= 4.89253100381726E-09
f(142)=f( 03 01 00 )= 6.66471433660047E-09
f(143)=f( 03 00 10 )= 0.35229448191160
f(144)=f( 03 00 01 )=  330.76671162514
f(145)=f( 02 20 00 )= -12389.381627853
f(146)=f( 02 11 00 )=  212861.34726061
f(149)=f( 02 02 00 )=  7583.2506661522
f(150)=f( 02 01 10 )= 2.54415888439787E-10
f(152)=f( 02 00 20 )= -2.0076971688682
f(153)=f( 02 00 11 )= -2075.2312117017
f(154)=f( 02 00 02 )=  4.8614096872223
f(155)=f( 01 30 00 )= 7.40792041676429E-09
f(156)=f( 01 21 00 )= 2.05883291330874E-08
f(157)=f( 01 20 10 )=-0.53529042536235
f(158)=f( 01 20 01 )=  93.904554169546
f(159)=f( 01 12 00 )=-2.61215632847230E-08
f(160)=f( 01 11 10 )= -3.6720314500957
f(161)=f( 01 11 01 )= -212.26272557163
f(165)=f( 01 03 00 )=-2.83383161190976E-09
f(166)=f( 01 02 10 )= 0.15940346968810
f(167)=f( 01 02 01 )= -38.870700999373
f(171)=f( 01 00 30 )= 1.01415279468839E-02
f(172)=f( 01 00 21 )= 3.68071933105609E-04
f(173)=f( 01 00 12 )=-6.65930436712669E-04
f(174)=f( 01 00 03 )=-2.13851125596395E-02
f(175)=f( 00 40 00 )=  1764.2193369969
f(176)=f( 00 31 00 )=  9514.4347875228
f(179)=f( 00 22 00 )= -6227.0170299587
f(182)=f( 00 20 20 )= 0.11708889760202
f(183)=f( 00 20 11 )= -1.7999560071408
f(184)=f( 00 20 02 )=-0.22321197489726
f(185)=f( 00 13 00 )= -6956.9519944126
f(188)=f( 00 11 20 )= 3.68164741059018E-03
f(189)=f( 00 11 11 )= 2.29356563091016E-02
f(190)=f( 00 11 02 )=  1.1225239665317
f(195)=f( 00 04 00 )=  311.45300632266
```

```
 f(198)=f( 00 02 20 )= 0.12041502369326
 f(199)=f( 00 02 11 )= -1.8120938764225
 f(200)=f( 00 02 02 )=-1.42919463980606E-02
 f(205)=f( 00 00 40 )=-0.21862430319272
 f(206)=f( 00 00 31 )=  58.625111502592
 f(207)=f( 00 00 22 )= 0.26191390160400
 f(208)=f( 00 00 13 )=  35.107274558903
 f(209)=f( 00 00 04 )= 0.13131966932472

end of MARYLIE run
```

10.9 Production of Lattice Function Plots

The production of lattice function plots requires 4 steps:

1. Specifying at what locations the lattice functions are to be calculated.

2. Calculating various geometric properties of the lattice, such as path length, at these locations and arranging to have these quantities written out onto a file.

3. Arranging to have the lattice functions calculated at these same locations and the desired results written out onto the same file.

4. Using the file generated in steps 2 and 3 above to produce lattice function plots.

These steps will be illustrated in this section for the case of the Proton Storage Ring example described previously in section 2.5.

Step 1 is most easily made in the case that lattice functions are to be computed only at locations between successive elements. If lattice functions are desired with finer spacing along the lattice, it is necessary to split elements. To illustrate element splitting, this example splits dipoles and long and medium-long drifts into 5 pieces, and quadrupoles and sextupoles in two. Splitting parallel faced (rectangular) bends is the most complicated operation. Here use is made of the factorization of section 6.6 that employs incoming and outgoing maps surrounding a normal entry (sector) bend. The normal entry bend in turn is easily split into any number (5 in this case) of wedges.

Compare Exhibit 2.5a in section 2.5 with the Master Input File of this section, Exhibit 10.9c. In Exhibit 10.9c the menu entries *drml, drl, bend, hfq*, and *hdq* found in Exhibit 2.5a are replaced with the entries *cdrml, cdrl, cbend, chfq*, and *chdq*, respectively, where the prefix letter *c* stands for *complete*. Inspection of the rest of the Master Input File in Exhibit 10.9c shows that the "complete" menu entries are not actually used. They merely serve as a reminder of things past and a resource for future use. Instead, as can be verified by inspection, the original menu entries *drml, drl,* $\cdots$ *hfq, hdq* now appear as *lines* in the *#lines* component of the Master Input File. The contents of these lines in turn contain new menu entries. For example, the line *drml* is defined by the statement:

```
 drml
     5*drml/5
```

Reference to the *#menu* component shows that *drml/5* is a drift whose length is 1/5 that of the original medium-long drift *drml*. Thus, the medium-long drift has been split into 5 pieces. Similarly, the sextupoles *hcs* and *vcs* have each been split into two pieces.

As mentioned earlier, splitting of the bend is somewhat more involved. Inspection of the line *bend* shows that it has the contents:

```
bend
    1*inprot      1*infrng      1*ingbdy      5*sbend/5     1*outgbdy  &
    1*outfrng     1*outprot
```

Reference to *#menu* shows that *sbend/5* is a normal entry (sector) bend whose angle is 1/5 that of the original bend. Thus, in view of the factorization described in Exhibit 6.6.1, the bend has also been split into 5 pieces.

Finally, the quadrupoles have been split into leading and trailing halves. For example, the horizontal focusing quadrupole *hfq* is defined by the statement:

```
hfq
    1*inhfq       1*outhfq
```

Reference to *#menu* shows that *inhfq* is a quadrupole of half the original length with a leading fringe field but no trailing fringe field. Similarly, *outhfq* is a half-length quadrupole with no leading fringe field but a trailing fringe field. Thus, the horizontal focusing quadrupole has been split into 2 pieces.

The discussion so far has been devoted to the sometimes necessary, but still ancillary, subject of splitting elements. The main subject is how to specify at what locations in the lattice various functions are to be calculated. Inspection of the *#lines* components of Exhibits 2.5a and 10.9c show that they both contain, in identical form, the lines *nsex, tsex, lsex, half*, and *ring*. In addition, Exhibit 10.9c also contains the lines *%tsex* and *%ring*. Examination of the contents of these lines shows that they are "split" versions of the lines *tsex* and *ring*, respectively, and in addition every entry is surrounded by the character *%*. These lines were prepared in a MaryLie run with the help of the *wcl* commands *wcl10* and *wcl15*. See the *#menu* component of Exhibit 10.9c and section 7.33. Observe that the *#loops* component of Exhibit 10.9c contains the loops *ltsex* and *lring*. Suppose these loops are invoked in turn in *#labor* followed by the commands *wcl15* and *wcl20* as shown below:

```
#labor
    1*fileout
    1*ltsex
    1*wcl10
    1*lring
    1*wcl15
    1*fin
```

Such a MaryLie run produces the contents of *%tsex* in file 10 and the contents of *%ring* in file 15. These contents can then be copied, using an editor, into the *#lines* component of a Master Input File and given the names *%tsex* and *%ring*, as was done to prepare the Master Input File of Exhibit 10.9c.

What is the point of having the lines *%tsex* and *%ring*? As will be seen, they can be used to produce MaryLie runs which compute both lattice functions and geometric quantities at every lattice location where the symbol % appears. Thus, the line *%ring* can be used to produce lattice function plots with data points between every element and within every split element in the entire ring. Similarly, the line *%pring*, which contains *%phalf* which in turn contains *%tsex*, can be used to produce lattice function plots for only the *tsex* portion of half the ring with data points between every element and within every split element in *tsex*.. This completes the description of step 1.

The next step is to actually arrange to have lattice functions calculated at each % location and the desired results written out onto a file. This can be accomplished by defining a *line* whose name is % and whose contents specify the work that is to be done. Suppose, as in this example, that what is desired are various lattice functions at all the locations specified by a % sign. Then, similar to the case of section 10.5, it is first necessary to do some "setup" work in which the total one-turn map is computed and stored. This is accomplished by the entries in the line *setup* as shown below:

```
setup
    1*iden        1*ring        1*stotmap        1*pli0     1*sq          1*iden
```

When *setup* is invoked in *#labor*, the entries *ring* and *stotmap* result in the computation and storage of the one-turn total map. The command *pli0* places path-length information in the fitting array. (It could have been omitted here.) In addition and for convenience, *setup* also contains the entry *sq* that selects the quantities to be written out. See section 8.26. In this case, *sq* receives its instructions from file 20, whose contents are listed in Exhibit 10.9a.

The stage has now been set to present a description of the contents of the line %. As can be verified from the *#lines* component of Exhibit 10.9c, it has the following entries:

```
%
    1*pli0        1*work
```

The entry *pli0* places current path-length information in the fitting/writing array so that it can be written out subsequently in response to the *wsq* command. The entry *work* is the name of a line that specifies the work to be done (in this case the computation of lattice functions).

In order to compute a lattice function at some location, it is first necessary to compute the Poincare surface of section map for this location. This is accomplished by the line *cpssmap* that appears in *work*:

```
work
    1*cpssmap     1*anal        1*wsq         1*restore
```

As can be seen, the entries in *cpssmap* result in storing the current map (*scurmap*), inverting it (*inv*), concatenating the result with the total map (*gtotmap*), and concatenating that result with the current map (*gcurmap*):

```
cpssmap
    1*scurmap     1*inv         1*gtotmap     1*gcurmap
```

That is, as in the example of section 10.5, there is the product relation

$$\text{current Poincare surface of section map} = \\ (\text{current accumulated map})^{-1}\ (\text{total map})\ (\text{current accumulated map}).$$

Once the current Poincare surface of section map has become available, it is analyzed using the command in the line *anal*, in this case a silent *tasm* command (and a *snor* command whose purpose will be described later):

```
anal
     1*stasm       1*snor
```

See the contents of *work* and section 8.2. Next, those results of *pli0* and *anal* (*tasm* in this case) previously selected by *sq* are written to an external file using the *wsq* command. See section 8.27. Finally, again see the contents of *work*, the line *restore* makes the accumulated map the current map:

```
restore
     1*iden        1*gcurmap
```

Steps 2 and 3 have now been accomplished. For example, if the lines *setup* and *%ring* are invoked in *#labor* (see the *#labor* component of Exhibit 10.9e), then at each lattice location in *%ring* marked by a % sign the path-length information and lattice functions specified by *sq* are written to an external file, in this case file 30. See Exhibit 10.9b.

Step 4 is accomplished by processing file 30 with some plotting program to produce various lattice function plots. Figures 9.1 through 9.4 show plots around the ring of the first-order horizontal dispersion functions *dz1* and *dz2* and the horizontal and vertical beta functions.

Sometimes it may be desirable to have lattice function plots for only some selected part of the lattice. This is also easily done. For example, suppose a lattice function plot is desired only for a *tsex* portion of the ring. This can be accomplished by replacing (in *#labor*) *%ring* with *%pring*. Figures 9.5 and 9.6 show, for example, plots of the first-order dispersion functions for just the final *tsex* portion of the ring.

In the examples presented so far all the lattice functions have been the usual *linear* lattice functions. MaryLie can also compute and produce *nonlinear* lattice functions. All lattice functions are various entries in or functions of the entries in the transforming map $\mathcal{A}$. This map is available through the commands *snor* and *dnor*. See sections 8.8 and 8.9. Now it can be understood why in this example a *snor* command is also part of the *anal* line and why, to illustrate the case of a nonlinear lattice function, the *sq* command has been instructed to select also the f(28) entry in $\mathcal{A}$ (available in buffer 1). This entry is the coefficient of the X^3 generator. See sections 8.8 and 14.1 and Exhibit 10.9a. Figure 9.7 shows this nonlinear lattice function for the final *tsex* portion of the ring.

Upon examining figures 9.1, 9.2, 9.3, 9.5, 9.6, and 9.7, the reader might wonder about the origin of what appear to be anomalous spikes. They are an artifact of the splitting process for dipoles. See section 10.13.1 for further explanation and how they can be avoided.

Figure 10.9.1: Plot of the first-order "X" dispersion function *dz1* for the entire ring.

Figure 10.9.2: Plot of the first-order "P_x" dispersion function *dz2* for the entire ring.

Figure 10.9.3: Plot of the horizontal beta function bx for the entire ring.

Figure 10.9.4: Plot of the vertical beta function *by* for the entire ring.

Figure 10.9.5: Plot of the first-order "X" dispersion function *dz1* for the final *tsex* portion of the ring.

Figure 10.9.6: Plot of the first-order "P_x" dispersion function *dz2* for the final *tsex* portion of the ring.

Figure 10.9.7: Plot, for the final *tsex* portion of the ring, of the nonlinear lattice function that is the coefficient of the X^3 generator in $\mathcal{A}$.

```
Exhibit 10.9a
Instructions in file 20 for the sq command:

 1  bf1( 28)
 2       dz1
 3       dz2
 4        bx
 5        by
 6        pl
 #


Exhibit 10.9b
First few lines of file 30 written as a result of the wsq command.
In agreement with Exhibit 10.9a above, the columns are bf1(28),
dz1, dz2, bx, by, and pl, respectively:

  2.97958E-03  3.26669E+00 -3.44101E-01  6.43759E+00  6.23191E+00  0.00000E+00
  2.97958E-03  3.10933E+00 -3.44101E-01  5.66815E+00  7.08761E+00  4.57292E-01
  2.97958E-03  2.95198E+00 -3.44101E-01  5.01700E+00  8.06134E+00  9.14584E-01
  2.97958E-03  2.79462E+00 -3.44101E-01  4.48413E+00  9.15310E+00  1.37188E+00
  2.97958E-03  2.63727E+00 -3.44101E-01  4.06956E+00  1.03629E+01  1.82917E+00
  2.97958E-03  2.47991E+00 -3.44101E-01  3.77328E+00  1.16907E+01  2.28646E+00
 -1.05970E-02  2.42416E+00 -1.02875E-01  3.75309E+00  1.21683E+01  2.53646E+00
 -2.54879E-02  2.42827E+00  1.35811E-01  3.95404E+00  1.20626E+01  2.78646E+00

Exhibit 10.9c
***MARYLIE 3.0***
Prerelease Development Version 6/17/99
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.9
      This is a MaryLie run that employs the pli and other type codes to
 produce lattice function plot data using the PSR as an example.  The
 dipoles and the long and medium-long drifts are split into 5 pieces.
 Quads and sextupoles are split in two.  This run generates lattice function
 plot data for the entire ring.  If the "labor" component is modified to
 take the form shown below, the run will generate lattice function plot
 data for just the final tsex portion of the ring:

 labor
   fileout
   setup
   %pring
   fin

#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
```

```
  1.00000000000000
#menu
 pli0     pli
  0.000000000000000E+00
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 cdrml    drft
   1.48646000000000
 drml/5   drft
  0.297292000000000
 cdrl     drft
   2.28646000000000
 drl/5    drft
  0.457292000000000
 cbend    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 inprot   prot
   18.0000000000000       1.00000000000000
 outprot  prot
   18.0000000000000       2.00000000000000
 infrng   frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       1.00000000000000
 outfrng  frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       2.00000000000000
 ingbdy   gbdy
  0.000000000000000E+00   18.0000000000000      0.000000000000000E+00
   1.20000000000000
 outgbdy  gbdy
  0.000000000000000E+00  0.000000000000000E+00   18.0000000000000
   1.20000000000000
 sbend/5  nbnd
   7.20000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000      0.000000000000000E+00  0.000000000000000E+00
 chfq     quad
  0.500000000000000       2.72000000000000       1.00000000000000
   1.00000000000000
 inhfq    quad
  0.250000000000000       2.72000000000000       1.00000000000000
  0.000000000000000E+00
 outhfq   quad
  0.250000000000000       2.72000000000000      0.000000000000000E+00
   1.00000000000000
 chdq     quad
  0.500000000000000      -1.92000000000000       1.00000000000000
   1.00000000000000
 inhdq    quad
  0.250000000000000      -1.92000000000000       1.00000000000000
  0.000000000000000E+00
```

```
 outhdq   quad
  0.250000000000000      -1.92000000000000      0.000000000000000E+00
   1.00000000000000
 chcs     sext
  0.500000000000000      0.000000000000000E+00
 hcs/2    sext
  0.250000000000000      0.000000000000000E+00
 cvcs     sext
  0.500000000000000      0.000000000000000E+00
 vcs/2    sext
  0.250000000000000      0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 tasm     tasm
   2.00000000000000      1.000000000000000E-03   12.0000000000000
  0.000000000000000E+00   10.0000000000000      0.000000000000000E+00
 stasm    tasm
   2.00000000000000      1.000000000000000E-03   2.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 snor     snor
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00
 iden     iden
 inv      inv
 stotmap  stm
   1.00000000000000
 gtotmap  gtm
   1.00000000000000       1.00000000000000
 scurmap  stm
   2.00000000000000
 gcurmap  gtm
   1.00000000000000       2.00000000000000
 sq       sq
   20.0000000000000      0.000000000000000E+00   1.00000000000000
   1.00000000000000
 wsq      wsq
   1.00000000000000       1.00000000000000       30.0000000000000
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 wcl10    wcl
   3.00000000000000       10.0000000000000       3.00000000000000
 wcl15    wcl
   3.00000000000000       15.0000000000000       3.00000000000000
 setzero  zer
  0.000000000000000E+00  1.000000000000000E-10  0.000000000000000E+00
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 fin      end
#lines
 drml
     5*drml/5
 drl
     5*drl/5
```

```
bend
    1*inprot      1*infrng      1*ingbdy      5*sbend/5     1*outgbdy  &
    1*outfrng     1*outprot
hfq
    1*inhfq       1*outhfq
hdq
    1*inhdq       1*outhdq
hcs
    2*hcs/2
vcs
    2*vcs/2
nsex
    1*drl         1*hdq         1*drs         1*bend        1*drs      &
    1*hfq         1*drl
tsex
    1*drl         1*hdq         1*drs         1*bend        1*drs      &
    1*hfq         1*drvs        1*hcs         1*drml
lsex
    1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
    1*bend        1*drs         1*hfq         1*drl
half
    1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
ring
    2*half
%tsex
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*inhdq       1*%           1*outhdq      1*%        &
    1*drs         1*%           1*inprot      1*%           1*infrng   &
    1*%           1*ingbdy      1*%           1*sbend/5     1*%        &
    1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
    1*%           1*sbend/5     1*%           1*outgbdy     1*%        &
    1*outfrng     1*%           1*outprot     1*%           1*drs      &
    1*%           1*inhfq       1*%           1*outhfq      1*%        &
    1*drvs        1*%           1*hcs/2       1*%           1*hcs/2    &
    1*%           1*drml/5      1*%           1*drml/5      1*%        &
    1*drml/5      1*%           1*drml/5      1*%           1*drml/5   &
    1*%
%ring
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*inhdq       1*%           1*outhdq      1*%        &
    1*drs         1*%           1*inprot      1*%           1*infrng   &
    1*%           1*ingbdy      1*%           1*sbend/5     1*%        &
    1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
    1*%           1*sbend/5     1*%           1*outgbdy     1*%        &
    1*outfrng     1*%           1*outprot     1*%           1*drs      &
    1*%           1*inhfq       1*%           1*outhfq      1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*inhdq       1*%           1*outhdq      1*%           1*drs      &
```

```
1*%          1*inprot     1*%          1*infrng     1*%        &
1*ingbdy     1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*outgbdy    1*%          1*outfrng  &
1*%          1*outprot    1*%          1*drs        1*%        &
1*inhfq      1*%          1*outhfq     1*%          1*drvs     &
1*%          1*hcs/2      1*%          1*hcs/2      1*%        &
1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
1*%          1*drml/5     1*%          1*drml/5     1*%        &
1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
1*%          1*drml/5     1*%          1*drml/5     1*%        &
1*vcs/2      1*%          1*vcs/2      1*%          1*drvs     &
1*%          1*inhdq      1*%          1*outhdq     1*%        &
1*drs        1*%          1*inprot     1*%          1*infrng   &
1*%          1*ingbdy     1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*outgbdy    1*%        &
1*outfrng    1*%          1*outprot    1*%          1*drs      &
1*%          1*inhfq      1*%          1*outhfq     1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*inhdq      1*%          1*outhdq     1*%          1*drs      &
1*%          1*inprot     1*%          1*infrng     1*%        &
1*ingbdy     1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*outgbdy    1*%          1*outfrng  &
1*%          1*outprot    1*%          1*drs        1*%        &
1*inhfq      1*%          1*outhfq     1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*inhdq    &
1*%          1*outhdq     1*%          1*drs        1*%        &
1*inprot     1*%          1*infrng     1*%          1*ingbdy   &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*outgbdy    1*%          1*outfrng    1*%        &
1*outprot    1*%          1*drs        1*%          1*inhfq    &
1*%          1*outhfq     1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*inhdq      1*%        &
1*outhdq     1*%          1*drs        1*%          1*inprot   &
1*%          1*infrng     1*%          1*ingbdy     1*%        &
1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
1*outgbdy    1*%          1*outfrng    1*%          1*outprot  &
1*%          1*drs        1*%          1*inhfq      1*%        &
1*outhfq     1*%          1*drl/5      1*%          1*drl/5    &
1*%          1*drl/5      1*%          1*drl/5      1*%        &
```

```
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*inhdq       1*%           1*outhdq   &
    1*%           1*drs         1*%           1*inprot      1*%        &
    1*infrng      1*%           1*ingbdy      1*%           1*sbend/5  &
    1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
    1*sbend/5     1*%           1*sbend/5     1*%           1*outgbdy  &
    1*%           1*outfrng     1*%           1*outprot     1*%        &
    1*drs         1*%           1*inhfq       1*%           1*outhfq   &
    1*%           1*drvs        1*%           1*hcs/2       1*%        &
    1*hcs/2       1*%           1*drml/5      1*%           1*drml/5   &
    1*%           1*drml/5      1*%           1*drml/5      1*%        &
    1*drml/5      1*%           1*drml/5      1*%           1*drml/5   &
    1*%           1*drml/5      1*%           1*drml/5      1*%        &
    1*drml/5      1*%           1*vcs/2       1*%           1*vcs/2    &
    1*%           1*drvs        1*%           1*inhdq       1*%        &
    1*outhdq      1*%           1*drs         1*%           1*inprot   &
    1*%           1*infrng      1*%           1*ingbdy      1*%        &
    1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
    1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
    1*outgbdy     1*%           1*outfrng     1*%           1*outprot  &
    1*%           1*drs         1*%           1*inhfq       1*%        &
    1*outhfq      1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*inhdq       1*%           1*outhdq   &
    1*%           1*drs         1*%           1*inprot      1*%        &
    1*infrng      1*%           1*ingbdy      1*%           1*sbend/5  &
    1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
    1*sbend/5     1*%           1*sbend/5     1*%           1*outgbdy  &
    1*%           1*outfrng     1*%           1*outprot     1*%        &
    1*drs         1*%           1*inhfq       1*%           1*outhfq   &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*inhdq       1*%           1*outhdq      1*%        &
    1*drs         1*%           1*inprot      1*%           1*infrng   &
    1*%           1*ingbdy      1*%           1*sbend/5     1*%        &
    1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
    1*%           1*sbend/5     1*%           1*outgbdy     1*%        &
    1*outfrng     1*%           1*outprot     1*%           1*drs      &
    1*%           1*inhfq       1*%           1*outhfq      1*%        &
    1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
    1*%           1*drl/5       1*%           1*drl/5       1*%
%phalf
    1*nsex        1*%tsex       1*lsex        1*nsex        1*nsex
%pring
    1*half        1*%phalf
setup
    1*iden        1*ring        1*stotmap     1*pli0        1*sq       &
    1*iden
```

```
 %
     1*pli0        1*work
 work
     1*cpssmap     1*anal        1*wsq         1*restore
 cpssmap
     1*scurmap     1*inv         1*gtotmap     1*gcurmap
 restore
     1*iden        1*gcurmap
 anal
     1*stasm       1*snor
#lumps
#loops
 ltsex
     1*tsex
 lring
     1*ring
#labor
    1*fileout
    1*setup
    1*%ring
    1*fin

*******************************
* Response to the sq command: *
*******************************

 In subroutine sq
accept  1:    bf1( 28)
accept  2:         dz1
accept  3:         dz2
accept  4:          bx
accept  5:          by
accept  6:          pl

 Aims/quantities selected :
No.     item      present value
----------------------------------
 1  bf1( 28) =      0.000000000E+00
 2       dz1 =      0.000000000E+00
 3       dz2 =      0.000000000E+00
 4        bx =      0.000000000E+00
 5        by =      0.000000000E+00
 6        pl =       90.2240000

end of MARYLIE run
```

10.10 Production of Ray Plots

For some purposes it is useful to plot a limited number of rays as they pass through a system. The production of such ray plot data is illustrated in this section for the case of the spot-forming system of section 2.2.

The production of ray plots requires 4 steps quite analogous to those described in section 10.9 for the production of lattice function plots:

1. Specifying at what lattice locations ray (phase-space) coordinates are to be calculated.

2. Calculating various geometric properties of the lattice, such as path length, at these locations and arranging to have these quantities written out onto a file.

3. Arranging to have the ray coordinates calculated at these same locations and the desired results written out onto the same file.

4. Using the file generated in steps 2 and 3 above to produce ray plots.

Step 1 in this case consists of deciding that, in order to produce a detailed plot, the quads and short drifts of section 2.2 should be sliced in 10 pieces, and the long drift should be sliced in 20 pieces. See the *#menu* and *#lines* components of Exhibit 10.10f. (We remark at this point that a *dims* command with user name *aperture* has also been placed in the menu and various lines. It is of no consequence for the purposes of this section, but is important for section 10.12) The original drift *drs* is renamed *cdrs* for reference purposes, an element *drs/10* with 1/10 the length is added to the menu, and an equivalent line with the original name *drs* is added to the *#lines* component:

```
drs
    10*drs/10
```

An analogous action is taken for *drl*, this time with 20 slices:

```
drl
    20*drl/20
```

The quadrupole specifications are also given by lines. For example, the horizontal focusing quadrupole *hfq* is defined by the line

```
hfq
     1*inhfq      10*hfq/10      1*outhfq
```

Here the elements *inhfq* and *outhfq* specify leading and trailing quadrupole fringe fields, respectively, and the element *hfq/10* has no fringe fields. See Exhibit 10.10f and section 6.9. Finally, a loop *lspot* with contents *spot* was invoked in *#labor* in an earlier MaryLie run along with the command *wcl10* to produce and write on file 10 the contents of the line *%spot*:

```
#labor
    1*fileout
    1*lspot
    1*wcl10
    1*end
```

The contents of file 10 were subsequently copied with an editor into the *#lines* component of the Master Input File and give the name *%spot*. See the *#lines* component of Exhibit 10.10f and section 7.33.

Steps 2 and 3 in the case of ray plots are relatively easy. One defines a line whose name is % and whose content is *work*.

```
%
    1*work
```

The entity *work* is in turn defined by the line

```
work
    1*rt          1*scmap       1*gtmap       1*gcmap       1*stmap    &
    1*pli0        1*wsq         1*iden
```

The net effect is that *work* is invoked at each appearance of the % symbol in the line *%spot* (see Exhibit 10.10f). When *work* is invoked a ray trace is first performed (due to the command *rt* appearing in *work*) using what we will call the *current* map. Here the current map is the map for the (current) element that occurs just before % in the line *%spot*. This map is then stored (*scmap*). Next the *total* map for all the elements prior to the current element is retrieved from storage (*gtmap*). Then the current map is brought back from storage (*gcmap*), concatenated with the prior total map to form the new total map, and the results stored for future use (*stmap*). Accumulated path length is then recorded (*pli0*) for this accumulated total map. Now both ray-trace and path-length information have been placed in the fitting/writing array. The selected contents of this array are then written out (*wsq*) to an external file (in this case file 30). Finally, the existing map is set back to the identity (*iden*), in preparation for the next element.

Step 4 is accomplished, as before, by processing file 30 with some plotting program to produce plots of ray-trace data as a function of path length.

Suppose, as is done in this example, one wishes to plot some rays that are in the horizontal (X) plane and initially parallel to the axis. The line *setup*, which is invoked at the beginning of *#labor* (see Exhibit 10.10f) reads in the initial conditions for these rays, initializes the total map to the identity, and selects the quantities to be written later by *wsq*:

```
setup
    1*raysin      1*iden      1*stmap      1*sq
```

Exhibit 10.10a shows the initial conditions file read in by *raysin*, and Exhibit 10.10b shows the instructions for the command *sq*. Exhibit 10.10c shows the beginning of file 30 in this case produced in response to the *wsq* commands. Figure 10.1 shows the associated ray plot produced by plotting the contents of this file 30.

Alternatively, if one wishes to examine rays in the vertical (Y) plane, one uses the initial conditions shown in Exhibit 10.10d and the instructions shown in Exhibit 10.10e. Figure 10.2 shows the ray plot produced in this case.

Figure 10.10.1: Spot-forming system ray plot for 4 initial rays launched in the horizontal plane.

Figure 10.10.2: Spot-forming system ray plot for 4 initial rays launched in the vertical plane.

Exhibit 10.10a
Contents of file 13 that specifies initial
conditions for 4 rays in the horizontal plane:

```
-.2 0 0 0 0 0
+.2 0 0 0 0 0
-.1 0 0 0 0 0
+.1 0 0 0 0 0
```

Exhibit 10.10b
Contents of file 20 that specifies instructions for the sq command
for writing out horizontal position coordinates for the 4 rays
whose initial conditions are specified in Exhibit 10.10a, and for
writing out path-length data:

```
 1  z(  1,1)
 2  z(  2,1)
 3  z(  3,1)
 4  z(  4,1)
 5        pl
```

Exhibit 10.10c
First few lines of file 30 that contains data written in response
to the wsq command. In accord with the order presented in Exhibit
10.10b, the first 4 columns are the horizontal (X) position
coordinates of rays 1 through 4, respectively, and the last column
gives path-length data:

```
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  0.00000E+00
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  5.00000E-02
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  1.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  1.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  2.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  2.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  3.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  3.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  4.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  4.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  5.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  5.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  6.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  6.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  7.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  7.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  8.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  8.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  9.00000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  9.50000E-01
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  1.00000E+00
 -2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01  1.00000E+00
 -1.99868E-01  1.99868E-01 -9.99132E-02  9.99132E-02  1.15000E+00
```

```
 -1.99306E-01  1.99306E-01 -9.96320E-02  9.96320E-02  1.30000E+00
 -1.98369E-01  1.98369E-01 -9.91641E-02  9.91641E-02  1.45000E+00
```

Exhibit 10.10d
Contents of file 13 that specifies initial
conditions for 4 rays in the vertical plane:

```
0 0 -.2 0 0 0
0 0 +.2 0 0 0
0 0 +.1 0 0 0
0 0 -.1 0 0 0
```

Exhibit 10.10e
Contents of file 20 that specifies instructions for the sq command
for writing out vertical position coordinates for the 4 rays
whose initial conditions are specified in Exhibit 10.10d, and for
writing out path-length data:

```
 1  z(  1,3)
 2  z(  2,3)
 3  z(  3,3)
 4  z(  4,3)
 5        pl
 #
```

Exhibit 10.10f

```
***MARYLIE 3.0***
Prerelease Development Version 6/17/99
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.10
      This is a MARYLIE run that demonstrates the production of ray plot data
 for the simple spot forming system of Section 2.2.  To do this all quads and
 drifts are split in 10 pieces, except for the final long drift which is split
 in 20 pieces.  The system is also slightly modified by preceding it with 2
 short drifts.   More precisely, since the elements are split in pieces,
 tracking is done piece by piece.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 pli0     pli
  0.000000000000000E+00
```

```
cdrs     drft
 0.500000000000000
cdrl     drft
  20.0026000000000
chfq     quad
  1.50000000000000      8.630000000000000E-02   1.00000000000000
  1.00000000000000
chdq     quad
  3.00000000000000     -8.289450000000000E-02   1.00000000000000
  1.00000000000000
drs/10   drft
 5.000000000000000E-02
drl/20   drft
  1.00013000000000
inhfq    quad
 0.000000000000000E+00  8.630000000000000E-02   1.00000000000000
 0.000000000000000E+00
outhfq   quad
 0.000000000000000E+00  8.630000000000000E-02  0.000000000000000E+00
  1.00000000000000
inhdq    quad
 0.000000000000000E+00 -8.289450000000000E-02   1.00000000000000
 0.000000000000000E+00
outhdq   quad
 0.000000000000000E+00 -8.289450000000000E-02  0.000000000000000E+00
  1.00000000000000
hfq/10   quad
 0.150000000000000      8.630000000000000E-02  0.000000000000000E+00
 0.000000000000000E+00
hdq/10   quad
 0.300000000000000     -8.289450000000000E-02  0.000000000000000E+00
 0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
inv      inv
wcl10    wcl
  3.00000000000000       10.0000000000000       3.00000000000000
iden     iden
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
rt       rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
sq       sq
  20.0000000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
wsq      wsq
  1.00000000000000       1.00000000000000       30.0000000000000
  1.00000000000000       2.00000000000000      0.000000000000000E+00
```

```
 scmap    stm
   1.00000000000000
 stmap    stm
   2.00000000000000
 gcmap    gtm
   1.00000000000000       1.00000000000000
 gtmap    gtm
   2.00000000000000       2.00000000000000
 end      end
#lines
 hfq
     1*inhfq      10*hfq/10      1*outhfq
 hdq
     1*inhdq      10*hdq/10      1*outhdq
 drs
    10*drs/10
 drl
    20*drl/20
 trip
     1*hfq         1*drs         1*hdq         1*drs         1*hfq
 spot
     2*drs         1*trip        1*drl
 ctrip
     1*chfq        1*cdrs        1*chdq        1*cdrs        1*chfq
 cspot
     2*cdrs        1*ctrip       1*cdrl
 %spot
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*inhfq       1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*outhfq      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*inhdq       1*%           1*hdq/10      1*%           1*hdq/10   &
     1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
     1*hdq/10      1*%           1*hdq/10      1*%           1*hdq/10   &
     1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
     1*hdq/10      1*%           1*outhdq      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*inhfq    &
```

```
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*outhfq      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%
 %
     1*work
 work
     1*rt          1*scmap       1*gtmap       1*gcmap       1*stmap    &
     1*pli0        1*wsq         1*iden
 setup
     1*raysin      1*iden        1*stmap       1*sq
#lumps
#loops
 lspot
     1*spot
#labor
    1*fileout
    1*setup
    1*%spot
    1*end

*******************************
* Response to raysin command: *
*******************************

    4 ray(s) read in from file  13

*******************************
* Response to the command sq: *
*******************************

 In subroutine sq
accept  1:    z(  1,1)
accept  2:    z(  2,1)
accept  3:    z(  3,1)
accept  4:    z(  4,1)
accept  5:          pl

 Aims/quantities selected :
No.     item      present value
-----------------------------------
 1  z(  1,1) =     -0.200000000
 2  z(  2,1) =      0.200000000
 3  z(  3,1) =     -0.100000000
```

```
 4  z(  4,1) =      0.100000000
 5        pl =      0.000000000E+00

end of MARYLIE run
```

10.11 Production of Beam Moment Plots

The production of beam moment plots is similar to the production of lattice function and ray plots, and again requires 4 steps:

1. Specifying at what lattice locations beam moments or rms spreads are to be calculated.
2. Calculating various geometric properties of the lattice, such as path length, at these locations and arranging to have these quantities written out onto a file.
3. Arranging to have the beam moments calculated at these same locations and the desired results written out onto the same file.
4. Using the file generated in steps 2 and 3 above to produce beam moment plots.

The production of beam moment plot data is illustrated in this section for the case of the spot-forming system of sections 2.2 and 10.10. It follows that steps 1 and 4 above are the same as those required to produce ray plots for the spot-forming system. (See section 10.10).

Steps 2 and 3 require further explanation. Examination of the *#labor* component of Exhibit 10.11c shows that it is identical to that in Exhibit 10.10f. However, inspection of the *#lines* component shows that there is now the line *setup1* that has the contents

```
setup1
     1*momgen      1*sq
```

and that the line *momgen*, which contains the commands for generating initial moments, has the contents

```
momgen
     1*ps3         1*bgen        1*gbuf2       1*mapout      1*stm1     &
     1*iden        1*tws         1*mapout      1*snor        1*gbuf1    &
     1*mapout      1*amap        1*gbuf1       1*mapout      1*stm1     &
     1*iden
```

The contents of *momgen* are somewhat analogous to those in the line *makeic* in Exhibit 10.7a. The commands *ps3* and *bgen* now are used to generate moments which are fetched, exhibited at the terminal, and placed in storage location 1 with the commands *gbuf2*, *mapout*, and *stm1*. See sections 8.14 and 8.34. Subsequently a linear map with specified twiss parameters is produced by *tws* and exhibited by *mapout*. This map is then analyzed by *snor* to produce a corresponding $\mathcal{A}$ that is fetched and exhibited by *gbuf1* and *mapout*. See section 8.8. The map $\mathcal{A}$ is then applied by *amap* to the moments produced earlier (and stored in location 1) to produce a set of moments matched to *tws*. See section 8.17. The resulting moments are fetched by *gbuf1*, exhibited by *mapout*, and put in storage location 1 by *stm1* for subsequent

use. Finally, the current transfer map is reset to the identity by the command *iden*. Note that all the *mapout* commands are used only to verify intermediate results, and need not be invoked once one is confident all calculations are proceeding properly.

The other difference from the ray plot example is in the line *%*:

```
%
     1*work1
```

It contains in turn the line *work1*

```
work1
     1*pli0       1*samap       1*wsq
```

This line invokes the command *pli0* to put path-length information in the fitting/writing array. It next invokes *samap* (a *silent amap* command) and the expected *wsq* command required to write out results. The *samap* command applies the current total transfer map (which, unlike the case of Exhibit 10.10f of the previous section, is now being accumulated directly) to the initial moments that are in storage location 1 to obtain current transformed moments. Again see section 8.17.

Exhibit 10.11a displays the instructions for *sq*. Reference to the description of *amap* in section 8.17 shows that the buffer 2 matrix contents *bm2(1,1)*, *bm2(2,2)*, *bm2(3,3)*, and *bm2(4,4)* are the root-mean-square spread quantities $\sqrt{\langle X^2\rangle}$, $\sqrt{\langle P_x^2\rangle}$, $\sqrt{\langle Y^2\rangle}$, and $\sqrt{\langle P_y^2\rangle}$, respectively. Also, the buffer 1 entry *bm1(1,2)* is the moment $\langle XP_x\rangle$. Thanks to the *wsq* command, these quantities are written to file 30 along with path length along the design orbit. See Exhibit 10.11b. Figures 11.1 through 11.5 show beam spread and moment plots produced from file 30. Finally, Exhibit 10.11c shows the MaryLie run itself.

Figure 10.11.1: Spot-forming system plot of the rms horizontal position spread $\sqrt{\langle X^2 \rangle}$.

Figure 10.11.2: Spot-forming system plot of the rms horizontal momentum spread $\sqrt{\langle P_x^2 \rangle}$.

Figure 10.11.3: Spot-forming system plot of the rms vertical position spread $\sqrt{\langle Y^2 \rangle}$.

Figure 10.11.4: Spot-forming system plot of the rms vertical momentum spread $\sqrt{\langle P_y^2 \rangle}$.

Figure 10.11.5: Spot-forming system plot of the horizontal moment $\langle XP_x \rangle$.

Exhibit 10.11a
Contents of file 20 that specifies instructions for the sq command
for writing out the spread and moment quantities sqrt<XX>,
sqrt<PxPx>, sqrt<YY>, sqrt<PyPy>, <XPx>, and path length.

```
 1  bm2(1,1)
 2  bm2(2,2)
 3  bm2(3,3)
 4  bm2(4,4)
 5  bm1(1,2)
 6        pl
```

Exhibit 10.11b
First few lines of file 30 that contains data written in response to
the wsq command. In accord with the ordering specified in Exhibit 10.11a
above, the first 4 columns are the spread and moment quantities sqrt<XX>,
sqrt<PxPx>, sqrt<YY>, sqrt<PyPy>, <XPx>, respectively, and the last column
is path length (pl).

```
  2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  0.00000E+00  0.00000E+00
  2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  3.12503E-07  5.00000E-02
  2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  6.25006E-07  1.00000E-01
  2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  9.37509E-07  1.50000E-01
  2.00001E-01  2.50000E-03  2.00001E-01  2.50000E-03  1.25001E-06  2.00000E-01
```

Exhibit 10.11c

```
***MARYLIE 3.0***
Prerelease Development Version 6/17/99
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.11
 This is a MARYLIE run that demonstrates the production of beam moment
 plot data for the simple spot forming system of Section 2.2.  To do this all
 quads and drifts are split in 10 pieces, except for the final long drift
 which is split in 20 pieces.  The system is also slightly modified by
 preceding it with 2 short drifts.

 Transformed moments are computed from the initial moments using the
 accumulated transfer map.  See the contents of the line work1.

 Alternatively, transformed moments could be computed from the
 moments just preceding using just the map for the current element
 piece.  See the contents of the line work2 which could be used in place
 of work1.  (In this case one should also use setup2 in place of setup1.)
 This approach is the "moment" analog of element-by-element (actually
 piece-by-piece) tracking.

 The beam parameters are those for 50 MeV protons.
```

```
#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 pli0     pli
  0.000000000000000E+00
 cdrs     drft
  0.500000000000000
 cdrl     drft
   20.0026000000000
 chfq     quad
   1.50000000000000      8.630000000000000E-02   1.00000000000000
   1.00000000000000
 chdq     quad
   3.00000000000000     -8.289450000000000E-02   1.00000000000000
   1.00000000000000
 drs/10   drft
  5.000000000000000E-02
 drl/20   drft
   1.00013000000000
 inhfq    quad
  0.000000000000000E+00  8.630000000000000E-02   1.00000000000000
  0.000000000000000E+00
 outhfq   quad
  0.000000000000000E+00  8.630000000000000E-02  0.000000000000000E+00
   1.00000000000000
 inhdq    quad
  0.000000000000000E+00 -8.289450000000000E-02   1.00000000000000
  0.000000000000000E+00
 outhdq   quad
  0.000000000000000E+00 -8.289450000000000E-02  0.000000000000000E+00
   1.00000000000000
 hfq/10   quad
  0.150000000000000      8.630000000000000E-02  0.000000000000000E+00
  0.000000000000000E+00
 hdq/10   quad
  0.300000000000000     -8.289450000000000E-02  0.000000000000000E+00
  0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 inv      inv
 wcl10    wcl
   3.00000000000000       10.0000000000000       3.00000000000000
 iden     iden
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 rt       rt
```

```
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
sq       sq
  20.0000000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
wsq      wsq
  1.00000000000000       1.00000000000000       30.0000000000000
  1.00000000000000       2.00000000000000      0.000000000000000E+00
bgen     bgen
  2.00000000000000       2.00000000000000      0.000000000000000E+00
  123.000000000000       1.00000000000000       3.00000000000000
ps3      ps3
 5.000000000000000E-04  5.000000000000000E-04  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
gbuf1    gbuf
  2.00000000000000       1.00000000000000
gbuf2    gbuf
  2.00000000000000       2.00000000000000
twx      twsm
  1.00000000000000       86.0000000000000      0.000000000000000E+00
  80.0000000000000
twy      twsm
  2.00000000000000       96.0000000000000      0.000000000000000E+00
  80.0000000000000
amap     amap
  2.00000000000000       1.00000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
samap    amap
  2.00000000000000      0.000000000000000E+00  -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
stm1     stm
  1.00000000000000
snor     snor
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00
scmap    stm
  2.00000000000000
stmap    stm
  3.00000000000000
gcmap    gtm
  1.00000000000000       2.00000000000000
gtmap    gtm
  2.00000000000000       3.00000000000000
end      end
#lines
hfq
    1*inhfq      10*hfq/10      1*outhfq
hdq
    1*inhdq      10*hdq/10      1*outhdq
drs
   10*drs/10
drl
   20*drl/20
```

```
trip
    1*hfq         1*drs         1*hdq         1*drs         1*hfq
spot
    2*drs         1*trip        1*drl
ctrip
    1*chfq        1*cdrs        1*chdq        1*cdrs        1*chfq
cspot
    2*cdrs        1*ctrip       1*cdrl
%spot
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*inhfq       1*%           1*hfq/10      1*%        &
    1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
    1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
    1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
    1*%           1*hfq/10      1*%           1*outhfq      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*inhdq       1*%           1*hdq/10      1*%           1*hdq/10   &
    1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
    1*hdq/10      1*%           1*hdq/10      1*%           1*hdq/10   &
    1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
    1*hdq/10      1*%           1*outhdq      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
    1*%           1*drs/10      1*%           1*drs/10      1*%        &
    1*drs/10      1*%           1*drs/10      1*%           1*inhfq    &
    1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
    1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
    1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
    1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
    1*%           1*outhfq      1*%           1*drl/20      1*%        &
    1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
    1*%           1*drl/20      1*%           1*drl/20      1*%        &
    1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
    1*%           1*drl/20      1*%           1*drl/20      1*%        &
    1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
    1*%           1*drl/20      1*%           1*drl/20      1*%        &
    1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
    1*%           1*drl/20      1*%
%
    1*work1
work1
    1*pli0        1*samap       1*wsq
work2
```

```
     1*scmap       1*samap       1*gbuf1       1*stm1        1*gtmap    &
     1*gcmap       1*stmap       1*pli0        1*wsq         1*iden
 setup1
     1*momgen      1*sq
 setup2
     1*momgen      1*iden        1*stmap       1*sq
 momgen
     1*ps3         1*bgen        1*gbuf2       1*mapout      1*stm1     &
     1*iden        1*tws         1*mapout      1*snor        1*gbuf1    &
     1*mapout      1*amap        1*gbuf1       1*mapout      1*stm1     &
     1*iden
 tws
     1*twx         1*twy
#lumps
#loops
 lspot
     1*spot
#labor
    1*fileout
    1*setup1
    1*%spot
    1*end

*********************************
* Response to the command bgen: *
*********************************

 analytically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 5.000000000000000E-004  0.000000000000000E+000  5.000000000000000E-004
 values of <y*y>, <y*py>, <py*py>:
 5.000000000000000E-004  0.000000000000000E+000  5.000000000000000E-004
 values of <t*t>, <t*pt>, <pt*pt>:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

*********************************************
* Moments written in "map" form in response *
* to the gbuf2 and mapout commands:         *
*********************************************

matrix for map is :

 5.00000E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  5.00000E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  5.00000E-04  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  5.00000E-04  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 5.00000000000000E-04
 f( 13)=f( 02 00 00 )= 5.00000000000000E-04
```

```
 f( 18)=f( 00 20 00 )= 5.00000000000000E-04
 f( 22)=f( 00 02 00 )= 5.00000000000000E-04
 f( 84)=f( 40 00 00 )= 5.62500000000000E-07
 f( 90)=f( 22 00 00 )= 1.87500000000000E-07
 f( 95)=f( 20 20 00 )= 1.87500000000000E-07
 f( 99)=f( 20 02 00 )= 1.87500000000000E-07
 f(140)=f( 04 00 00 )= 5.62500000000000E-07
 f(145)=f( 02 20 00 )= 1.87500000000000E-07
 f(149)=f( 02 02 00 )= 1.87500000000000E-07
 f(175)=f( 00 40 00 )= 5.62500000000000E-07
 f(179)=f( 00 22 00 )= 1.87500000000000E-07
 f(195)=f( 00 04 00 )= 5.62500000000000E-07

*********************************
* Response to the stm1 command: *
*********************************

map stored in location   1

************************************
* Map produced by the tws command: *
************************************

matrix for map is :

 6.97565E-02  7.98051E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-1.24696E-02  6.97565E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.04528E-01  7.95618E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.24315E-02 -1.04528E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

*********************************
* Response to the snor command: *
*********************************

det in fxpt is     0.4110E+01

***********************************************
* The matching map script A written in        *
* response to the gbuf1 and mapout commands:  *
***********************************************

matrix for map is :

 8.94427E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.11803E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  8.94427E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.11803E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :

**********************************
* Response to the amap command: *
**********************************

map gotten from location   1

*****************************************
* Moments written in "map" form by amap: *
*****************************************

 transformed moments

matrix for map is :

 4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 4.00000000000000E-02
 f( 13)=f( 02 00 00 )= 6.25000000000000E-06
 f( 18)=f( 00 20 00 )= 4.00000000000000E-02
 f( 22)=f( 00 02 00 )= 6.25000000000000E-06
 f( 84)=f( 40 00 00 )= 3.60000000000000E-03
 f( 90)=f( 22 00 00 )= 1.87500000000000E-07
 f( 95)=f( 20 20 00 )= 1.20000000000000E-03
 f( 99)=f( 20 02 00 )= 1.87500000000000E-07
 f(140)=f( 04 00 00 )= 8.78906250000000E-11
 f(145)=f( 02 20 00 )= 1.87500000000000E-07
 f(149)=f( 02 02 00 )= 2.92968750000000E-11
 f(175)=f( 00 40 00 )= 3.60000000000000E-03
 f(179)=f( 00 22 00 )= 1.87500000000000E-07
 f(195)=f( 00 04 00 )= 8.78906250000000E-11

**************************************************
* Response to the gbuf1 and mapout commands       *
* showing the contents of buffer 1 in "map" form: *
**************************************************

matrix for map is :

 4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
```

```
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 4.00000000000000E-02
 f( 13)=f( 02 00 00 )= 6.25000000000000E-06
 f( 18)=f( 00 20 00 )= 4.00000000000000E-02
 f( 22)=f( 00 02 00 )= 6.25000000000000E-06
 f( 84)=f( 40 00 00 )= 3.60000000000000E-03
 f( 90)=f( 22 00 00 )= 1.87500000000000E-07
 f( 95)=f( 20 20 00 )= 1.20000000000000E-03
 f( 99)=f( 20 02 00 )= 1.87500000000000E-07
 f(140)=f( 04 00 00 )= 8.78906250000000E-11
 f(145)=f( 02 20 00 )= 1.87500000000000E-07
 f(149)=f( 02 02 00 )= 2.92968750000000E-11
 f(175)=f( 00 40 00 )= 3.60000000000000E-03
 f(179)=f( 00 22 00 )= 1.87500000000000E-07
 f(195)=f( 00 04 00 )= 8.78906250000000E-11

**********************************
* Response to the stm1 command: *
**********************************

map stored in location   1

*******************************
* Response to the sq command: *
*******************************

 In subroutine sq
accept  1:    bm2(1,1)
accept  2:    bm2(2,2)
accept  3:    bm2(3,3)
accept  4:    bm2(4,4)
accept  5:    bm1(1,2)
accept  6:          pl

 Aims/quantities selected :
No.     item      present value
----------------------------------
 1  bm2(1,1) =      0.200000000
 2  bm2(2,2) =      2.500000000E-03
 3  bm2(3,3) =      0.200000000
 4  bm2(4,4) =      2.500000000E-03
 5  bm1(1,2) =      0.000000000E+00
 6        pl =      0.000000000E+00

end of MARYLIE run
```

10.12 Production of Composite Plots with Geometry

Sections 10.9, 10.10, and 10.11 illustrated the production of simple plots of various quantities as a function of path length. The path-length data for these plots were obtained using *pli* commands. It is often useful to have various kinds of plots be accompanied by some schematic, similar to those of figure 2.5.2, that illustrates the underlying lattice. Such plots are easily produced, from output files written by the *geom* command, with the aid of the program POSTER. See sections 1.4.2 and 8.38. The *geom* command acts on *loops* to provide detailed geometric information as well as path-length data. The subsections below illustrate the use of the *geom* command. As will be seen, the use of *geom* rather than *pli* to provide path-length information can also sometimes simplify the logic of some entries in the Master Input File.

10.12.1 Lattice Function Plots with Geometry

Figures 10.12.1.1 and 10.12.1.2 show lattice function plots, with additional geometrical data, for the Proton Storage Ring. Exhibit 10.12.1f shows the MaryLie run used to produce these plots. Much of this Master Input File is similar to that of Exhibit 10.9c. However, there are a few differences that are described below.

First, the line *setup* now takes the form

```
setup
     1*iden       1*ring       1*stotmap       1*sq       1*iden
```

There is now no need for a *pli0* command since, as will be seen, path-length data will be provided by *geom*. Correspondingly, there is now no need to select *pl* in response to the *sq* command. Exhibit 10.12.1a shows how file 20 is modified in this case.

Second, the line % now takes the form

```
%
     1*work       1*dp
```

The line *work* specifies the work to be done, and is the same as before in Exhibit 10.9c. The entry *dp* is used in a subsequent computation of geometrical properties of the lattice. See section 6.26.

Third, the *#labor* component of the Master Input File now has the contents

```
#labor
 1*fileout
 1*setup
 1*%ring
 1*l%ring
 1*cgeom
 1*merf
 1*fin
```

Figure 10.12.1.1: Various lattice functions for the Proton Storage Ring treated in section 10.9 accompanied by a schematic of the lattice. The quadrupole gradient is also illustrated.

Figure 10.12.1.2: An expanded version of a portion of figure 10.12.1.1.

The first 3 entries (*1*fileout*, *1*setup*, *1*%ring*) function as before except that only lattice function, and not path-length, information is written on file 30. See Exhibit 10.12.1b. The entry *1*l%ring* invokes a *loop* whose contents is *%ring*. See the *#loops* component of Exhibit 10.12.1f. The entry *cgeom* invokes a line that contains a *geom* command preceded by the commands *ps1* and *ps2* that set the parameters required by *geom*:

```
cgeom
     1*ps1       1*ps2       1*geom
```

So doing causes various geometric properties of the lattice to be computed and written to the external file IFILE, in this case file 40, at each occurrence of the command *dp* (which in turn occurs in the line *%*). See section 8.38. Exhibit 10.12.1c shows the first few lines of file 40. In particular, the first column of file 40 is path length along the design orbit. Finally, the *merf* command in *#labor* merges the first column of file 40 with the remaining columns of file 30 and writes the results on file 50. (See the contents of *#labor* and section 8.40.) Exhibit 10.12.1d shows the first few lines of file 50. Note that the contents of Exhibits 10.12.1d and 10.9b are identical except for the interchange of various columns. Therefore, if desired, this file could be used to produce (with the aid of any of a variety of graphics programs) simple lattice-function plots of the form shown in section 10.9.

In addition to file 40 (which contains path-length and coordinate data), *geom* also writes simple element description and path-length information to file KFILE, in this case file 60. See section 8.38. Exhibit 10.12.1e shows the first few lines of file 60. By processing with POSTER simultaneously file 50 (written by *merf*) and file 60, it is possible to produce plots accompanied by a schematic of the underlying lattice. Figures 12.1.1 and 12.1.2 show composite lattice plots produced in this way for the Proton Storage Ring.

As mentioned in section 10.9, in the lattice function plots of figures 10.9.1 through 10.9.3 and 10.9.5 through 10.9.7 there are some unexpected spikes. Inspection shows that these spikes are also present in figures 12.1.1 and 12.1.2, and from these figures it is clear that these spikes are associated with the entries and exits of dipoles. These spikes are unphysical, and are an artifact of the procedure used to slice (split) the dipoles.

As described in sections 6.6 and 10.9, the slicing of parallel faced bends entails use of the maps *inprot* and *ingbdy* at the entrance to the dipole, and the use of the maps *outgbdy* and *outprot* at the exit of the dipole. It is these maps, which have no length and therefore give results at the same path-length s value, that produce the spikes.

Closer inspection of figures 12.1.1 and 12.1.2 shows that there are no spikes associated with the first dipole, but there are spikes associated with the subsequent dipoles. There are no spikes associated with the first dipole because, for the MaryLie run that produced this figure, the computation and writing out of lattice functions at points between *inprot* and *ingbdy* and between *outgbdy* and *outprot* were suppressed for the first dipole. Comparison of the lines *%ring* in Exhibits 10.9c and 10.12.1f shows one (easily reversible) way of doing this. What has been done is to replace by *$* signs the *%* signs that occur in the first dipole between *inprot* and *ingbdy* and between *outgbdy* and *outprot*. The *$* sign refers to the line

```
$
     1*mark
```

So doing replaces *work* and *dp* (which cause computation and writing of lattice functions and geometric information) by the no effect marker element *mark*. See section 6.25. Alternatively, one could simply remove these % signs.

```
Exhibit 10.12.1a
Instructions in file 20 for the sq command:

 1  bf1( 28)
 2       dz1
 3       dz2
 4        bx
 5        by
 #
```

```
Exhibit 10.12.1b
First few lines of file 30 written as a result of the wsq command.
In agreement with Exhibit 10.12.1a above, the columns are bf1(28),
dz1, dz2, bx, and by, respectively:

 2.97958E-03  3.26669E+00 -3.44101E-01  6.43759E+00  6.23191E+00
 2.97958E-03  3.10933E+00 -3.44101E-01  5.66815E+00  7.08761E+00
 2.97958E-03  2.95198E+00 -3.44101E-01  5.01700E+00  8.06134E+00
 2.97958E-03  2.79462E+00 -3.44101E-01  4.48413E+00  9.15310E+00
 2.97958E-03  2.63727E+00 -3.44101E-01  4.06956E+00  1.03629E+01
 2.97958E-03  2.47991E+00 -3.44101E-01  3.77328E+00  1.16907E+01
-1.05970E-02  2.42416E+00 -1.02875E-01  3.75309E+00  1.21683E+01
-2.54879E-02  2.42827E+00  1.35811E-01  3.95404E+00  1.20626E+01
```

```
Exhibit 10.12.1c
First few lines of file 40 written as a result of the geom command.
The first column is path length along the design orbit:

 0.00000E+00  1.40000E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 4.57292E-01  1.40000E+01  0.00000E+00  4.57292E-01  1.81329E-09  0.00000E+00
 9.14584E-01  1.40000E+01  0.00000E+00  9.14584E-01  3.62659E-09  0.00000E+00
 1.37188E+00  1.40000E+01  0.00000E+00  1.37188E+00  5.43988E-09  0.00000E+00
 1.82917E+00  1.40000E+01  0.00000E+00  1.82917E+00  7.25318E-09  0.00000E+00
 2.28646E+00  1.40000E+01  0.00000E+00  2.28646E+00  9.06647E-09  0.00000E+00
 2.53646E+00  1.40000E+01  0.00000E+00  2.53646E+00  1.00578E-08  0.00000E+00
 2.78646E+00  1.40000E+01  0.00000E+00  2.78646E+00  1.10491E-08  0.00000E+00
```

```
Exhibit 10.12.1d
First few lines of file 50 written as a result of the merf command.
Note that the first column (path length) is the same as that in
Exhibit 10.12.1c above, and the remaining columns (lattice functions) are
the same as in Exhibit 10.12.1b:

  0.0000E+00  0.2980E-02  0.3267E+01 -0.3441E+00  0.6438E+01  0.6232E+01
  0.4573E+00  0.2980E-02  0.3109E+01 -0.3441E+00  0.5668E+01  0.7088E+01
  0.9146E+00  0.2980E-02  0.2952E+01 -0.3441E+00  0.5017E+01  0.8061E+01
  0.1372E+01  0.2980E-02  0.2795E+01 -0.3441E+00  0.4484E+01  0.9153E+01
  0.1829E+01  0.2980E-02  0.2637E+01 -0.3441E+00  0.4070E+01  0.1036E+02
```

```
  0.2286E+01  0.2980E-02  0.2480E+01 -0.3441E+00  0.3773E+01  0.1169E+02
  0.2536E+01 -0.1060E-01  0.2424E+01 -0.1029E+00  0.3753E+01  0.1217E+02
  0.2786E+01 -0.2549E-01  0.2428E+01  0.1358E+00  0.3954E+01  0.1206E+02

Exhibit 10.12.1e
First few lines of file 60 written as a result of the geom command.
This file provides simple element and path-length information:

drl/5     1  1  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  4.57292E-01  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  4.57292E-01  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  9.14584E-01  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  9.14584E-01  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  1.37188E+00  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  1.37188E+00  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  1.82917E+00  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  1.82917E+00  0.00000E+00  0.00000E+00  0.00000E+00
drl/5     1  1  2.28646E+00  0.00000E+00  0.00000E+00  0.00000E+00
inhdq     1  9  2.28646E+00  0.00000E+00  0.00000E+00 -1.92000E+00
inhdq     1  9  2.53646E+00  0.00000E+00  0.00000E+00 -1.92000E+00
outhdq    1  9  2.53646E+00  0.00000E+00  0.00000E+00 -1.92000E+00
outhdq    1  9  2.78646E+00  0.00000E+00  0.00000E+00 -1.92000E+00
drs       1  1  2.78646E+00  0.00000E+00  0.00000E+00  0.00000E+00


Exhibit 10.12.1f
***MARYLIE 3.0***
Prerelease Development Version 6/17/99
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment

 Exhibit 10.12.1

 This is a MaryLie run that employs the geom and other type codes to
 produce lattice function and geometrical data using the PSR as an
 example.  The dipoles and the long and medium-long drifts are split into
 5 pieces.  Quads and sextupoles are split in two.  This run generates
 data for the entire ring.

 For the first dipole at the beginning of the line %ring, the computation
 and writing of lattice function and geometrical data (produced by the
 commands work and dp) is suppressed between inprot and ingbdy at the
 entrance to the dipole, and between outgbdy and outprot at the exit of
 the dipole.  This is accomplished by replacing % by $.  Here % is the
 name of the line

 %
     1*work        1*dp

 and $ is the name of the line
```

```
 $
     1*mark

 So doing removes from the lattice function plots (for the first dipole)
 the unphysical spikes that are an artifact of the slicing (splitting)
 procedure for dipoles.

#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 mark     mark
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 cdrml    drft
   1.48646000000000
 drml/5   drft
  0.297292000000000
 cdrl     drft
   2.28646000000000
 drl/5    drft
  0.457292000000000
 cbend    pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 inprot   prot
   18.0000000000000       1.00000000000000
 outprot  prot
   18.0000000000000       2.00000000000000
 infrng   frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       1.00000000000000
 outfrng  frng
   18.0000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.20000000000000       2.00000000000000
 ingbdy   gbdy
  0.000000000000000E+00   18.0000000000000      0.000000000000000E+00
   1.20000000000000
 outgbdy  gbdy
  0.000000000000000E+00  0.000000000000000E+00   18.0000000000000
   1.20000000000000
 sbend/5  nbnd
   7.20000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000      0.000000000000000E+00  0.000000000000000E+00
 chfq     quad
  0.500000000000000       2.72000000000000       1.00000000000000
   1.00000000000000
 inhfq    quad
```

```
 0.250000000000000       2.72000000000000       1.00000000000000
 0.000000000000000E+00
outhfq   quad
 0.250000000000000       2.72000000000000      0.000000000000000E+00
  1.00000000000000
chdq     quad
 0.500000000000000      -1.92000000000000       1.00000000000000
  1.00000000000000
inhdq    quad
 0.250000000000000      -1.92000000000000       1.00000000000000
 0.000000000000000E+00
outhdq   quad
 0.250000000000000      -1.92000000000000      0.000000000000000E+00
  1.00000000000000
chcs     sext
 0.500000000000000      0.000000000000000E+00
hcs/2    sext
 0.250000000000000      0.000000000000000E+00
cvcs     sext
 0.500000000000000      0.000000000000000E+00
vcs/2    sext
 0.250000000000000      0.000000000000000E+00
fileout  pmif
  1.00000000000000       12.0000000000000       3.00000000000000
tasm     tasm
  2.00000000000000      1.000000000000000E-03   12.0000000000000
 0.000000000000000E+00   10.0000000000000      0.000000000000000E+00
stasm    tasm
  2.00000000000000      1.000000000000000E-03   2.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
snor     snor
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00
iden     iden
inv      inv
dp       dp
stotmap  stm
  1.00000000000000
gtotmap  gtm
  1.00000000000000       1.00000000000000
scurmap  stm
  2.00000000000000
gcurmap  gtm
  1.00000000000000       2.00000000000000
sq       sq
  20.0000000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
wsq      wsq
  1.00000000000000       1.00000000000000       30.0000000000000
  1.00000000000000       2.00000000000000      0.000000000000000E+00
wcl10    wcl
  3.00000000000000       10.0000000000000       3.00000000000000
wcl15    wcl
```

```
  3.00000000000000       15.0000000000000       3.00000000000000
 geom     geom
  3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  1.00000000000000       2.00000000000000      0.000000000000000E+00
 ps1      ps1
  14.0000000000000      0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 ps2      ps2
  40.0000000000000      0.000000000000000E+00   60.0000000000000
  70.0000000000000      0.000000000000000E+00   1.00000000000000
 merf     usr12
  40.0000000000000       30.0000000000000       50.0000000000000
  1.00000000000000       5.00000000000000      0.000000000000000E+00
 setzero  zer
 0.000000000000000E+00  1.000000000000000E-10  0.000000000000000E+00
 mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
 fin      end
#lines
 drml
     5*drml/5
 drl
     5*drl/5
 bend
     1*inprot      1*infrng      1*ingbdy      5*sbend/5     1*outgbdy  &
     1*outfrng     1*outprot
 hfq
     1*inhfq       1*outhfq
 hdq
     1*inhdq       1*outhdq
 hcs
     2*hcs/2
 vcs
     2*vcs/2
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
 %tsex
     1*%           1*drl/5       1*%           1*drl/5       1*%        &
     1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
     1*%           1*inhdq       1*%           1*outhdq      1*%        &
     1*drs         1*%           1*inprot      1*%           1*infrng   &
```

```
    1*%          1*ingbdy     1*%          1*sbend/5    1*%        &
    1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
    1*%          1*sbend/5    1*%          1*outgbdy    1*%        &
    1*outfrng    1*%          1*outprot    1*%          1*drs      &
    1*%          1*inhfq      1*%          1*outhfq     1*%        &
    1*drvs       1*%          1*hcs/2      1*%          1*hcs/2    &
    1*%          1*drml/5     1*%          1*drml/5     1*%        &
    1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
    1*%
%ring
    1*%          1*drl/5      1*%          1*drl/5      1*%        &
    1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
    1*%          1*inhdq      1*%          1*outhdq     1*%        &
    1*drs        1*%          1*inprot     1*$          1*infrng   &
    1*$          1*ingbdy     1*%          1*sbend/5    1*%        &
    1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
    1*%          1*sbend/5    1*%          1*outgbdy    1*$        &
    1*outfrng    1*$          1*outprot    1*%          1*drs      &
    1*%          1*inhfq      1*%          1*outhfq     1*%        &
    1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
    1*%          1*drl/5      1*%          1*drl/5      1*%        &
    1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
    1*%          1*drl/5      1*%          1*drl/5      1*%        &
    1*inhdq      1*%          1*outhdq     1*%          1*drs      &
    1*%          1*inprot     1*%          1*infrng     1*%        &
    1*ingbdy     1*%          1*sbend/5    1*%          1*sbend/5  &
    1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
    1*sbend/5    1*%          1*outgbdy    1*%          1*outfrng  &
    1*%          1*outprot    1*%          1*drs        1*%        &
    1*inhfq      1*%          1*outhfq     1*%          1*drvs     &
    1*%          1*hcs/2      1*%          1*hcs/2      1*%        &
    1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
    1*%          1*drml/5     1*%          1*drml/5     1*%        &
    1*drml/5     1*%          1*drml/5     1*%          1*drml/5   &
    1*%          1*drml/5     1*%          1*drml/5     1*%        &
    1*vcs/2      1*%          1*vcs/2      1*%          1*drvs     &
    1*%          1*inhdq      1*%          1*outhdq     1*%        &
    1*drs        1*%          1*inprot     1*%          1*infrng   &
    1*%          1*ingbdy     1*%          1*sbend/5    1*%        &
    1*sbend/5    1*%          1*sbend/5    1*%          1*sbend/5  &
    1*%          1*sbend/5    1*%          1*outgbdy    1*%        &
    1*outfrng    1*%          1*outprot    1*%          1*drs      &
    1*%          1*inhfq      1*%          1*outhfq     1*%        &
    1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
    1*%          1*drl/5      1*%          1*drl/5      1*%        &
    1*drl/5      1*%          1*drl/5      1*%          1*drl/5    &
    1*%          1*drl/5      1*%          1*drl/5      1*%        &
    1*inhdq      1*%          1*outhdq     1*%          1*drs      &
    1*%          1*inprot     1*%          1*infrng     1*%        &
    1*ingbdy     1*%          1*sbend/5    1*%          1*sbend/5  &
    1*%          1*sbend/5    1*%          1*sbend/5    1*%        &
    1*sbend/5    1*%          1*outgbdy    1*%          1*outfrng  &
    1*%          1*outprot    1*%          1*drs        1*%        &
```

```
1*inhfq       1*%           1*outhfq      1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*drl/5       1*%           1*inhdq    &
1*%           1*outhdq      1*%           1*drs         1*%        &
1*inprot      1*%           1*infrng      1*%           1*ingbdy   &
1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
1*%           1*outgbdy     1*%           1*outfrng     1*%        &
1*outprot     1*%           1*drs         1*%           1*inhfq    &
1*%           1*outhfq      1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*inhdq       1*%        &
1*outhdq      1*%           1*drs         1*%           1*inprot   &
1*%           1*infrng      1*%           1*ingbdy      1*%        &
1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
1*outgbdy     1*%           1*outfrng     1*%           1*outprot  &
1*%           1*drs         1*%           1*inhfq       1*%        &
1*outhfq      1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*inhdq       1*%           1*outhdq   &
1*%           1*drs         1*%           1*inprot      1*%        &
1*infrng      1*%           1*ingbdy      1*%           1*sbend/5  &
1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
1*sbend/5     1*%           1*sbend/5     1*%           1*outgbdy  &
1*%           1*outfrng     1*%           1*outprot     1*%        &
1*drs         1*%           1*inhfq       1*%           1*outhfq   &
1*%           1*drvs        1*%           1*hcs/2       1*%        &
1*hcs/2       1*%           1*drml/5      1*%           1*drml/5   &
1*%           1*drml/5      1*%           1*drml/5      1*%        &
1*drml/5      1*%           1*drml/5      1*%           1*drml/5   &
1*%           1*drml/5      1*%           1*drml/5      1*%        &
1*drml/5      1*%           1*vcs/2       1*%           1*vcs/2    &
1*%           1*drvs        1*%           1*inhdq       1*%        &
1*outhdq      1*%           1*drs         1*%           1*inprot   &
1*%           1*infrng      1*%           1*ingbdy      1*%        &
1*sbend/5     1*%           1*sbend/5     1*%           1*sbend/5  &
1*%           1*sbend/5     1*%           1*sbend/5     1*%        &
1*outgbdy     1*%           1*outfrng     1*%           1*outprot  &
1*%           1*drs         1*%           1*inhfq       1*%        &
1*outhfq      1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*drl/5       1*%           1*drl/5    &
1*%           1*drl/5       1*%           1*drl/5       1*%        &
1*drl/5       1*%           1*inhdq       1*%           1*outhdq   &
1*%           1*drs         1*%           1*inprot      1*%        &
1*infrng      1*%           1*ingbdy      1*%           1*sbend/5  &
```

```
     1*%            1*sbend/5      1*%            1*sbend/5      1*%         &
     1*sbend/5      1*%            1*sbend/5      1*%            1*outgbdy   &
     1*%            1*outfrng      1*%            1*outprot      1*%         &
     1*drs          1*%            1*inhfq        1*%            1*outhfq    &
     1*%            1*drl/5        1*%            1*drl/5        1*%         &
     1*drl/5        1*%            1*drl/5        1*%            1*drl/5     &
     1*%            1*drl/5        1*%            1*drl/5        1*%         &
     1*drl/5        1*%            1*drl/5        1*%            1*drl/5     &
     1*%            1*inhdq        1*%            1*outhdq       1*%         &
     1*drs          1*%            1*inprot       1*%            1*infrng    &
     1*%            1*ingbdy       1*%            1*sbend/5      1*%         &
     1*sbend/5      1*%            1*sbend/5      1*%            1*sbend/5   &
     1*%            1*sbend/5      1*%            1*outgbdy      1*%         &
     1*outfrng      1*%            1*outprot      1*%            1*drs       &
     1*%            1*inhfq        1*%            1*outhfq       1*%         &
     1*drl/5        1*%            1*drl/5        1*%            1*drl/5     &
     1*%            1*drl/5        1*%            1*drl/5        1*%
 %phalf
     1*nsex         1*%tsex        1*lsex         1*nsex         1*nsex
 %pring
     1*half         1*%phalf
 setup
     1*iden         1*ring         1*stotmap      1*sq           1*iden
 %
     1*work         1*dp
 work
     1*cpssmap      1*anal         1*wsq          1*restore
 cpssmap
     1*scurmap      1*inv          1*gtotmap      1*gcurmap
 restore
     1*iden         1*gcurmap
 anal
     1*stasm        1*snor
 $
     1*mark
 cgeom
     1*ps1          1*ps2          1*geom
#lumps
#loops
 ltsex
     1*tsex
 lring
     1*ring
 l%ring
     1*%ring
 l%pring
     1*%pring
#labor
    1*fileout
    1*setup
    1*%ring
    1*l%ring
    1*cgeom
```

```
    1*merf
    1*fin

*******************************
* Response to the sq command: *
*******************************

 In subroutine sq
accept  1:    bf1( 28)
accept  2:         dz1
accept  3:         dz2
accept  4:          bx
accept  5:          by

 Aims/quantities selected :
No.     item      present value
-----------------------------------
 1  bf1( 28) =      0.000000000E+00
 2       dz1 =      0.000000000E+00
 3       dz2 =      0.000000000E+00
 4        bx =      0.000000000E+00
 5        by =      0.000000000E+00

********************************
* Response to the merf command: *
********************************

 number of lines read =         279

end of MARYLIE run
```

10.12.2 Ray Plots with Geometry

Figure 10.12.2.1 shows a ray plot, with additional geometric data, for the spot-forming system of sections 2.2 and 10.10. Exhibit 10.12.2f shows the MaryLie run used to produce this plot. Much of this Master Input File is similar to that of Exhibit 10.10f. However, there are a few differences that are described below.

First, there is no need to store the total map since path-length data will now be produced by a *geom* command. Therefore the line *setup* can take the simpler form

```
setup
     1*raysin     1*sq
```

Correspondingly, there is now no need to select *pl* in response to the *sq* command. Exhibit 10.12.2a shows how file 20 is modified in this case.

Second, the line % now takes the form

```
%
     1*work      1*dp
```

and the line *work*, which specifies the work to be done, can now take the simple form

```
work
     1*rt     1*wsq     1*iden
```

When the line *work* is invoked at each occurrence of the % symbol in the line *%spot* (see Exhibit 10.12.2f) a ray trace will be performed using the current map, selected results from this ray trace will be written to an external file (in this case file 30) by the command *wsq*, and the current map will be reset to the identity map as a result of the command *iden*. The entry *dp* in %, which follows *work*, is used in connection with *geom* in a subsequent computation of geometrical properties of the spot-forming system. See section 6.26.

Third, the *#labor* component of the Master Input File now has the contents

```
#labor
 1*fileout
 1*setup
 1*%spot
 1*l%spot
 1*cgeom
 1*merf
 1*end
```

The first 3 entries (*1*fileout*, *1*setup*, *1*%spot*) function as before except that only ray, and not path-length, information is written on file 30. See Exhibit 10.12.2b. The entry *1*l%spot* invokes a *loop* whose contents is *%spot*. See the *#loops* component of Exhibit 10.12.2f. The entry *cgeom* invokes a line that contains a *geom* command preceded by the commands *ps1* and *ps2* that set the parameters required by geom:

```
cgeom
     1*ps1   1*ps2  1*geom
```

Figure 10.12.2.1: Horizontal plane ray plot for the spot-forming system treated in section 10.10 accompanied by a schematic showing the lengths and locations of quadrupoles and their apertures.

So doing causes various geometric properties of the spot-forming system to be computed and written to the external file IFILE, in this case file 40, at each occurrence of the command *dp* (which in turn occurs in the line *%*). See section 8.38. Exhibit 10.12.2c shows the first few lines of file 40. In particular, the first column of file 40 is path length along the design orbit. Finally the *merf* command in *#labor* merges the first column of file 40 with the remaining columns of file 30 and writes the result on file 50. (See the contents of *#labor* and section 8.40.) Exhibit 10.12.2d shows the first few lines of file 50. Note that the contents of Exhibits 10.12.2d and 10.10c are identical except for the interchange of various columns. Therefore, if desired, this file could be used to produce (with the aid of any of a variety of graphics programs) simple ray plots of the form shown in section 10.10.

In addition to file 40 (which contains path-length and coordinate data), *geom* also writes simple element description and path-length information to the file KFILE, in this case file 60. See section 8.38. Exhibit 10.12.2e shows the first few lines of file 60. By processing with POSTER simultaneously file 50 (written by *merf*) and file 60, it is possible to produce ray plots accompanied by geometrical information about the associated beam line. Figure 10.12.2 shows a composite ray plot produced in this way for the spot-forming system of sections 2.2 and 10.10. This composite plot displays the lengths, locations, and apertures of the quadrupoles in the beam line as well as ray data.

There is one last detail to be described. Observe, by comparing Exhibits 10.10f and 10.12.2f, that the lines *hfq* and *hdq* in this case contain a *dims* command with user name *aperture*. This command is recognized by *geom*, affects the contents of the file 60 it writes, and is used to specify the aperture size for the quadrupoles that appear in figure 10.12.2. See section 7.40.

```
Exhibit 10.12.2a
Instructions in file 20 for the sq command:

 1  z(  1,1)
 2  z(  2,1)
 3  z(  3,1)
 4  z(  4,1)
 #


Exhibit 10.12.2b
First few lines of file 30 that contains data written in response
to the wsq command.  In accord with the order presented in Exhibit
10.12.2a, the columns are the horizontal (X) position coordinates
of rays 1 through 4, respectively:

-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
```

```
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00000E-01  2.00000E-01 -1.00000E-01  1.00000E-01
-2.00056E-01  2.00056E-01 -1.00007E-01  1.00007E-01
-1.99868E-01  1.99868E-01 -9.99132E-02  9.99132E-02
-1.99306E-01  1.99306E-01 -9.96320E-02  9.96320E-02
-1.98369E-01  1.98369E-01 -9.91641E-02  9.91641E-02
```

Exhibit 10.12.2c
First few lines of file 40 that contains data written in response to
the geom command. Column 1 is path length along the design orbit:

```
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 5.00000E-02  0.00000E+00  0.00000E+00  5.00000E-02  5.31071E-10  0.00000E+00
 1.00000E-01  0.00000E+00  0.00000E+00  1.00000E-01  1.06214E-09  0.00000E+00
 1.50000E-01  0.00000E+00  0.00000E+00  1.50000E-01  1.59321E-09  0.00000E+00
 2.00000E-01  0.00000E+00  0.00000E+00  2.00000E-01  2.12429E-09  0.00000E+00
 2.50000E-01  0.00000E+00  0.00000E+00  2.50000E-01  2.65536E-09  0.00000E+00
 3.00000E-01  0.00000E+00  0.00000E+00  3.00000E-01  3.18643E-09  0.00000E+00
 3.50000E-01  0.00000E+00  0.00000E+00  3.50000E-01  3.71750E-09  0.00000E+00
 4.00000E-01  0.00000E+00  0.00000E+00  4.00000E-01  4.24857E-09  0.00000E+00
 4.50000E-01  0.00000E+00  0.00000E+00  4.50000E-01  4.77964E-09  0.00000E+00
 5.00000E-01  0.00000E+00  0.00000E+00  5.00000E-01  5.31071E-09  0.00000E+00
 5.50000E-01  0.00000E+00  0.00000E+00  5.50000E-01  5.84179E-09  0.00000E+00
 6.00000E-01  0.00000E+00  0.00000E+00  6.00000E-01  6.37286E-09  0.00000E+00
 6.50000E-01  0.00000E+00  0.00000E+00  6.50000E-01  6.90393E-09  0.00000E+00
 7.00000E-01  0.00000E+00  0.00000E+00  7.00000E-01  7.43500E-09  0.00000E+00
 7.50000E-01  0.00000E+00  0.00000E+00  7.50000E-01  7.96607E-09  0.00000E+00
 8.00000E-01  0.00000E+00  0.00000E+00  8.00000E-01  8.49714E-09  0.00000E+00
 8.50000E-01  0.00000E+00  0.00000E+00  8.50000E-01  9.02821E-09  0.00000E+00
 9.00000E-01  0.00000E+00  0.00000E+00  9.00000E-01  9.55929E-09  0.00000E+00
 9.50000E-01  0.00000E+00  0.00000E+00  9.50000E-01  1.00904E-08  0.00000E+00
 1.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.06214E-08  0.00000E+00
 1.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.06214E-08  0.00000E+00
 1.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  1.06214E-08  0.00000E+00
 1.15000E+00  0.00000E+00  0.00000E+00  1.15000E+00  1.22146E-08  0.00000E+00
 1.30000E+00  0.00000E+00  0.00000E+00  1.30000E+00  1.38079E-08  0.00000E+00
 1.45000E+00  0.00000E+00  0.00000E+00  1.45000E+00  1.54011E-08  0.00000E+00
```

Exhibit 10.12.2d
First few lines of file 50 that contains data written in response to

the merf command. Column 1 is path length along the design orbit, and
the next 4 columnns are the horizontal position coordinates of the 4 rays:

```
  0.0000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.5000E-01 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.1000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.1500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.2000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.2500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.3000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.3500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.4000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.4500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.5000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.5500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.6000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.6500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.7000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.7500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.8000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.8500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.9000E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.9500E+00 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.1000E+01 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.1000E+01 -0.2000E+00  0.2000E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.1000E+01 -0.2001E+00  0.2001E+00 -0.1000E+00  0.1000E+00  0.0000E+00
  0.1150E+01 -0.1999E+00  0.1999E+00 -0.9991E-01  0.9991E-01  0.0000E+00
  0.1300E+01 -0.1993E+00  0.1993E+00 -0.9963E-01  0.9963E-01  0.0000E+00
  0.1450E+01 -0.1984E+00  0.1984E+00 -0.9916E-01  0.9916E-01  0.0000E+00
```

Exhibit 10.12.2e
First few lines of file 60 that contains data written in response to
the geom command. It contains simple element description and
path-length information:

```
drs/10    1  1  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  2.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  2.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  2.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  2.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  3.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  3.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  3.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  3.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  4.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  4.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
```

```
drs/10    1  1  4.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  4.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  6.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  6.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  6.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  6.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  7.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  7.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  7.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  7.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  8.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  8.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  8.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  8.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  9.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  9.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  9.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  9.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
inhfq     1  9  1.00000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.00000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.15000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.15000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.30000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.30000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.45000E+00  3.00000E-01  0.00000E+00  8.63000E-02
hfq/10    1  9  1.45000E+00  3.00000E-01  0.00000E+00  8.63000E-02


Exhibit 10.12.2f
***MARYLIE 3.0***
Prerelease Development Version 6/17/99
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.12.2
      This is a MARYLIE run that demonstrates the production of ray and
 geometric data for the simple spot forming system of Section 2.2.
 To do this all quads and drifts are split in 10 pieces, except for the
 final long drift which is split in 20 pieces.  The system is also slightly
 modified by preceding it with 2 short drifts.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
```

```
  1.00000000000000
  1.00000000000000
#menu
 cdrs     drft
  0.500000000000000
 cdrl     drft
   20.0026000000000
 chfq     quad
   1.50000000000000      8.630000000000000E-02   1.00000000000000
   1.00000000000000
 chdq     quad
   3.00000000000000     -8.289450000000000E-02   1.00000000000000
   1.00000000000000
 drs/10   drft
  5.000000000000000E-02
 drl/20   drft
   1.00013000000000
 inhfq    quad
  0.000000000000000E+00  8.630000000000000E-02   1.00000000000000
  0.000000000000000E+00
 outhfq   quad
  0.000000000000000E+00  8.630000000000000E-02  0.000000000000000E+00
   1.00000000000000
 inhdq    quad
  0.000000000000000E+00 -8.289450000000000E-02   1.00000000000000
  0.000000000000000E+00
 outhdq   quad
  0.000000000000000E+00 -8.289450000000000E-02  0.000000000000000E+00
   1.00000000000000
 hfq/10   quad
  0.150000000000000      8.630000000000000E-02  0.000000000000000E+00
  0.000000000000000E+00
 hdq/10   quad
  0.300000000000000     -8.289450000000000E-02  0.000000000000000E+00
  0.000000000000000E+00
 aperture dims
  0.300000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 inv      inv
 wcl10    wcl
   3.00000000000000       10.0000000000000       3.00000000000000
 mark     mark
 iden     iden
 raysin   rt
   13.0000000000000       14.0000000000000      -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 rt       rt
  0.000000000000000E+00   14.0000000000000       5.00000000000000
```

```
  1.00000000000000       1.00000000000000      0.000000000000000E+00
 sq       sq
  20.0000000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
 wsq      wsq
  1.00000000000000       1.00000000000000       30.0000000000000
  1.00000000000000       2.00000000000000      0.000000000000000E+00
 dp       dp
 geom     geom
  3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  1.00000000000000       2.00000000000000      0.000000000000000E+00
 ps1      ps1
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 ps2      ps2
  40.0000000000000      0.000000000000000E+00   60.0000000000000
  0.000000000000000E+00  0.000000000000000E+00   1.00000000000000
 merf     usr12
  40.0000000000000       30.0000000000000       50.0000000000000
  1.00000000000000       4.00000000000000      0.000000000000000E+00
 end      end
#lines
 hfq
     1*aperture    1*inhfq      10*hfq/10      1*outhfq
 hdq
     1*aperture    1*inhdq      10*hdq/10      1*outhdq
 drs
    10*drs/10
 drl
    20*drl/20
 trip
     1*hfq         1*drs         1*hdq         1*drs         1*hfq
 spot
     2*drs         1*trip        1*drl
 ctrip
     1*chfq        1*cdrs        1*chdq        1*cdrs        1*chfq
 cspot
     2*cdrs        1*ctrip       1*cdrl
 %spot
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*aperture    1*%           1*inhfq       1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*outhfq      1*%           1*drs/10      1*%           1*drs/10   &
```

```
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*aperture    1*%           1*inhdq    &
     1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
     1*hdq/10      1*%           1*hdq/10      1*%           1*hdq/10   &
     1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
     1*hdq/10      1*%           1*hdq/10      1*%           1*hdq/10   &
     1*%           1*outhdq      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*aperture    1*%        &
     1*inhfq       1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*outhfq      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%
 %
     1*work        1*dp
 work
     1*rt          1*wsq         1*iden
 setup
     1*raysin      1*sq
 cgeom
     1*ps1         1*ps2         1*geom
#lumps
#loops
 lspot
     1*spot
 l%spot
     1*%spot
#labor
    1*fileout
    1*setup
    1*%spot
    1*l%spot
    1*cgeom
    1*merf
    1*end

******************************
* Response to raysin command: *
******************************
```

```
    4 ray(s) read in from file  13

*******************************
* Response to the command sq: *
*******************************

 In subroutine sq
accept  1:    z(  1,1)
accept  2:    z(  2,1)
accept  3:    z(  3,1)
accept  4:    z(  4,1)

 Aims/quantities selected :
No.     item      present value
----------------------------------
 1  z(  1,1) =     -0.200000000
 2  z(  2,1) =      0.200000000
 3  z(  3,1) =     -0.100000000
 4  z(  4,1) =      0.100000000

****************************
* Response to merf command: *
****************************

 number of lines read =         100

end of MARYLIE run
```

10.12.3 Beam Moment Plots with Geometry

Figure 10.12.3.1 shows a position spread plot, with additional geometric data, for the spot-forming system of sections 2.2 and 10.11. Exhibit 10.12.3f shows the MaryLie run used to produce this plot. Much of this Master Input File is similar to that of Exhibit 10.11c. However, there are a few differences that are described below.

First, because path-length information will now be provided by *geom*, file 20 which provides instruction for *sq* is shortened to exclude *pl* as shown in Exhibit 10.12.3a. Correspondingly, as shown in Exhibit 10.12.3b, file 30 now contains only spread and moment quantities.

Second, the line % now takes the form

```
%
     1*work1     1*dp
```

and the line *work1* itself takes the simpler form

```
work1
     1*samap     1*wsq
```

since there is now no need to invoke a *pli0* command because path-length information will be computed by *geom*.

Third, the *#labor* component of the Master Input File now has the contents

```
#labor
 1*fileout
 1*setup1
 1*%spot
 1*l%spot
 1*cgeom
 1*merf
 1*end
```

The first 3 entries (*1*fileout*, *1*setup1*, *1*%spot*) function as before except that (as already mentioned) only spead and moment, and not path-length, information is written on file 30. See Exhibit 10.12.3b. The entry *1*l%spot* invokes a loop whose contents is *%spot*. See the *#loops* component of Exhibit 10.12.3f. The entry *cgeom* invokes a line that contains a *geom* command preceded by the commands *ps1* and *ps2* that set the parameters required by geom:

```
cgeom
     1*ps1   1*ps2  1*geom
```

Figure 10.12.3.1: The rms position spreads $\sqrt{\langle X^2 \rangle}$ and $\sqrt{\langle Y^2 \rangle}$ for the spot-forming system treated in section 10.11 accompanied by a schematic showing the lengths and locations of quadrupoles and their apertures.

So doing causes various geometric properties of the spot-forming system to be computed and written to the external file IFILE, in this case file 40, at each occurrence of the command *dp* (which in turn occurs in the line %). See section 8.38. Exhibit 10.12.3c shows the first few lines of file 40. In particular, the first column of file 40 is path length along the design orbit. Finally the *merf* command in *#labor* merges the first column of file 40 with the remaining columns of file 30 and writes the result on file 50. (See the contents of *#labor* and section 8.40.) Exhibit 10.12.3d shows the first few lines of file 50. Note that the contents of Exhibits 10.12.3d and 10.11b are identical except for the interchange of various columns. Therefore, if desired, this file could be used to produce (with the aid of any of a variety of graphics programs) simple spread and moment plots of the form shown in section 10.11.

In addition to file 40 (which contains path-length and coordinate data), *geom* also writes simple element description and path-length information to the file KFILE, in this case file 60. See section 8.38. Exhibit 10.12.3e shows the first few lines of file 60. By processing with POSTER simultaneously file 50 (written by *merf*) and file 60, it is possible to produce spread and moment plots accompanied by geometrical information about the associated beam line. Figure 10.12.3 shows a composite spread plot produced in this way for the spot-forming system of sections 2.2 and 10.11. This composite plot displays the lengths, locations, and apertures of the quadrupoles in the beam line as well as spread data.

There is one last detail to be described. Observe, by comparing Exhibits 10.11c and 10.12.3f, that the lines *hfq* and *hdq* in this case contain a *dims* command with user name *aperture*. This command is recognized by *geom*, affects the contents of the file 60 it writes, and is used to specify the aperture size for the quadrupoles that appear in figure 10.12.3. See section 7.40.

```
Exhibit 10.12.3a
Instructions in file 20 for the sq command:

 1  bm2(1,1)
 2  bm2(2,2)
 3  bm2(3,3)
 4  bm2(4,4)
 5  bm1(1,2)
 #


Exhibit 10.12.3b
First few lines of file 30 that contains data written in response to the
wsq command.  In accord with the ordering specified in Exhibit 10.12a
above, the columns are the spread and moment quantities sqrt<XX>,
sqrt<PxPx>, sqrt<YY>, sqrt<PyPy>, and <XPx>, respectively.

 2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  0.00000E+00
 2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  3.12503E-07
 2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  6.25006E-07
 2.00000E-01  2.50000E-03  2.00000E-01  2.50000E-03  9.37509E-07
 2.00001E-01  2.50000E-03  2.00001E-01  2.50000E-03  1.25001E-06


Exhibit 10.12.3c
First few lines of file 40 that contains data written in response to
```

```
the geom command.  Column 1 is path length data along the design orbit:

 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 5.00000E-02  0.00000E+00  0.00000E+00  5.00000E-02  5.31071E-10  0.00000E+00
 1.00000E-01  0.00000E+00  0.00000E+00  1.00000E-01  1.06214E-09  0.00000E+00
 1.50000E-01  0.00000E+00  0.00000E+00  1.50000E-01  1.59321E-09  0.00000E+00
 2.00000E-01  0.00000E+00  0.00000E+00  2.00000E-01  2.12429E-09  0.00000E+00


Exhibit 10.12.3d
First few lines of file 40 that contains data written in response to
the merf command.  Column 1 is path length data along the design orbit,
and the next 4 columns are spread and moment quantities:

  0.0000E+00  0.2000E+00  0.2500E-02  0.2000E+00  0.2500E-02  0.0000E+00
  0.5000E-01  0.2000E+00  0.2500E-02  0.2000E+00  0.2500E-02  0.3125E-06
  0.1000E+00  0.2000E+00  0.2500E-02  0.2000E+00  0.2500E-02  0.6250E-06
  0.1500E+00  0.2000E+00  0.2500E-02  0.2000E+00  0.2500E-02  0.9375E-06
  0.2000E+00  0.2000E+00  0.2500E-02  0.2000E+00  0.2500E-02  0.1250E-05


Exhibit 10.12.3e
First few lines of file 60 that contains data written in response to
the geom command.  It contains simple element description and
path-length information:

drs/10    1  1  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  5.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  1.50000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  2.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00
drs/10    1  1  2.00000E-01  0.00000E+00  0.00000E+00  0.00000E+00


Exhibit 10.12.3f
***MARYLIE 3.0***
Prerelease Development Version 6/17/99
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.12.3
      This is a MARYLIE run that demonstrates the production of beam moment
 and geometric data for the simple spot forming system of Section 2.2.
 To do this all quads and drifts are split in 10 pieces, except for the
 final long drift  which is split in 20 pieces.  The system is also
 slightly modified by preceding it with 2 short drifts.

 Transformed moments are computed from the initial moments using the
```

```
 accumulated transfer map.  See the contents of the line work1.

 Alternatively, transformed moments could be computed from the
 moments just preceding using just the map for the current element
 slice.  See the contents of the line work2.  This approach is the
 "moment" analog of element-by-element (actually slice-by-slice)
 tracking.

 The beam parameters are those for 50 MeV protons.

#beam
  1.03527440851950
 5.328901960570000E-002
  1.00000000000000
  1.00000000000000
#menu
 cdrs     drft
  0.500000000000000
 cdrl     drft
   20.0026000000000
 chfq     quad
   1.50000000000000      8.630000000000000E-02   1.00000000000000
   1.00000000000000
 chdq     quad
   3.00000000000000     -8.289450000000000E-02   1.00000000000000
   1.00000000000000
 drs/10   drft
  5.000000000000000E-02
 drl/20   drft
   1.00013000000000
 inhfq    quad
  0.000000000000000E+00  8.630000000000000E-02   1.00000000000000
  0.000000000000000E+00
 outhfq   quad
  0.000000000000000E+00  8.630000000000000E-02  0.000000000000000E+00
   1.00000000000000
 inhdq    quad
  0.000000000000000E+00 -8.289450000000000E-02   1.00000000000000
  0.000000000000000E+00
 outhdq   quad
  0.000000000000000E+00 -8.289450000000000E-02  0.000000000000000E+00
   1.00000000000000
 hfq/10   quad
  0.150000000000000      8.630000000000000E-02  0.000000000000000E+00
  0.000000000000000E+00
 hdq/10   quad
  0.300000000000000     -8.289450000000000E-02  0.000000000000000E+00
  0.000000000000000E+00
 aperture dims
  0.300000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
```

```
mapout   ptm
  3.00000000000000       3.00000000000000      0.000000000000000E+00
 0.000000000000000E+00   1.00000000000000
inv      inv
wcl10    wcl
  3.00000000000000       10.0000000000000       3.00000000000000
mark     mark
iden     iden
raysin   rt
  13.0000000000000       14.0000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
rt       rt
 0.000000000000000E+00   14.0000000000000       5.00000000000000
  1.00000000000000       1.00000000000000      0.000000000000000E+00
sq       sq
  20.0000000000000      0.000000000000000E+00   1.00000000000000
  1.00000000000000
wsq      wsq
  1.00000000000000       1.00000000000000       30.0000000000000
  1.00000000000000       2.00000000000000      0.000000000000000E+00
dp       dp
geom     geom
  3.00000000000000      0.000000000000000E+00  0.000000000000000E+00
  1.00000000000000       2.00000000000000      0.000000000000000E+00
ps1      ps1
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
ps2      ps2
  40.0000000000000      0.000000000000000E+00   60.0000000000000
 0.000000000000000E+00  0.000000000000000E+00   1.00000000000000
merf     usr12
  40.0000000000000       30.0000000000000       50.0000000000000
  1.00000000000000       5.00000000000000      0.000000000000000E+00
bgen     bgen
  2.00000000000000       2.00000000000000      0.000000000000000E+00
  123.000000000000       1.00000000000000       3.00000000000000
ps3      ps3
 5.000000000000000E-04  5.000000000000000E-04  0.000000000000000E+00
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
gbuf1    gbuf
  2.00000000000000       1.00000000000000
gbuf2    gbuf
  2.00000000000000       2.00000000000000
twx      twsm
  1.00000000000000       86.0000000000000      0.000000000000000E+00
  80.0000000000000
twy      twsm
  2.00000000000000       96.0000000000000      0.000000000000000E+00
  80.0000000000000
amap     amap
  2.00000000000000       1.00000000000000      -1.00000000000000
 0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
samap    amap
```

```
   2.00000000000000      0.000000000000000E+00  -1.00000000000000
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 stm1     stm
   1.00000000000000
 snor     snor
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00
 end      end
#lines
 hfq
     1*aperture    1*inhfq      10*hfq/10      1*outhfq
 hdq
     1*aperture    1*inhdq      10*hdq/10      1*outhdq
 drs
    10*drs/10
 drl
    20*drl/20
 trip
     1*hfq         1*drs         1*hdq         1*drs         1*hfq
 spot
     2*drs         1*trip        1*drl
 ctrip
     1*chfq        1*cdrs        1*chdq        1*cdrs        1*chfq
 cspot
     2*cdrs        1*ctrip       1*cdrl
 %spot
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*aperture    1*%           1*inhfq       1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*outhfq      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*aperture    1*%           1*inhdq    &
     1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
     1*hdq/10      1*%           1*hdq/10      1*%           1*hdq/10   &
     1*%           1*hdq/10      1*%           1*hdq/10      1*%        &
     1*hdq/10      1*%           1*hdq/10      1*%           1*hdq/10   &
     1*%           1*outhdq      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*drs/10      1*%        &
     1*drs/10      1*%           1*drs/10      1*%           1*drs/10   &
     1*%           1*drs/10      1*%           1*aperture    1*%        &
```

```
     1*inhfq       1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*hfq/10      1*%           1*hfq/10   &
     1*%           1*hfq/10      1*%           1*hfq/10      1*%        &
     1*hfq/10      1*%           1*outhfq      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%           1*drl/20   &
     1*%           1*drl/20      1*%           1*drl/20      1*%        &
     1*drl/20      1*%           1*drl/20      1*%
 %
     1*work1       1*dp
 work1
     1*samap       1*wsq
 work2
     1*samap       1*wsq         1*gbuf1       1*stm1        1*iden
 setup
     1*momgen      1*sq
 momgen
     1*ps3         1*bgen        1*gbuf2       1*mapout      1*stm1     &
     1*iden        1*tws         1*mapout      1*snor        1*gbuf1    &
     1*mapout      1*amap        1*gbuf1       1*mapout      1*stm1     &
     1*iden
 tws
     1*twx         1*twy
 cgeom
     1*ps1         1*ps2         1*geom
#lumps
#loops
 lspot
     1*spot
 l%spot
     1*%spot
#labor
    1*fileout
    1*setup
    1*%spot
    1*l%spot
    1*cgeom
    1*merf
    1*end

*********************************
* Response to the command bgen: *
*********************************

 analytically computed values of selected moments
 values of <x*x>, <x*px>, <px*px>:
 5.000000000000000E-004  0.000000000000000E+000  5.000000000000000E-004
 values of <y*y>, <y*py>, <py*py>:
```

```
 5.000000000000000E-004  0.000000000000000E+000  5.000000000000000E-004
 values of <t*t>, <t*pt>, <pt*pt>:
 0.000000000000000E+000  0.000000000000000E+000  0.000000000000000E+000

*********************************************
* Moments written in "map" form in response *
* to the gbuf2 and mapout commands:         *
*********************************************

matrix for map is :

 5.00000E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  5.00000E-04  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  5.00000E-04  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  5.00000E-04  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 5.00000000000000E-04
 f( 13)=f( 02 00 00 )= 5.00000000000000E-04
 f( 18)=f( 00 20 00 )= 5.00000000000000E-04
 f( 22)=f( 00 02 00 )= 5.00000000000000E-04
 f( 84)=f( 40 00 00 )= 5.62500000000000E-07
 f( 90)=f( 22 00 00 )= 1.87500000000000E-07
 f( 95)=f( 20 20 00 )= 1.87500000000000E-07
 f( 99)=f( 20 02 00 )= 1.87500000000000E-07
 f(140)=f( 04 00 00 )= 5.62500000000000E-07
 f(145)=f( 02 20 00 )= 1.87500000000000E-07
 f(149)=f( 02 02 00 )= 1.87500000000000E-07
 f(175)=f( 00 40 00 )= 5.62500000000000E-07
 f(179)=f( 00 22 00 )= 1.87500000000000E-07
 f(195)=f( 00 04 00 )= 5.62500000000000E-07

*********************************
* Response to the stm1 command: *
*********************************

map stored in location   1

************************************
* Map produced by the tws command: *
************************************

matrix for map is :

 6.97565E-02  7.98051E+01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
-1.24696E-02  6.97565E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.04528E-01  7.95618E+01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00 -1.24315E-02 -1.04528E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00
```

```
nonzero elements in generating polynomial are :

**********************************
* Response to the snor command: *
**********************************

det in fxpt is     0.4110E+01

*****************************************
* The matching map script A written in   *
* response to gbuf1 and mapout commands: *
*****************************************

matrix for map is :

 8.94427E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  1.11803E-01  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  8.94427E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  1.11803E-01  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.00000E+00

nonzero elements in generating polynomial are :

**********************************
* Response to the amap command: *
**********************************

map gotten from location   1

*****************************************
* Moments written in "map" form by amap: *
*****************************************

 transformed moments

matrix for map is :

 4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 4.00000000000000E-02
 f( 13)=f( 02 00 00 )= 6.25000000000000E-06
 f( 18)=f( 00 20 00 )= 4.00000000000000E-02
 f( 22)=f( 00 02 00 )= 6.25000000000000E-06
 f( 84)=f( 40 00 00 )= 3.60000000000000E-03
```

```
 f( 90)=f( 22 00 00 )= 1.87500000000000E-07
 f( 95)=f( 20 20 00 )= 1.20000000000000E-03
 f( 99)=f( 20 02 00 )= 1.87500000000000E-07
 f(140)=f( 04 00 00 )= 8.78906250000000E-11
 f(145)=f( 02 20 00 )= 1.87500000000000E-07
 f(149)=f( 02 02 00 )= 2.92968750000000E-11
 f(175)=f( 00 40 00 )= 3.60000000000000E-03
 f(179)=f( 00 22 00 )= 1.87500000000000E-07
 f(195)=f( 00 04 00 )= 8.78906250000000E-11

**************************************************
* Response to the gbuf1 and mapout commands       *
* showing the contents of buffer 1 in "map" form: *
**************************************************

matrix for map is :

 4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  4.00000E-02  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  6.25000E-06  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00
 0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00

nonzero elements in generating polynomial are :

 f(  7)=f( 20 00 00 )= 4.00000000000000E-02
 f( 13)=f( 02 00 00 )= 6.25000000000000E-06
 f( 18)=f( 00 20 00 )= 4.00000000000000E-02
 f( 22)=f( 00 02 00 )= 6.25000000000000E-06
 f( 84)=f( 40 00 00 )= 3.60000000000000E-03
 f( 90)=f( 22 00 00 )= 1.87500000000000E-07
 f( 95)=f( 20 20 00 )= 1.20000000000000E-03
 f( 99)=f( 20 02 00 )= 1.87500000000000E-07
 f(140)=f( 04 00 00 )= 8.78906250000000E-11
 f(145)=f( 02 20 00 )= 1.87500000000000E-07
 f(149)=f( 02 02 00 )= 2.92968750000000E-11
 f(175)=f( 00 40 00 )= 3.60000000000000E-03
 f(179)=f( 00 22 00 )= 1.87500000000000E-07
 f(195)=f( 00 04 00 )= 8.78906250000000E-11

*********************************
* Response to the stm1 command: *
*********************************

map stored in location   1

******************************
* Response to the sq command: *
******************************


 In subroutine sq
```

```
accept  1:    bm2(1,1)
accept  2:    bm2(2,2)
accept  3:    bm2(3,3)
accept  4:    bm2(4,4)
accept  5:    bm1(1,2)

 Aims/quantities selected :
No.     item      present value
-----------------------------------
 1  bm2(1,1) =      0.200000000
 2  bm2(2,2) =      2.500000000E-03
 3  bm2(3,3) =      0.200000000
 4  bm2(4,4) =      2.500000000E-03
 5  bm1(1,2) =      0.000000000E+00

**********************************
* Response to the merf command: *
**********************************

 number of lines read =         100

end of MARYLIE run
```

10.13 Production of Layout Drawings

The *geom* command can also be used to write simple element description and path-length and coordinate information for a lattice on file LFILE, and this information can be processed by POSTER to produce floor-plan layout drawings. See sections 1.4.2 and 8.38. To produce LFILE, it is only necessary to define a *loop* whose contents is the lattice to be drawn, invoke this loop in *#labor*, and follow this invocation by a *geom* command. There is no need to subdivide the lattice.

Figure 13 shows a floor-plan layout drawing of the Proton Storage Ring produced in this way; and Exhibit 10.13 shows the Master Input File for the associated MaryLie run.

Figure 10.13.1: Floor plan of the Proton Storage Ring of section 2.5. Compare with figure 2.5.1. The y axis is out of the plane of the paper.

```
Exhibit 10.13
***MARYLIE 3.0***
Prerelease Development Version 8/21/98
Copyright 1987 Alex J. Dragt
All rights reserved

Data input complete; going into #labor.
#comment
 Exhibit 10.13.
      This is a MaryLie run that employs the geom type code
 to produce a layout drawing using the PSR as an example.
#beam
  4.86914813175970
 0.849425847892200
  1.00000000000000
  1.00000000000000
#menu
 drvs     drft
  0.300000000000000
 drs      drft
  0.450000000000000
 drml     drft
   1.48646000000000
 drl      drft
   2.28646000000000
 bend     pbnd
   36.0000000000000      0.000000000000000E+00  0.500000000000000
   1.20000000000000
 hfq      quad
  0.500000000000000       2.72000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hdq      quad
  0.500000000000000      -1.92000000000000      0.000000000000000E+00
  0.000000000000000E+00
 hcs      sext
  0.500000000000000      0.000000000000000E+00
 vcs      sext
  0.500000000000000      0.000000000000000E+00
 fileout  pmif
   1.00000000000000       12.0000000000000       3.00000000000000
 mapout   ptm
   3.00000000000000       3.00000000000000      0.000000000000000E+00
  0.000000000000000E+00   1.00000000000000
 chrom    tasm
   2.00000000000000      1.000000000000000E-03   1.00000000000000
  0.000000000000000E+00   3.00000000000000      0.000000000000000E+00
 geom     geom
   2.00000000000000      0.000000000000000E+00  0.000000000000000E+00
   1.00000000000000       2.00000000000000      0.000000000000000E+00
 ps1      ps1
   14.0000000000000      0.000000000000000E+00  0.000000000000000E+00
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
 ps2      ps2
```

```
  0.000000000000000E+00  0.000000000000000E+00  0.000000000000000E+00
   80.0000000000000      0.000000000000000E+00   5.00000000000000
 fin      end
#lines
 nsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drl
 tsex
     1*drl         1*hdq         1*drs         1*bend        1*drs      &
     1*hfq         1*drvs        1*hcs         1*drml
 lsex
     1*drml        1*vcs         1*drvs        1*hdq         1*drs      &
     1*bend        1*drs         1*hfq         1*drl
 half
     1*nsex        1*tsex        1*lsex        1*nsex        1*nsex
 ring
     2*half
 cgeom
     1*ps1         1*ps2         1*geom
#lumps
#loops
 lring
     1*ring
#labor
    1*fileout
    1*lring
    1*cgeom
    1*fin

end of MARYLIE run
```

Chapter 11

References

11.1 Fundamental Constants

CODATA Internationally recommended values for the Fundamental Physical Constants, http://physics.nist.gov/cuu/Constants/. See also P. Mohr and B. Taylor, "The Fundamental Physical Constants", *Physics Today Buyers' Guide 2003* (August 2003). http://www.physicstoday.org/guide/fundcon.html

P.G. Harris, *Observation of High-Lying Resonances in the H^- Ion*, Los Alamos Technical Report LA-11843-T (1990).

11.2 Other Charged Particle Beam Transport Codes

K.L. Brown, *TRANSPORT*, SLAC-75, revision 3 (1975); K.L. Brown, F. Rothacker, D.C. Carey, and Ch. Iselin, *TRANSPORT*, SLAC-91, revision 2 (1977).

F.C. Iselin, E. Forest, F. Schmidt, H. Grote, *MAD*, http://wwwslap.cern.ch/mad/.

M. Berz, *COSY Infinity*, http://cosy.pa.msu.edu/.

11.3 Accelerator Handbooks

A.W. Chao and M. Tigner, eds, *Handbook of Accelerator Physics and Engineering*, World Scientific (2000).

11.4 Conventional Accelerator Theory

D.A. Edwards and M.J. Syphers, *An Introduction to the Physics of High Energy Accelerators*, John Wiley (1993).

H. Wiedemann, *Particle Accelerator Physics I, Basic Principles and Linear Beam Dynamics*, Springer (1993).

H. Wiedemann, *Particle Accelerator Physics II, Nonlinear and Higher-Order Beam Dynamics*, Springer (1995).

M. Reiser, *Theory and Design of Charged Particle Beams*, John Wiley (1994).

S.Y. Lee, *Accelerator Physics*, World Scientific (1999).

E.D. Courant and H. Snyder, "Theory of the alternating-gradient synchrotron", Annals Phys. **3**, p. 1 (1958).

D.C. Carey, *The Optics of Charged Particle Beams (Accelerators and Storage Rings*, Vol. 6), Harwood Academic Publishers (1987).

P.J. Bryant and K. Johnsen, *The Principles of Circular Accelerators and Storage Rings*, Cambridge University Press (1993).

11.5 General Lie Algebraic References

A.J. Dragt and J.M. Finn, "Lie series and invariant functions for analytic symplectic maps," J. Math Phys. **17**, 2215 (1976).

A.J. Dragt and E. Forest, "Computation of nonlinear behavior of Hamiltonian systems using Lie algebraic methods", J. Math. Physics **24**, p. 2734 (1983).

11.6 Lie Algebraic/Map Based Accelerator Theory

A.J. Dragt "A Method of transfer maps for linear and nonlinear beam elements," IEEE Trans. Nuc. Sci. **NS-26**, p. 3601 (1979).

A.J. Dragt, "Lectures on nonlinear orbit dynamics," *Physics of High Energy Particle Accelerators*, AIP Conference Proceedings No. 87, R.A. Carrigan et al., editors (1982).

A.J. Dragt et al., "Lie Algebraic Treatment of Linear and Nonlinear Beam Dynamics," Ann. Rev. Nucl. Part. Sci. **38**, p. 455 (1988).

A.J. Dragt et al.,"General moment invariants for linear Hamiltonian systems", Phys. Rev. A **45**, p. 2572 (1992).

A.J. Dragt, "Summary of the Working Group on Maps", Particle Accelerators **55**, p. 499 (1996).

A.J. Dragt, "Lie Methods for Nonlinear Dynamics with Applications to Accelerator Physics", 850 pages, University of Maryland Physics Department Technical Report (2003).

E. Forest, *Beam Dynamics, A New Attitude and Framework*, Harwood Academic Publishers (1998).

L. Michelotti, *Intermediate Classical Dynamics with Applications to Beam Physics*, John Wiley (1995).

11.7 Applications of Lie Algebraic/Map Based Methods to Specific Problems in Accelerator Physics

A.J. Dragt, "Exact Numerical Calculation of Chromaticity in Small Rings", Particle Accelerators **12**, p. 205 (1982).

J. Irwin, "The application of Lie Algebra techniques to beam transport design", Nucl. Instrum. Meth. **A298**, p. 460 (1990).

J. Irwin, Using Lie Algebraic Maps for the Design and Operation of Colliders, Particle Accelerators **54**, 107 (1996).

Y. Yan, J. Irwin, and T. Chan, Resonance Basis Maps and nPB Tracking for Dynamic Aperture Studies, Particle Accelerators **55**, 17 (1996).

Y. Yan, J. Irwin, Y. Cai, M. Donald, and Y. Nosochkov, "Dynamic Aperture Improvement of PEP-II Lattices Using Resonance Basis Lie Generators", SLAC Publication (1996).

11.8 Rare Earth Cobalt Quadrupoles

K. Halbach, Nuc. Inst. Meth. **187**, p. 109 (1981).

11.9 Combined Function Bend

A.J. Dragt, et al., "Numerical Third-Order Transfer Map for Combined Function Dipole", University of Maryland Physics Department Technical Report (1989).

11.10 Solenoid

A.J. Dragt, "Numerical third-order transfer map for solenoid", Nucl. Instrum. Meth. **A298**, p. 441 (1990).

11.11 MaryLie (Giorgilli) Indexing Scheme

A. Giorgilli, Comp. Phys. Comm. **16**, p. 331 (1979).

11.12 Random Number Generation

W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical Recipes in FORTRAN: The Art of Scientific Computing* (Cambridge U.P., New York, 1992), 2nd ed. See also W.H. Press and S.A. Teukolsky, "Portable Random Number Generators", Computers in Physics **6**, p. 522 (1992).

Chapter 12

Comments and Suggestion Form

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Name _____ E-mail _____ Date _____
Organization _____
Street _____
City _____ State _____ Zip Code _____
or Country

Mail to: Alex J. Dragt (dragt@physics.umd.edu)
Physics Department
University of Maryland
College Park, Maryland 20742
http://www.physics.umd.edu/dsat

Chapter 13

Quick Reference Guide to Elements, Commands, Files, and Messages

13.1 Alphabetical Listing of All Type-Code Mnemonics

All type-code mnemonics, as currently available in MaryLie 3.0, are listed below alphabetically along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Purpose	Subsection
aim	Specify quantities to be fit or optimized and set target values.	9.5
amap	Apply map to a function or moments.	8.17
arc	Arc for accounting purposes.	6.31
arot	Axial rotation.	6.14
asni	Apply script N inverse.	8.29
bell	Ring bell at terminal.	7.31
bgen	Generate beam.	8.34
bip	Begin inner procedure.	9.1
bop	Begin outer procedure.	9.2
cbm	Change or write out beam parameters.	7.39
cdf	Change drop file.	7.30
cf	Close files.	7.23
cfbd	Combined function bend.	6.8
cfqd	Combined function magnetic quadrupole.	6.24
cfrn	Change or write out values of fringe field parameters for combined function dipole.	6.29
circ	Set parameters and circulate.	7.3

Type Code	Purpose	Subsection
cod	Compute off-momentum closed orbit data.	8.1
con1 ⋮ con5	Constraints.	9.12
cplm	"Compressed" approximation to low order multipoles.	6.17
cps1 ⋮ cps9	Capture parameter set.	9.15
csym	Check symplectic condition.	8.31
ctr	Change tune range.	8.28
cxp	Chromatic expansion.	8.24
dia	Dynamic invariant analysis.	8.11
dims	Dimensions.	7.40
dism	Dispersion matrix.	6.22
dnor	Dynamic normal form analysis.	8.9
dp	Data point.	6.26
dpol	Dispersion polynomial.	7.38
drft	Drift space.	6.1
eapt	Aperture the beam with an elliptic aperture.	7.19
end	Halt execution. Must be an entry of a #labor list.	7.1
exp	Compute exponential.	8.7
fadm	Fourier analyze dynamic map.	8.16
fasm	Fourier analyze static map.	8.15
fit	Carry out fitting operation.	9.7
flag	Change or write out values of flags and defaults.	9.17
fps	Free parameter set.	9.16
frng	Used for hard-edge dipole fringe fields.	6.7
ftm	Filter the existing transfer map.	7.22
fwa	Copy file to working array.	8.39

Type Code	Purpose	Subsection
gbdy	Used for the body of a general bending magnet.	6.6
gbnd	General bending magnet.	6.4
gbuf	Get buffer contents.	8.14
geom	Compute geometry of a loop.	8.38
grad	Compute gradient matrix.	9.13
gtm	Get a transfer map from storage.	7.17
iden	Replace existing transfer map by the identity map.	7.8
inf	Change or write out values of infinities.	7.35
inv	Replace existing transfer map by its inverse.	7.9
jmap	Map with matrix part J.	6.18
lnf	Compute logarithm of normal form.	8.41
mark	Marker.	6.25
mask	Mask off selected portions of existing transfer map.	7.13
merf	Merge files.	8.40
mn	Compute matrix norm.	8.33
moma	Moment analysis.	8.37
mrt0	Merit function (sum of squares).	9.10
mrt1 ⋮ mrt5	Merit functions (user written).	9.11
mss	Minimize sum of squares optimization.	9.8
nbnd	Normal entry bending magnet, with or without fringe fields.	6.2
num	Number lines in a file.	7.27
octe	Electric octupole.	6.12
octm	Magnetic octupole.	6.11
of	Open files.	7.24
opt	General optimization.	9.9
padd	Add two polynomials.	8.19

Type Code	Purpose	Subsection
paws	Pause.	7.34
pb	Poisson bracket two polynomials.	8.21
pbnd	Parallel faced bending magnet, with fringe fields and equal entry and exit angles.	6.3
pdnf	Compute power of dynamic normal form.	8.13
pli	Path length information.	7.43
pmif	Print contents of Master Input File (file 11).	7.4
pmul	Multiply two polynomials.	8.20
pnlp	Compute power of nonlinear part.	8.30
pold	Polar decomposition of a map.	8.22
ppa	Principal planes analysis.	8.36
prot	Used for leading and trailing pole face rotations.	6.5
ps1 ⋮ ps9	Parameter set specification.	7.25
psnf	Compute power of static normal form.	8.12
psp	Polynomial scalar product.	8.32
ptm	Print transfer map.	7.7
pval	Evaluate a polynomial.	8.23
quad	Magnetic quadrupole.	6.9
radm	Resonance analyze dynamic map.	8.5
rapt	Aperture the beam with a rectangular aperture.	7.18
rasm	Resonance analyze static map.	8.4
rbnd	Rectangular bending magnet.	6.3
recm	REC quadrupole multiplet.	6.27
rev	Replace existing transfer map by its reversed map.	7.11
revf	Replace existing transfer map by reverse factorized form.	7.12
rmap	Random map.	6.30
rps1 ⋮ rps9	Random parameter set specification.	7.26

Type Code	Purpose	Subsection
rset	Reset menu entries.	9.18
rt	Perform a ray trace.	7.2
r****	Random counterpart of the element with type-code mnemonic ****.	6.19
sbnd	Sector bending magnet.	6.2
scan	Scan parameter space.	9.14
sext	Magnetic sextupole.	6.10
shoa	Show contents of arrays.	7.44
sia	Static invariant analysis.	8.10
smul	Multiply a polynomial by a scalar.	8.18
snor	Static normal form analysis.	8.8
sol	Solenoid.	6.23
spce	Space for accounting purposes.	6.28
sq	Select quantities.	8.26
sqr	Square the existing transfer map.	7.15
srfc	Short RF cavity.	6.13
stm	Store the existing transfer map.	7.16
symp	Symplectify matrix portion of transfer map.	7.14
tadm	Twiss analyze dynamic map.	8.3
tasm	Twiss analyze static map.	8.2
tbas	Translate basis.	8.6
thlm	"Thin lens" approximation to low order multipoles.	6.16
tic	Translate initial conditions.	8.35
time	Write out execution time.	7.29
tip	Terminate inner procedure.	9.3
tmi	Input matrix elements and polynomial coefficients from an external file.	7.5
tmo	Ouptut matrix elements and polynomial coefficients to an external file.	7.6
top	Terminate outer procedure.	9.4
tpol	Twiss polynomial.	7.37
tqm	Transfer quadratic moments.	8.25
tran	Replace existing transfer map by its "transpose".	7.10

Type Code	Element	Subsection
twsm	Linear matrix transformation specified in terms of twiss parameters.	6.15
usr1 ⋮ usr10	User specified subroutines that act on phase space data.	6.20
usr11 ⋮ usr20	User specified subroutines that produce or act on maps.	6.21
vary	Specify quantities to be varied.	9.6
wcl	Write contents of a loop.	7.33
whst	Write history of beam loss.	7.21
wmrt	Write out value of merit function.	7.32
wnd	Window a beam.	7.20
wnda	Window a beam in all planes.	7.42
wps	Write out parameters in a parameter set.	7.28
wsq	Write selected quantities.	8.27
wuca	Write out contents of ucalc array.	7.41
zer	Change or write out values of zeroes.	7.36

13.2 Element Type-Code Mnemonics (Alphabetical Order)

The beam-line elements and their type code mnemonics, as currently available in MaryLie 3.0, are listed below alphabetically along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Element	Subsection
arc	Arc for accounting purposes.	6.31
arot	Axial rotation.	6.14
cfbd	Combined function bend.	6.8
cfqd	Combined function magnetic quadrupole.	6.24
cfrn	Change or write out values of fringe field parameters for combined function dipole.	6.29
cplm	"Compressed" approximation to low order multipoles.	6.17
dism	Dispersion matrix.	6.22
dp	Data point.	6.26
drft	Drift space.	6.1
frng	Used for hard-edge dipole fringe fields.	6.7
gbdy	Used for the body of a general bending magnet.	6.6
gbnd	General bending magnet.	6.4
jmap	Map with matrix part J.	6.18
mark	Marker.	6.25
nbnd	Normal entry bending magnet, with or without fringe fields.	6.2
octe	Electric octupole.	6.12
octm	Magnetic octupole.	6.11
pbnd	Parallel faced bending magnet, with fringe fields and equal entry and exit angles.	6.3
prot	Used for leading and trailing pole face rotations.	6.5
quad	Magnetic quadrupole.	6.9

Type Code	Element	Subsection
rbnd	Rectangular bending magnet.	6.3
recm	REC quadrupole multiplet.	6.27
rmap	Random map.	6.30
r****	Random counterpart of the element with type-code mnemonic ****.	6.19
sbnd	Sector bending magnet.	6.2
sext	Magnetic sextupole.	6.10
sol	Solenoid.	6.23
spce	Space for accounting purposes.	6.28
srfc	Short RF cavity.	6.13
thlm	"Thin lens" approximation to low order multipoles.	6.16
twsm	Linear matrix transformation specified in terms of twiss parameters.	6.15
usr1 ⋮ usr10	User specified subroutines that act on phase space data.	6.20
usr11 ⋮ usr20	User specified subroutines that produce or act on maps.	6.21

13.3 Element Type-Code Mnemonics (Functional Order)

The beam-line elements and their type code mnemonics, as currently available in MaryLie 3.0, are listed below according to function along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Element	Subsection
drft	Drift space	6.1
	dipole bend magnets	
nbnd sbnd	a) Normal entry (sector) bending magnet, with or without fringe fields.	6.2
pbnd rbnd	b) Parallel faced (rectangular) bending magnet, with fringe fields and equal entry and exit angles.	6.3
gbnd	c) General bending magnet.	6.4
prot	d) Used for leading and trailing pole face rotations.	6.5
gbdy	e) Used for the body of a general bending magnet.	6.6
frng	f) Used for hard-edge dipole fringe fields.	6.7
cfbd	g) Combined function bend.	6.8
cfrn	h) Change or write out values of fringe field parameters for combined function dipole.	6.29
sol	Solenoid.	6.23
quad	Magnetic quadrupole.	6.9
cfqd	Combined function magnetic quadrupole.	6.24
recm	REC quadrupole multiplet.	6.27
sext	Magnetic sextupole.	6.10
octm	Magnetic octupole.	6.11
octe	Electric octupole.	6.12
srfc	Short RF cavity.	6.13
arot	Axial rotation.	6.14
thlm	"Thin lens" approximation to low order multipoles.	6.16

Type Code	Element	Subsection
cplm	"Compressed" approximation to low order multipoles.	6.17
twsm	Linear matrix transformation specified in terms of twiss parameters.	6.15
dism	Dispersion matrix.	6.22
jmap	Map with matrix part J.	6.18
rmap	Random map.	6.30
mark	Marker.	6.25
dp	Data point.	6.26
spce	Space for accounting purposes.	6.28
arc	Arc for accounting purposes.	6.31
usr1 ⋮ usr10	User specified subroutines that act on phase space data .	6.20
usr11 ⋮ usr20	User specified subroutines that produce or act on maps.	6.21
r****	Random counterpart of the element with type-code mnemonic ****.	6.19

13.4 Simple Command Type-Code Mnemonics (Alphabetical Order)

Simple commands and their type code mnemonics, as currently available in MaryLie 3.0, are listed below alphabetically along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Command	Subsection
bell	Ring bell at terminal.	7.31
cbm	Change or write out beam parameters.	7.39
cdf	Change drop file.	7.30
cf	Close files.	7.23
circ	Set parameters and circulate.	7.3
dims	Dimensions.	7.40
dpol	Dispersion polynomial.	7.38
eapt	Aperture the beam with an elliptic aperture.	7.19
end	Halt execution. Must be an entry of a #labor list.	7.1
ftm	Filter the existing transfer map.	7.22
gtm	Get a transfer map from storage.	7.17
iden	Replace existing transfer map by the identity map.	7.8
inf	Change or write out values of infinities.	7.35
inv	Replace existing transfer map by its inverse.	7.9
mask	Mask off selected portions of existing transfer map.	7.13
num	Number lines in a file.	7.27
of	Open files.	7.24
paws	Pause.	7.34
pli	Path length information.	7.43
pmif	Print contents of Master Input File (file 11).	7.4
ps1 ⋮ ps9	Parameter set specification.	7.25
ptm	Print transfer map.	7.7

Type Code	Command	Subsection
rapt	Aperture the beam with a rectangular aperture.	7.18
rev	Replace existing transfer map by its reversed map.	7.11
revf	Replace existing transfer map by reverse factorized form.	7.12
rps1 ⋮ rps9	Random parameter set specification.	7.26
rt	Perform a ray trace.	7.2
shoa	Show contents of arrays.	7.44
sqr	Square the existing transfer map.	7.15
stm	Store the existing transfer map.	7.16
symp	Symplectify matrix portion of transfer map.	7.14
time	Write out execution time.	7.29
tmi	Input matrix elements and polynomial coefficients from an external file.	7.5
tmo	Ouptut matrix elements and polynomial coefficients to an external file.	7.6
tpol	Twiss polynomial.	7.37
tran	Replace existing transfer map by its “transpose”.	7.10
wcl	Write contents of a loop.	7.33
whst	Write history of beam loss.	7.21
wmrt	Write out value of merit function.	7.32
wnd	Window a beam.	7.20
wnda	Window a beam in all planes.	7.42
wps	Write out parameters in a parameter set.	7.28
wuca	Write out contents of ucalc array.	7.41
zer	Change or write out values of zeroes.	7.36

13.5 Simple Command Type-Code Mnemonics (Functional Order)

Simple commands and their type code mnemonics, as currently available in MaryLie 3.0, are listed below according to function along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Command	Subsection
end	Halt execution. Must be an entry of a #labor list.	7.1
of	Open files.	7.24
cf	Close files.	7.23
rt	Perform a ray trace.	7.2
num	Number lines in a file.	7.27
circ	Set parameters and circulate.	7.3
wcl	Write contents of a loop.	7.33
rapt	Aperture the beam with a rectangular aperture.	7.18
eapt	Aperture the beam with an elliptic aperture.	7.19
wnd	Window a beam.	7.20
wnda	Window a beam in all planes.	7.42
whst	Write history of beam loss.	7.21
dims	Dimensions.	7.40
pmif	Print contents of Master Input File (file 11).	7.4
ptm	Print transfer map.	7.7
pli	Path length information.	7.43
tmi	Input matrix elements and polynomial coefficients from an external file.	7.5
tmo	Ouptut matrix elements and polynomial coefficients to an external file.	7.6
ps1 ⋮ ps9	Parameter set specification.	7.25
rps1 ⋮ rps9	Random parameter set specification.	7.26
wps	Write out parameters in a parameter set.	7.28
stm	Store the existing transfer map.	7.16

Type Code	Command	Subsection
gtm	Get a transfer map from storage.	7.17
mask	Mask off selected portions of existing transfer map.	7.13
ftm	Filter the existing transfer map.	7.22
sqr	Square the existing transfer map.	7.15
symp	Symplectify matrix portion of transfer map.	7.14
iden	Replace existing transfer map by the identity map.	7.8
inv	Replace existing transfer map by its inverse.	7.9
rev	Replace existing transfer map by its reversed map.	7.11
revf	Replace existing transfer map by reverse factorized form.	7.12
tran	Replace existing transfer map by its "transpose".	7.10
tpol	Twiss polynomial.	7.37
dpol	Dispersion polynomial.	7.38
time	Write out execution time.	7.29
cdf	Change drop file.	7.30
bell	Ring bell at terminal.	7.31
wmrt	Write out value of merit function.	7.32
paws	Pause.	7.34
inf	Change or write out values of infinities.	7.35
zer	Change or write out values of zeroes.	7.36
cbm	Change or write out beam parameters.	7.39
wuca	Write out contents of ucalc array.	7.41
shoa	Show contents of arrays.	7.44

13.6 Advanced Command Type-Code Mnemonics (Alphabetical Order)

Advanced commands and their type code mnemonics, as currently available in MaryLie 3.0, are listed below alphabetically along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Command	Subsection
amap	Apply map to a function or moments.	8.17
asni	Apply script N inverse.	8.29
bgen	Generate beam.	8.34
cod	Compute off-momentum closed orbit data.	8.1
csym	Check symplectic condition.	8.31
ctr	Change tune range.	8.28
cxp	Chromatic expansion.	8.24
dia	Dynamic invariant analysis.	8.11
dnor	Dynamic normal form analysis.	8.9
exp	Compute exponential.	8.7
fadm	Fourier analyze dynamic map.	8.16
fasm	Fourier analyze static map.	8.15
fwa	Copy file to working array.	8.39
gbuf	Get buffer contents.	8.14
geom	Compute geometry of a loop.	8.38
lnf	Compute logarithm of normal form.	8.40
merf	Merge files.	8.41
mn	Compute matrix norm.	8.33
moma	Moment analysis.	8.37
padd	Add two polynomials.	8.19
pb	Poisson bracket two polynomials.	8.21
pdnf	Compute power of dynamic normal form.	8.13
pmul	Multiply two polynomials.	8.20
pnlp	Compute power of nonlinear part.	8.30
pold	Polar decomposition of a map.	8.22
ppa	Principal planes analysis.	8.36

Type Code	Command	Subsection
psnf	Compute power of static normal form.	8.12
psp	Polynomial scalar product.	8.32
pval	Evaluate a polynomial.	8.23
radm	Resonance analyze dynamic map.	8.5
rasm	Resonance analyze static map.	8.4
sia	Static invariant analysis.	8.10
smul	Multiply a polynomial by a scalar.	8.18
snor	Static normal form analysis.	8.8
sq	Select quantities.	8.26
tadm	Twiss analyze dynamic map.	8.3
tasm	Twiss analyze static map.	8.2
tbas	Translate basis.	8.6
tic	Translate initial conditions.	8.35
tqm	Transfer quadratic moments.	8.25
wsq	Write selected quantities.	8.27

13.7 Advanced Command Type-Code Mnemonics (Functional Order)

Advanced commands and their type code mnemonics, as currently available in MaryLie 3.0, are listed below according to function along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Command	Subsection
cod	Compute off-momentum closed orbit data.	8.1
tasm	Twiss analyze static map.	8.2
cxp	Chromatic expansion.	8.24
tadm	Twiss analyze dynamic map.	8.3
ctr	Change tune range.	8.28
snor	Static normal form analysis.	8.8
dnor	Dynamic normal form analysis.	8.9
asni	Apply script N inverse.	8.29
rasm	Resonance analyze static map.	8.4
radm	Resonance analyze dynamic map.	8.5
sia	Static invariant analysis.	8.10
dia	Dynamic invariant analysis.	8.11
psnf	Compute power of static normal form.	8.12
pdnf	Compute power of dynamic normal form.	8.13
pnlp	Compute power of nonlinear part.	8.30
fasm	Fourier analyze static map.	8.15
fadm	Fourier analyze dynamic map.	8.16
pold	Polar decomposition of a map.	8.22
ppa	Principal planes analysis.	8.36
tbas	Translate basis.	8.6
exp	Compute exponential.	8.7
lnf	Compute logarithm of normal form.	8.41
gbuf	Get buffer contents.	8.14
amap	Apply map to a function or moments.	8.17
moma	Moment analysis.	8.37
tqm	Transfer quadratic moments.	8.25

Type Code	Command	Subsection
bgen	Generate beam.	8.34
tic	Translate initial conditions.	8.35
smul	Multiply a polynomial by a scalar.	8.18
padd	Add two polynomials.	8.19
pmul	Multiply two polynomials.	8.20
pb	Poisson bracket two polynomials.	8.21
mn	Compute matrix norm.	8.33
psp	Polynomial scalar product.	8.32
pval	Evaluate a polynomial.	8.23
sq	Select quantities.	8.26
wsq	Write selected quantities.	8.27
csym	Check symplectic condition.	8.31
geom	Compute geometry of a loop.	8.38
fwa	Copy file to working array.	8.39
merf	Merge files.	8.40

13.8 Procedures and Fitting and Optimization Commands (Alphabetical Order)

Procedures and fitting and optimization commands and their type code mnemonics, as currently available in MaryLie 3.0, and listed below alphabetically along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Procedure/Command	Subsection
aim	Specify quantities to be fit or optimized and set target values.	9.5
bip	Begin inner procedure.	9.1
bop	Begin outer procedure.	9.2
con1 ⋮ con5	Constraints.	9.12
cps1 ⋮ cps9	Capture parameter set.	9.15
fit	Carry out fitting operation.	9.7
flag	Change or write out values of flags and defaults.	9.17
fps	Free parameter set.	9.16
grad	Compute gradient matrix.	9.13
mrt0	Merit function (sum of squares).	9.10
mrt1 ⋮ mrt5	Merit functions (user written).	9.11
mss	Minimize sum of squares optimization.	9.8
opt	General optimization.	9.9
rset	Reset menu entries.	9.18
scan	Scan parameter space.	9.14
tip	Terminate inner procedure.	9.3
top	Terminate outer procedure.	9.4
vary	Specify quantities to be varied.	9.6

13.9 Procedures and Fitting and Optimization Commands (Functional Order)

Procedures and fitting and optimization commands and their type code mnemonics, as currently availably in MaryLie 3.0, are listed below according to function along with the subsections that describe them in detail and/or give examples of their use.

Type Code	Procedure/Command	Subsection
bip	Begin inner procedure.	9.1
bop	Begin outer procedure.	9.2
tip	Terminate inner procedure.	9.3
top	Terminate outer procedure.	9.4
aim	Specify quantities to be fit or optimized and set target values.	9.5
vary	Specify quantities to be varied.	9.6
fit	Carry out fitting operation.	9.7
mss	Minimize sum of squares optimization.	9.8
opt	General optimization.	9.9
mrt0	Merit function (sum of squares).	9.10
mrt1 ⋮ mrt5	Merit functions (user written).	9.11
con1 ⋮ con5	Constraints.	9.12
grad	Compute gradient matrix.	9.13
scan	Scan parameter space.	9.14
cps1 ⋮ cps9	Capture parameter set.	9.15
fps	Free parameter set.	9.16
flag	Change or write out values of flags and defaults.	9.17
rset	Reset menu entries.	9.18

13.10 Input File Functions and Formats

Listed below are input files used by MaryLie (and PREP) along with the subsections that describe them in detail.

File	Function	Subsections
5	Terminal input to PREP	5.1.1
	Terminal input to MARYLIE	9.5, 9.6
9	Restart for PREP	5.3, 5.14
11	Master Input File for MARYLIE	4.2, 4.4.1, 5.5.1, 5.6.3 5.7.2, 5.7.4, 5.8.2, 5.9.2 5.10.2, 5.11.2
∗	Phase Space Initial Conditions	4.2, 4.4.2, 7.2, 7.3
∗	Matrix and Polynomial Input	4.2, 4.4.3, 7.5, 7.6
∗	Data for Random Elements and Parameter Sets	4.2, 6.19, 7.27
∗	Data for Select Quantities Command	8.26
∗	Data for Moment Analysis	8.37
∗	Data for Fitting and Optimization Commands	9.5, 9.6

13.11 Output File Functions and Formats

Listed below are output files used by MaryLie (and PREP) along with the subsections that describe them in detail.

File	Function	Subsections
6	Terminal output from PREP and MaryLie	4.2, 4.5.1, 4.5.2, 5.5.1 7.2, 7.3, 7.7
10	Output from PREP	5.1.2
12	Extensive MARYLIE Output	4.2, 4.5.1, 4.5.2, 7.2, 7.3, 7.4, 7.7, 7.31
∗	Final Conditions from Ray Trace	4.2, 4.5.2, 7.2, 7.3
∗	Matrix and Polynomial Output	4.2, 4.4.3, 4.5.3, 7.6
∗	History of Beam Loss	7.22
∗	Results of Numbering Lines in a File	7.28
∗	Contents of a Loop	7.34
∗	Results of Evaluating a Polynomial	8.23
∗	Values of Selected Quantities	8.27
∗	Moment Data	8.34
∗	Transformed Moments	8.37
∗	Geometrical Information	8.38
∗	Data for Fitting and Optimization Commands	9.5, 9.6

13.12 Warning and Error Messages

Several MaryLie routines issue warning or error messages if there are detectable errors in the master input file or various quantities are out of bounds. Most messages are meant to be self explanatory. Some that may require further explanation are listed below.

1. "Eig4: Eigenvalues off the unit circle!";"Eig6: Eigenvalues off the unit circle!": The commands *tasm*, *tadm*, *snor*, *dnor*, *pold*, and *moma* require the calculation of the eigenvalues of 4×4 or 6×6 symplectic matrices, and these eigenvalues are expected to lie on the unit circle. If they do not, an error message is issued and the calculations associated with these commands are terminated. However, for whatever it is worth, the MaryLie run itself is not terminated.

Chapter 14

Tables of Indices, Exponents, and Basis Elements

Monomials in the usual Cartesian basis are labelled by an index (called the MaryLie index) using the Giorgilli scheme. (See the references in Chapter 11.) Consider the monomial

$$X^{j_1} P_x^{j_2} Y^{j_3} P_y^{j_4} \tau^{j_5} P_\tau^{j_6}, \tag{14.1}$$

where the exponents j_1 through j_6 are integers. For ℓ taking the values 1 through 6, define integers $n(\ell; j_1, \cdots j_6)$ by the relation

$$n(\ell; j_1, \cdots j_6) = \ell - 1 + \sum_{k=0}^{\ell-1} j_{6-k}. \tag{14.2}$$

Then the monomial (14.1) is given an index i by the rule

$$i(j_1, \cdots j_6) = \sum_{\ell=1}^{6} \text{Binomial } [n(\ell; j_1, \cdots j_6), \ell]. \tag{14.3}$$

Here the quantities

$$\text{Binomial } [n, \ell] = \begin{pmatrix} n \\ \ell \end{pmatrix} = \begin{cases} 0 & n < \ell \\ \frac{n!}{\ell!(n-\ell)!} & n \geq \ell \end{cases} \tag{14.4}$$

are the usual binomial coefficients. The rule (14.3) establishes a one-to-one correspondence between the set $\{j_1, \cdots j_6\}$ of exponents and the positive integers i. Each index $i(j_1, \cdots j_6)$ corresponds to a unique 6-tuple of exponents $(j_1, \cdots j_6)$ and vice versa. These indices and their associated exponents are tabulated in section 14.1.

Static (τ independent) polynomials in the static resonance basis are designated using the notation

$$\begin{aligned} Rabcde &= \text{Re}[(X + iP_x)^a (X - iP_x)^b (Y + iP_y)^c (Y - iP_y)^d P_\tau^e], \\ Iabcde &= \text{Im}[(X + iP_x)^a (X - iP_x)^b (Y + iP_y)^c (Y - iP_y)^d P_\tau^e]. \end{aligned} \tag{14.5}$$

Here the quantities (exponents) a through e are integers. If a term contains τ, the Cartesian basis is employed, and the monomial is designated by all six exponents j_1 through j_6 just as

in the Cartesian case. The polynomials in the static resonance basis, along with their static resonance MaryLie indices, are listed in section 14.2. The static resonance MaryLie indices have been assigned in such a way that various related resonant terms appear consecutively.

Polynomials in the dynamic resonance basis are designated by the notation

$$Rabcdef = \mathrm{Re}[(X + iP_x)^a (X - iP_x)^b (Y + iP_y)^c (Y - iP_y)^d (\tau + iP_\tau)^e (\tau - iP_\tau)^f],$$

$$Iabcdef = \mathrm{Im}[(X + iP_x)^a (X - iP_x)^b (Y + iP_y)^c (Y - iP_y)^d (\tau + iP_\tau)^e (\tau - iP_\tau)^f]. \qquad (14.6)$$

The polynomials in the dynamic resonance basis, along with their dynamic resonance MaryLie indices, are listed in section 14.3. The dynamic resonance MaryLie indices have been assigned in such a way that various related resonant terms appear consecutively.

14.1 MaryLie Indices for Monomials in Cartesian Basis

Index	Exponents of		
	$X\ P_x$	$Y\ P_y$	$\tau\ P_\tau$
1	1 0	0 0	0 0
2	0 1	0 0	0 0
3	0 0	1 0	0 0
4	0 0	0 1	0 0
5	0 0	0 0	1 0
6	0 0	0 0	0 1
7	2 0	0 0	0 0
8	1 1	0 0	0 0
9	1 0	1 0	0 0
10	1 0	0 1	0 0
11	1 0	0 0	1 0
12	1 0	0 0	0 1
13	0 2	0 0	0 0
14	0 1	1 0	0 0
15	0 1	0 1	0 0
16	0 1	0 0	1 0
17	0 1	0 0	0 1
18	0 0	2 0	0 0
19	0 0	1 1	0 0
20	0 0	1 0	1 0
21	0 0	1 0	0 1
22	0 0	0 2	0 0
23	0 0	0 1	1 0
24	0 0	0 1	0 1
25	0 0	0 0	2 0
26	0 0	0 0	1 1
27	0 0	0 0	0 2
28	3 0	0 0	0 0
29	2 1	0 0	0 0
30	2 0	1 0	0 0
31	2 0	0 1	0 0
32	2 0	0 0	1 0
33	2 0	0 0	0 1
34	1 2	0 0	0 0
35	1 1	1 0	0 0

Index	Exponents of		
	$X\ P_x$	$Y\ P_y$	$\tau\ P_\tau$
36	1 1	0 1	0 0
37	1 1	0 0	1 0
38	1 1	0 0	0 1
39	1 0	2 0	0 0
40	1 0	1 1	0 0
41	1 0	1 0	1 0
42	1 0	1 0	0 1
43	1 0	0 2	0 0
44	1 0	0 1	1 0
45	1 0	0 1	0 1
46	1 0	0 0	2 0
47	1 0	0 0	1 1
48	1 0	0 0	0 2
49	0 3	0 0	0 0
50	0 2	1 0	0 0
51	0 2	0 1	0 0
52	0 2	0 0	1 0
53	0 2	0 0	0 1
54	0 1	2 0	0 0
55	0 1	1 1	0 0
56	0 1	1 0	1 0
57	0 1	1 0	0 1
58	0 1	0 2	0 0
59	0 1	0 1	1 0
60	0 1	0 1	0 1
61	0 1	0 0	2 0
62	0 1	0 0	1 1
63	0 1	0 0	0 2
64	0 0	3 0	0 0
65	0 0	2 1	0 0
66	0 0	2 0	1 0
67	0 0	2 0	0 1
68	0 0	1 2	0 0
69	0 0	1 1	1 0
70	0 0	1 1	0 1

Index	Exponents of		
	$X\ P_x$	$Y\ P_y$	$\tau\ P_\tau$
71	0 0	1 0	2 0
72	0 0	1 0	1 1
73	0 0	1 0	0 2
74	0 0	0 3	0 0
75	0 0	0 2	1 0
76	0 0	0 2	0 1
77	0 0	0 1	2 0
78	0 0	0 1	1 1
79	0 0	0 1	0 2
80	0 0	0 0	3 0
81	0 0	0 0	2 1
82	0 0	0 0	1 2
83	0 0	0 0	0 3
84	4 0	0 0	0 0
85	3 1	0 0	0 0
86	3 0	1 0	0 0
87	3 0	0 1	0 0
88	3 0	0 0	1 0
89	3 0	0 0	0 1
90	2 2	0 0	0 0
91	2 1	1 0	0 0
92	2 1	0 1	0 0
93	2 1	0 0	1 0
94	2 1	0 0	0 1
95	2 0	2 0	0 0
96	2 0	1 1	0 0
97	2 0	1 0	1 0
98	2 0	1 0	0 1
99	2 0	0 2	0 0
100	2 0	0 1	1 0
101	2 0	0 1	0 1
102	2 0	0 0	2 0
103	2 0	0 0	1 1
104	2 0	0 0	0 2
105	1 3	0 0	0 0

Index	Exponents of		
	$X\ P_x$	$Y\ P_y$	$\tau\ P_\tau$
106	1 2	1 0	0 0
107	1 2	0 1	0 0
108	1 2	0 0	1 0
109	1 2	0 0	0 1
110	1 1	2 0	0 0
111	1 1	1 1	0 0
112	1 1	1 0	1 0
113	1 1	1 0	0 1
114	1 1	0 2	0 0
115	1 1	0 1	1 0
116	1 1	0 1	0 1
117	1 1	0 0	2 0
118	1 1	0 0	1 1
119	1 1	0 0	0 2
120	1 0	3 0	0 0
121	1 0	2 1	0 0
122	1 0	2 0	1 0
123	1 0	2 0	0 1
124	1 0	1 2	0 0
125	1 0	1 1	1 0
126	1 0	1 1	0 1
127	1 0	1 0	2 0
128	1 0	1 0	1 1
129	1 0	1 0	0 2
130	1 0	0 3	0 0
131	1 0	0 2	1 0
132	1 0	0 2	0 1
133	1 0	0 1	2 0
134	1 0	0 1	1 1
135	1 0	0 1	0 2
136	1 0	0 0	3 0
137	1 0	0 0	2 1
138	1 0	0 0	1 2
139	1 0	0 0	0 3
140	0 4	0 0	0 0

Index	Exponents of		
	$X\ P_x$	$Y\ P_y$	$\tau\ P_\tau$
141	0 3	1 0	0 0
142	0 3	0 1	0 0
143	0 3	0 0	1 0
144	0 3	0 0	0 1
145	0 2	2 0	0 0
146	0 2	1 1	0 0
147	0 2	1 0	1 0
148	0 2	1 0	0 1
149	0 2	0 2	0 0
150	0 2	0 1	1 0
151	0 2	0 1	0 1
152	0 2	0 0	2 0
153	0 2	0 0	1 1
154	0 2	0 0	0 2
155	0 1	3 0	0 0
156	0 1	2 1	0 0
157	0 1	2 0	1 0
158	0 1	2 0	0 1
159	0 1	1 2	0 0
160	0 1	1 1	1 0
161	0 1	1 1	0 1
162	0 1	1 0	2 0
163	0 1	1 0	1 1
164	0 1	1 0	0 2
165	0 1	0 3	0 0
166	0 1	0 2	1 0
167	0 1	0 2	0 1
168	0 1	0 1	2 0
169	0 1	0 1	1 1
170	0 1	0 1	0 2
171	0 1	0 0	3 0
172	0 1	0 0	2 1
173	0 1	0 0	1 2
174	0 1	0 0	0 3
175	0 0	4 0	0 0

Index	Exponents of		
	$X\ P_x$	$Y\ P_y$	$\tau\ P_\tau$
176	0 0	3 1	0 0
177	0 0	3 0	1 0
178	0 0	3 0	0 1
179	0 0	2 2	0 0
180	0 0	2 1	1 0
181	0 0	2 1	0 1
182	0 0	2 0	2 0
183	0 0	2 0	1 1
184	0 0	2 0	0 2
185	0 0	1 3	0 0
186	0 0	1 2	1 0
187	0 0	1 2	0 1
188	0 0	1 1	2 0
189	0 0	1 1	1 1
190	0 0	1 1	0 2
191	0 0	1 0	3 0
192	0 0	1 0	2 1
193	0 0	1 0	1 2
194	0 0	1 0	0 3
195	0 0	0 4	0 0
196	0 0	0 3	1 0
197	0 0	0 3	0 1
198	0 0	0 2	2 0
199	0 0	0 2	1 1
200	0 0	0 2	0 2
201	0 0	0 1	3 0
202	0 0	0 1	2 1
203	0 0	0 1	1 2
204	0 0	0 1	0 3
205	0 0	0 0	4 0
206	0 0	0 0	3 1
207	0 0	0 0	2 2
208	0 0	0 0	1 3
209	0 0	0 0	0 4

14.2 MaryLie Indices for Polynomials in Static Resonance Basis

SR Index	Cart Index	Designation	Element
1	6	R00001	P_τ
2	1	R10000	X
3	2	I10000	P_x
4	3	R00100	Y
5	4	I00100	P_y
6	5	000010	τ
7	-	R11000	$X^2 + P_x^2$
8	-	R00110	$Y^2 + P_y^2$
9	27	R00002	P_τ^2
10	21	R10001	XP_τ
11	17	I10001	P_xP_τ
12	21	R00101	YP_τ
13	24	I00101	P_yP_τ
14	-	R20000	$X^2 - P_x^2$
15	-	I20000	$2XP_x$
16	-	R00200	$Y^2 - P_y^2$
17	-	I00200	$2YP_y$
18	-	R10100	$XY - P_xP_y$
19	-	I10100	$XP_y + YP_x$
20	-	R10010	$XY + P_xP_y$
21	-	I10010	$-XP_y + YP_x$
22	11	100010	$X\tau$
23	16	010010	$P_x\tau$
24	20	001010	$Y\tau$
25	23	000110	$P_y\tau$
26	25	000020	τ^2
27	26	000011	τP_τ
28	-	R11001	$(X^2 + P_x^2)P_\tau$
29	-	R00111	$(Y^2 + P_y^2)P_\tau$
30	83	R00003	P_τ^3
31	48	R10002	XP_τ^2
32	63	I10002	$P_xP_\tau^2$
33	73	R00102	YP_τ^2
34	79	I00102	$P_yP_\tau^2$
35	-	R20001	$(X^2 - P_x^2)P_\tau$

SR Index	Cart Index	Designation	Element
36	-	I20001	$2XP_xP_\tau$
37	-	R00201	$(Y^2 - P_y^2)P_\tau$
38	-	I00201	$2YP_yP_\tau$
39	-	R10101	$(XY - P_xP_y)P_\tau$
40	-	I10101	$(XP_y + YP_x)P_\tau$
41	-	R10011	$(XY + P_xP_y)P_\tau$
42	-	I10011	$(-XP_y + YP_x)P_\tau$
43	-	R21000	$X(X^2 + P_x^2)$
44	-	I21000	$P_x(X^2 + P_x^2)$
45	-	R00210	$Y(Y^2 + P_y^2)$
46	-	I00210	$P_y(Y^2 + P_y^2)$
47	-	R10110	$X(Y^2 + P_y^2)$
48	-	I10110	$P_x(Y^2 + P_y^2)$
49	-	R11100	$Y(X^2 + P_x^2)$
50	-	I11100	$P_y(X^2 + P_x^2)$
51	-	R30000	$X^3 - 3XP_x^2$
52	-	I30000	$3X^2P_x - P_x^3$
53	-	R00300	$Y^3 - 3YP_y^2$
54	-	I00300	$3Y^2P_y - P_y^3$
55	-	R20100	$X^2Y - P_x^2Y - 2XP_xP_y$
56	-	I20100	$X^2P_y - P_x^2P_y + 2XP_xY$
57	-	R10200	$XY^2 - XP_y^2 - 2P_xYP_y$
58	-	I10200	$P_xY^2 - P_xP_y^2 + 2XYP_y$
59	-	R20010	$X^2Y - P_x^2Y + 2XP_xP_y$
60	-	I20010	$-X^2P_y + P_x^2P_y + 2XP_xY$
61	-	R01200	$XY^2 - XP_y^2 + 2P_xYP_y$
62	-	I01200	$-P_xY^2 + P_xP_y^2 + 2XYP_y$
63	32	200010	$X^2\tau$
64	37	110010	$XP_x\tau$
65	41	101010	$XY\tau$
66	44	100110	$XP_y\tau$
67	46	100020	$X\tau^2$
68	47	100011	$X\tau P_\tau$
69	52	020010	$P_x^2\tau$
70	56	011010	$P_xY\tau$

SR Index	Cart Index	Designation	Element
71	59	010110	$P_x P_y \tau$
72	61	010020	$P_x \tau^2$
73	62	010011	$P_x \tau P_\tau$
74	66	002010	$Y^2 \tau$
75	69	001110	$Y P_y \tau$
76	71	001020	$Y \tau^2$
77	72	001011	$Y \tau P_\tau$
78	75	000210	$P_y^2 \tau$
79	77	000120	$P_y \tau^2$
80	78	000111	$P_y \tau P_\tau$
81	80	000030	τ^3
82	81	000021	$\tau^2 P_\tau$
83	82	000012	τP_τ^2
84	-	R11002	$(X^2 + P_x^2) P_\tau^2$
85	-	R00112	$(Y^2 + P_y^2) P_\tau^2$
86	209	R00004	P_τ^4
87	-	R22000	$(X^2 + P_x^2)^2$
88	-	R00220	$(Y^2 + P_y^2)^2$
89	-	R11110	$(X^2 + P_x^2)(Y^2 + P_y^2)$
90	139	R10003	$X P_\tau^3$
91	174	I10003	$P_x P_\tau^3$
92	194	R00103	$Y P_\tau^3$
93	204	I00103	$P_y P_\tau^3$
94	-	R20002	$(X^2 - P_x^2) P_\tau^2$
95	-	I20002	$2 X P_x P_\tau^2$
96	-	R00202	$(Y^2 - P_y^2) P_\tau^2$
97	-	I00202	$2 Y P_y P_\tau^2$
98	-	R10102	$(XY - P_x P_y) P_\tau^2$
99	-	I10102	$(X P_y + Y P_x) P_\tau^2$
100	-	R10012	$(XY + P_x P_y) P_\tau^2$
101	-	I10012	$(-X P_y + Y P_x) P_\tau^2$
102	-	R21001	$X(X^2 + P_x^2) P_\tau$
103	-	I21001	$P_x(X^2 + P_x^2) P_\tau$
104	-	R00211	$Y(Y^2 + P_y^2) P_\tau$
105	-	I00211	$P_y(Y^2 + P_y^2) P_\tau$

SR Index	Cart Index	Designation	Element
106	-	R10111	$X(Y^2+P_y^2)P_\tau$
107	-	I10111	$P_x(Y^2+P_y^2)P_\tau$
108	-	R11101	$Y(X^2+P_x^2)P_\tau$
109	-	I11101	$P_y(X^2+P_x^2)P_\tau$
110	-	R30001	$(X^3-3XP_x^2)P_\tau$
111	-	I30001	$(3X^2P_x-P_x^3)P_\tau$
112	-	R00301	$(Y^3-3YP_y^2)P_\tau$
113	-	I00301	$(3Y^2P_y\tau-P_y^3)P_\tau$
114	-	R20101	$(X^2Y-P_x^2Y-2XP_xP_y)P_\tau$
115	-	I20101	$(X^2P_y-P_x^2P_y+2XP_xP_y)P_\tau$
116	-	R10201	$(XY^2-XP_y^2-2P_xYP_y)P_\tau$
117	-	I10201	$(P_xY^2-P_xP_y^2+2XYP_y)P_\tau$
118	-	R20011	$(X^2Y-P_x^2Y+2XP_xP_y)P_\tau$
119	-	I20011	$(-X^2P_y+P_x^2P_y+2XP_xY)P_\tau$
120	-	R01201	$(XY^2-XP_y^2+2P_xYP_y)P_\tau$
121	-	I01201	$(-P_xY^2+P_xP_y^2+2XYP_y0P_\tau$
122	-	R31000	$X^4-P_x^4$
123	-	I31000	$2X^3P_x+2XP_x^3$
124	-	R00310	$Y^4-P_y^4$
125	-	I00310	$2Y^3P_y+2YP_y^3$
126	-	R20110	$(X^2-P_x^2)(Y^2+P_y^2)$
127	-	I20110	$2XP_x(Y^2P_y^2)$
128	-	R11200	$(X^2+P_x^2)(Y^2-P_y^2)$
129	-	I11200	$2(X^2+P_x^2)YP_y$
130	-	R21100	$(X^2+P_x^2)(XY-P_xP_y)$
131	-	I21100	$(X^2+P_x^2)(XP_y+YP_x)$
132	-	R10210	$(Y^2+P_y^2)(XY-P_xP_y)$
133	-	I10210	$(Y^2+P_y^2)(XP_y+YP_x)$
134	-	R21010	$(X^2+P_x^2)(XY+P_xP_y)$
135	-	I21010	$(X^2+P_x^2)(-XP_y+YP_x)$
136	-	R01210	$(Y^2+P_y^2)(XY+P_xP_y)$
137	-	I01210	$(Y^2+P_y^2)(XP_y-YP_x)$
138	-	R40000	$X^4-6X^2P_x^2+P_x^4$
139	-	I40000	$4X^3P_x-4XP_x^3$
140	-	R00400	$Y^4-6Y^2P_y^2+P_y^4$

SR Index	Cart Index	Designation	Element
141	-	I00400	$4Y^3P_y - 4YP_y^3$
142	-	R30100	$X^3Y - 3XP_x^2Y - 3X^2P_xP_y + P_x^3P_y$
143	-	I30100	$X^3P_y + 3X^2P_xY - 3XP_x^2P_y - P_x^3Y$
144	-	R10300	$XY^3 - 3P_xY^2P_y - 3XYP_y^2 + P_xP_y^3$
145	-	I10300	$P_xY^3 + 3XY^2P_y - 3P_xYP_y^2 - XP_y^3$
146	-	R30010	$X^3Y - 3XP_x^2Y + 3X^2P_xP_y - P_x^3P_y$
147	-	I30010	$-X^3P_y + 3X^2P_xY + 3XP_x^2P_y - P_x^3Y$
148	-	R01300	$XY^3 + 3P_xY^2P_y - 3XYP_y^2 - P_xP_y^3$
149	-	I01300	$-P_xY^3 + 3XY^2P_y + 3P_xYP_y^2 - XP_y^3$
150	-	R20200	$X^2Y^2 - P_x^2Y^2 - X^2P_y^2 + P_x^2P_y^2 - 4XP_xYP_y$
151	-	I20200	$2XP_xY^2 + 2X^2YP_y - 2P_x^2YP_y - 2XP_xP_y^2$
152	-	R20020	$X^2Y^2 - P_x^2Y^2 - X^2P_y^2 + P_x^2P_y^2 + 4XP_xYP_y$
153	-	I20020	$2XP_xY^2 - 2X^2YP_y + 2P_x^2YP_y - 2XP_xP_y^2$
154	88	300010	$x^3\tau$
155	93	210010	$x^2P_x\tau$
156	97	201010	$x^2Y\tau$
157	100	200110	$x^2P_y\tau$
158	102	200020	$x^2\tau^2$
159	103	200011	$x^2\tau P_\tau$
160	108	120010	$XP_x^2\tau$
161	112	111010	$XP_xY\tau$
162	115	110110	$XP_xP_y\tau$
163	117	110020	$XP_x\tau^2$
164	118	110011	$XP_x\tau P_\tau$
165	122	102010	$XY^2\tau$
166	125	101110	$XYP_y\tau$
167	127	101020	$XY\tau^2$
168	128	101011	$XY\tau P_\tau$
169	131	100210	$XP_y^2\tau$
170	133	100120	$XP_y\tau^2$
171	134	100111	$XP_y\tau P_\tau$
172	136	100030	$X\tau^3$
173	137	100021	$X\tau^2P_\tau$
174	138	100012	$X\tau P_\tau^2$
175	143	030010	$P_X^3\tau$

SR Index	Cart Index	Designation	Element
176	147	021010	$P_x^2 Y \tau$
177	150	020110	$P_x^2 P_Y \tau$
178	152	020020	$P_x^2 \tau^2$
179	153	020011	$P_x^2 \tau P_\tau$
180	157	012010	$P_x Y^2 \tau$
181	160	011110	$P_x Y P_y \tau$
182	162	011020	$P_x Y \tau^2$
183	163	011011	$P_x Y \tau P_\tau$
184	166	010210	$P_x P_y^2 \tau$
185	168	010120	$P_X P_y \tau^2$
186	169	010111	$P_x P_y \tau P_\tau$
187	171	010030	$P_x \tau^3$
188	172	010021	$P_x \tau^2 P_\tau$
189	173	010012	$P_x \tau P_\tau^2$
190	177	003010	$Y^3 \tau$
191	180	002110	$Y^2 P_y \tau$
192	182	002020	$Y^2 \tau^2$
193	183	002011	$Y^2 \tau P_\tau$
194	186	001210	$Y P_y^2 \tau$
195	188	001120	$Y P_y \tau^2$
196	189	001111	$Y P_y \tau P_\tau$
197	191	001030	$Y \tau^3$
198	192	001021	$Y \tau^2 P_\tau$
199	193	001012	$Y \tau P_\tau^2$
200	196	000310	$P_y^3 \tau$
201	198	000220	$P_y^2 \tau^2$
202	199	000211	$P_y^2 \tau P_\tau$
203	201	000130	$P_y \tau^3$
204	202	000121	$P_y \tau^2 P_\tau$
205	203	000112	$P_y \tau P_\tau^2$
206	205	000040	τ^4
207	206	000031	$\tau^3 P_\tau$
208	207	000022	$\tau^2 P_\tau^2$
209	208	000013	τP_τ^3

14.3 MaryLie Indices for Polynomials in Dynamic Resonance Basis

DR Index	Designation	Element
1	R100000	X
2	I100000	P_x
3	R001000	Y
4	I001000	P_y
5	R000010	τ
6	I000010	P_τ
7	R110000	$X^2 + P_x^2$
8	R001100	$Y^2 + P_y^2$
9	R000011	$\tau^2 + P_\tau^2$
10	R000020	$\tau^2 - P_\tau^2$
11	I000020	$2\tau P_\tau$
12	R100010	$X\tau - P_x P_\tau$
13	I100010	$XP_\tau + P_x\tau$
14	R100001	$X\tau + P_x P_\tau$
15	I100001	$-XP_\tau + P_x\tau$
16	R001010	$Y\tau - P_y P_\tau$
17	I001010	$YP_\tau + P_y\tau$
18	R001001	$Y\tau + P_y P_\tau$
19	I001001	$-YP_\tau + P_y\tau$
20	R200000	$X^2 - P_x^2$
21	I200000	$2XP_x$
22	R002000	$Y^2 - P_y^2$
23	I002000	$2YP_y$
24	R101000	$XY - P_x P_y$
25	I101000	$XP_y + P_x Y$
26	R100100	$XY + P_x P_y$
27	I100100	$-XP_y + P_x Y$
28	R110010	$X^2\tau + P_x^2\tau$
29	I110010	$X^2 P_\tau + P_x^2 P_\tau$
30	R001110	$Y^2\tau + P_y^2\tau$
31	I001110	$Y^2 P_\tau + P_y^2 P_\tau$
32	R000030	$\tau^3 - 3\tau P_\tau^2$
33	I000030	$3\tau^2 P_\tau - P_\tau^3$
34	R000021	$\tau^3 + \tau P_\tau^2$
35	I000021	$\tau^2 P_\tau + P_\tau^3$

DR Index	Designation	Element
36	R100011	$X\tau^2 + XP_\tau^2$
37	I100011	$P_x\tau^2 + P_xP_\tau^2$
38	R001011	$Y\tau^2 + YP_\tau^2$
39	I001011	$P_y\tau^2 + P_yP_\tau^2$
40	R100020	$X\tau^2 - XP_\tau^2 - 2P_x\tau P_\tau$
41	I100020	$2X\tau P_\tau + P_x\tau^2 - P_xP_\tau^2$
42	R100002	$X\tau^2 - XP_\tau^2 + 2P_x\tau P_\tau$
43	I100002	$-2X\tau P_\tau + P_x\tau^2 - P_xP_\tau^2$
44	R001020	$Y\tau^2 - YP_\tau^2 - 2P_y\tau P_\tau$
45	I001020	$2Y\tau P_\tau + P_y\tau^2 - P_yP_\tau^2$
46	R001002	$Y\tau^2 - YP_\tau^2 + 2P_y\tau P_\tau$
47	I001002	$-2Y\tau P_\tau + P_y\tau^2 - P_yP_\tau^2$
48	R200010	$X^2\tau - 2XP_xP_\tau - P_x^2\tau$
49	I200010	$X^2P_\tau + 2XP_x\tau - P_x^2P_\tau$
50	R200001	$X^2\tau + 2XP_xP_\tau - P_x^2\tau$
51	I200001	$-X^2P_\tau + 2XP_x\tau + P_x^2P_\tau$
52	R002010	$Y^2\tau - 2YP_yP_\tau - P_y^2\tau$
53	I002010	$Y^2P_\tau + 2YP_y\tau - P_y^2P_\tau$
54	R002001	$Y^2\tau + 2YP_yP_\tau - P_y^2\tau$
55	I002001	$-Y^2P_\tau + 2YP_y\tau + P_y^2P_\tau$
56	R101010	$XY\tau - XP_yP_\tau - P_xYP_\tau - P_xP_y\tau$
57	I101010	$XYP_\tau + XP_y\tau + P_xY\tau - P_xP_yP_\tau$
58	R101001	$XY\tau + XP_yP_\tau + P_xYP_\tau - P_xP_y\tau$
59	I101001	$-XYP_\tau + XP_y\tau + P_xY\tau + P_xP_yP_\tau$
60	R100110	$XY\tau + XP_yP_\tau - P_xYP_\tau + P_xP_y\tau$
61	I100110	$XYP_\tau - XP_y\tau + P_xY\tau + P_xP_yP_\tau$
62	R100101	$XY\tau - XP_yP_\tau + P_xYP_\tau + P_xP_y\tau$
63	I100101	$-XYP_\tau - XP_y\tau + P_xY\tau - P_xP_yP_\tau$
64	R210000	$X^3 + XP_x^2$
65	I210000	$X^2P_x + P_x^3$
66	R002100	$Y^3 + YP_y^2$
67	I002100	$Y^2P_y + P_y^3$
68	R101100	$XY^2 + XP_y^2$
69	I101100	$P_xY^2 + P_xP_y^2$
70	R111000	$X^2Y + P_x^2Y$

DR Index	Designation	Element
71	I111000	$X^2P_y + P_x^2P_y$
72	R300000	$X^3 - 3XP_x^2$
73	I300000	$3X^2P_x - P_x^3$
74	R003000	$Y^3 - 3YP_y^2$
75	I003000	$3Y^2P_y - P_y^3$
76	R201000	$X^2Y - 2XP_xP_y - P_x^2Y$
77	I201000	$X^2P_y + 2XP_xY - P_x^2P_y$
78	R102000	$XY^2 - XP_y^2 - 2P_xYP_y$
79	I102000	$2XYP_y + P_xY^2 - P_xP_y^2$
80	R200100	$X^2Y + 2XP_xP_y - P_x^2Y$
81	I200100	$-X^2P_y + 2XP_xY + P_x^2P_y$
82	R100200	$XY^2 - XP_y^2 + 2P_xYP_y$
83	I100200	$-2XYP_y + P_xY^2 - P_xP_y^2$
84	R220000	$X^4 + 2X^2P_x^2 + P_x^4$
85	R002200	$Y^4 + 2Y^2P_y^2 + P_y^4$
86	R000022	$\tau^4 + 2\tau^2P_\tau^2 + P_\tau^4$
87	R111100	$X^2Y^2 + X^2P_y^2 + P_x^2Y^2 + P_x^2P_y^2$
88	R110011	$X^2\tau^2 + X^2P_\tau^2 + P_x^2\tau^2 + P_x^2P_\tau^2$
89	R001111	$Y^2\tau^2 + Y^2P_\tau^2 + P_y^2\tau^2 + P_y^2P_\tau^2$
90	R110020	$X^2\tau^2 - X^2P_\tau^2 + P_x^2\tau^2 - P_x^2P_\tau^2$
91	I110020	$2X^2\tau P_\tau + 2P_x^2\tau P_\tau$
92	R001120	$Y^2\tau^2 - Y^2P_\tau^2 + P_y^2\tau^2 - P_y^2P_\tau^2$
93	I001120	$2Y^2\tau P_\tau + 2P_y^2\tau P_\tau$
94	R000040	$\tau^4 - 6\tau^2P_\tau^2 + P_\tau^4$
95	I000040	$4\tau^3P_\tau - 4\tau P_\tau^3$
96	R000031	$\tau^4 - P_\tau^4$
97	I000031	$2\tau^3P_\tau + 2\tau P_\tau^3$
98	R100030	$X\tau^3 - 3X\tau P_\tau^2 - 3P_x\tau^2P_\tau + P_xP_\tau^3$
99	I100030	$3X\tau^2P_\tau - XP_\tau^3 + P_x\tau^3 - 3P_x\tau P_\tau^2$
100	R100003	$X\tau^3 - 3X\tau P_\tau^2 + 3P_x\tau^2P_\tau - P_xP_\tau^3$
101	I100003	$-3X\tau^2P_\tau + XP_\tau^3 + P_x\tau^3 - 3P_x\tau P_\tau^2$
102	R001030	$Y\tau^3 - 3Y\tau P_\tau^2 - 3P_y\tau^2P_\tau + P_yP_\tau^3$
103	I001030	$3Y\tau^2P_\tau - YP_\tau^3 + P_y\tau^3 - 3P_y\tau P_\tau^2$
104	R001003	$Y\tau^3 - 3Y\tau P_\tau^2 + 3P_y\tau^2P_\tau - P_yP_\tau^3$
105	I001003	$-3Y\tau^2P_\tau + YP_\tau^3 + P_y\tau^3 - 3P_y\tau P_\tau^2$

DR Index	Designation	Element
106	R100021	$X\tau^3 + X\tau P_\tau^2 - P_x\tau^2 P_\tau - P_x P_\tau^3$
107	I100021	$X\tau^2 P_\tau + X P_\tau^3 + P_x\tau^3 + P_x\tau P_\tau^2$
108	R100012	$X\tau^3 + X\tau P_\tau^2 + P_x\tau^2 P_\tau + P_x P_\tau^3$
109	I100012	$-X\tau^2 P_\tau - X P_\tau^3 + P_x\tau^3 + P_x\tau P_\tau^2$
110	R001021	$Y\tau^3 + Y\tau P_\tau^2 - P_y\tau^2 P_\tau - P_y P_\tau^3$
111	I001021	$Y\tau^2 P_\tau + Y P_\tau^3 + P_y\tau^3 + P_y\tau P_\tau^2$
112	R001012	$Y\tau^3 + Y\tau P_\tau^2 + P_y\tau^2 P_\tau + P_y P_\tau^3$
113	I001012	$-Y\tau^2 P_\tau - Y P_\tau^3 + P_y\tau^3 + P_y\tau P_\tau^2$
114	R200011	$X^2\tau^2 + X^2 P_\tau^2 - P_x^2\tau^2 - P_x^2 P_\tau^2$
115	I200011	$2X P_x\tau^2 + 2X P_x P_\tau^2$
116	R002011	$Y^2\tau^2 + Y^2 P_\tau^2 - P_y^2\tau^2 - P_y^2 P_\tau^2$
117	I002011	$2Y P_y\tau^2 + 2Y P_y P_\tau^2$
118	R200020	$X^2\tau^2 - X^2 P_\tau^2 - 4X P_x\tau P_\tau - P_x^2\tau^2 + P_x^2 P_\tau^2$
119	I200020	$2X^2\tau P_\tau + 2X P_x\tau^2 - 2X P_x P_\tau^2 - 2P_x^2\tau P_\tau$
120	R200002	$X^2\tau^2 - X^2 P_\tau^2 + 4X P_x\tau P_\tau - P_x^2\tau^2 + P_x^2 P_\tau^2$
121	I200002	$-2X^2\tau P_\tau + 2X P_x\tau^2 - 2X P_x P_\tau^2 + 2P_x^2\tau P_\tau$
122	R002020	$Y^2\tau^2 - Y^2 P_\tau^2 - 4Y P_y\tau P_\tau - P_y^2\tau^2 + P_y^2 P_\tau^2$
123	I002020	$2Y^2\tau P_\tau + 2Y P_y\tau^2 - 2Y P_y P_\tau^2 - 2P_y^2\tau P_\tau$
124	R002002	$Y^2\tau^2 - Y^2 P_\tau^2 + 4Y P_y\tau P_\tau - P_y^2\tau^2 + P_y^2 P_\tau^2$
125	I002002	$-2Y^2\tau P_\tau + 2Y P_y\tau^2 - 2Y P_y P_\tau^2 + 2P_y^2\tau P_\tau$
126	R101011	$XY\tau^2 + XY P_\tau^2 - P_x P_y\tau^2 - P_x P_y P_\tau^2$
127	I101011	$X P_y\tau^2 + X P_y P_\tau^2 + P_x Y\tau^2 + P_x Y P_\tau^2$
128	R100111	$XY\tau^2 + XY P_\tau^2 + P_x P_y\tau^2 + P_x P_y P_\tau^2$
129	I100111	$-X P_y\tau^2 - X P_y P_\tau^2 + P_x Y\tau^2 + P_x Y P_\tau^2$
130	R101020	$XY\tau^2 - XY P_\tau^2 - 2X P_y\tau P_\tau - 2P_x Y\tau P_\tau - P_x P_y\tau^2 + P_x P_y P_\tau^2$
131	I101020	$2XY\tau P_\tau + X P_y\tau^2 - X P_y P_\tau^2 + P_x Y\tau^2 - P_x Y P_\tau^2 - 2P_x P_y\tau P_\tau$
132	R101002	$XY\tau^2 - XY P_\tau^2 + 2X P_y\tau P_\tau + 2P_x Y\tau P_\tau - P_x P_y\tau^2 + P_x P_y P_\tau^2$
133	I101002	$-2XY\tau P_\tau + X P_y\tau^2 - X P_y P_\tau^2 + P_x Y\tau^2 - P_x Y P_\tau^2 + 2P_x P_y\tau P_\tau$
134	R100120	$XY\tau^2 - XY P_\tau^2 + 2X P_y\tau P_\tau - 2P_x Y\tau P_\tau + P_x P_y\tau^2 - P_x P_y P_\tau^2$
135	I100120	$2XY\tau P_\tau - X P_y\tau^2 + X P_y P_\tau^2 + P_x Y\tau^2 - P_x Y P_\tau^2 + 2P_x P_y\tau P_\tau$
136	R100102	$XY\tau^2 - XY P_\tau^2 - 2X P_y\tau P_\tau + 2P_x Y\tau P_\tau + P_x P_y\tau^2 - P_x P_y P_\tau^2$
137	I100102	$-2XY\tau P_\tau - X P_y\tau^2 + X P_y P_\tau^2 + P_x Y\tau^2 - P_x Y P_\tau^2 - 2P_x P_y\tau P_\tau$
138	R210010	$X^3\tau - X^2 P_x P_\tau + X P_x^2\tau - P_x^3 P_\tau$
139	I210010	$X^3 P_\tau + X^2 P_x\tau + X P_x^2 P_\tau + P_x^3\tau$
140	R210001	$X^3\tau + X^2 P_x P_\tau + X P_x^2\tau + P_x^3 P_\tau$

DR Index	Designation	Element
141	I210001	$-X^3P_\tau + X^2P_x\tau - XP_x^2P_\tau + P_x^3\tau$
142	R002110	$Y^3\tau - Y^2P_yP_\tau + YP_y^2\tau - P_y^3P_\tau$
143	I002110	$Y^3P_\tau + Y^2P_y\tau + YP_y^2P_\tau + P_y^3\tau$
144	R002101	$Y^3\tau + Y^2P_yP_\tau + YP_y^2\tau + P_y^3P_\tau$
145	I002101	$-Y^3P_\tau + Y^2P_y\tau - YP_y^2P_\tau + P_y^3\tau$
146	R101110	$XY^2\tau + XP_y^2\tau - P_xY^2P_\tau - P_xP_y^2P_\tau$
147	I101110	$XY^2P_\tau + XP_y^2P_\tau + P_xY^2\tau + P_xP_y^2\tau$
148	R101101	$XY^2\tau + XP_y^2\tau + P_xY^2P_\tau + P_xP_y^2P_\tau$
149	I101101	$-XY^2P_\tau - XP_y^2P_\tau + P_xY^2\tau + P_xP_y^2\tau$
150	R111010	$X^2Y\tau - X^2P_yP_\tau + P_x^2Y\tau - P_x^2P_yP_\tau$
151	I111010	$X^2YP_\tau + X^2P_y\tau + P_x^2YP_\tau + P_x^2P_y\tau$
152	R111001	$X^2Y\tau + X^2P_yP_\tau + P_x^2Y\tau + P_x^2P_yP_\tau$
153	I111001	$-X^2YP_\tau + X^2P_y\tau - P_x^2YP_\tau + P_x^2P_y\tau$
154	R300010	$X^3\tau - 3X^2P_xP_\tau - 3XP_x^2\tau + P_x^3P_\tau$
155	I300010	$X^3P_\tau + 3X^2P_x\tau - 3XP_x^2P_\tau - P_x^3\tau$
156	R300001	$X^3\tau + 3X^2P_xP_\tau - 3XP_x^2\tau - P_x^3P_\tau$
157	I300001	$-X^3P_\tau + 3X^2P_x\tau + 3XP_x^2P_\tau - P_x^3\tau$
158	R003010	$Y^3\tau - 3Y^2P_yP_\tau - 3YP_y^2\tau + P_y^3P_\tau$
159	I003010	$Y^3P_\tau + 3Y^2P_y\tau - 3YP_y^2P_\tau - P_y^3\tau$
160	R003001	$Y^3\tau + 3Y^2P_yP_\tau - 3YP_y^2\tau - P_y^3P_\tau$
161	I003001	$-Y^3P_\tau + 3Y^2P_y\tau + 3YP_y^2P_\tau - P_y^3\tau$
162	R201010	$X^2Y\tau - X^2P_yP_\tau - 2XP_xYP_\tau - 2XP_xP_y\tau - P_x^2Y\tau + P_x^2P_yP_\tau$
163	I201010	$X^2YP_\tau + X^2P_y\tau + 2XP_xY\tau - 2XP_xP_yP_\tau - P_x^2YP_\tau - P_x^2P_y\tau$
164	R201001	$X^2Y\tau + X^2P_yP_\tau + 2XP_xYP_\tau - 2XP_xP_y\tau - P_x^2Y\tau - P_x^2P_yP_\tau$
165	I201001	$-X^2YP_\tau + X^2P_y\tau + 2XP_xY\tau + 2XP_xP_yP_\tau + P_x^2YP_\tau - P_x^2P_y\tau$
166	R102010	$XY^2\tau - 2XYP_yP_\tau - XP_y^2\tau - P_xY^2P_\tau - 2P_xYP_y\tau + P_xP_y^2P_\tau$
167	I102010	$XY^2P_\tau + 2XYP_y\tau - XP_y^2P_\tau + P_xY^2\tau - 2P_xYP_yP_\tau - P_xP_y^2\tau$
168	R102001	$XY^2\tau + 2XYP_yP_\tau - XP_y^2\tau + P_xY^2P_\tau - 2P_xYP_y\tau - P_xP_y^2P_\tau$
169	I102001	$-XY^2P_\tau + 2XYP_y\tau + XP_y^2P_\tau + P_xY^2\tau + 2P_xYP_yP_\tau - P_xP_y^2\tau$
170	R200110	$X^2Y\tau + X^2P_yP_\tau - 2XP_xYP_\tau + 2XP_xP_y\tau - P_x^2Y\tau - P_x^2P_yP_\tau$
171	I200110	$X^2YP_\tau - X^2P_y\tau + 2XP_xY\tau + 2XP_xP_yP_\tau - P_x^2YP_\tau + P_x^2P_y\tau$
172	R200101	$X^2Y\tau - X^2P_yP_\tau + 2XP_xYP_\tau + 2XP_xP_y\tau - P_x^2Y\tau + P_x^2P_yP_\tau$
173	I200101	$-X^2YP_\tau - X^2P_y\tau + 2XP_xY\tau - 2XP_xP_yP_\tau + P_x^2YP_\tau + P_x^2P_y\tau$
174	R100201	$XY^2\tau - 2XYP_yP_\tau - XP_y^2\tau + P_xY^2P_\tau + 2P_xYP_y\tau - P_xP_y^2P_\tau$
175	I100201	$-XY^2P_\tau - 2XYP_y\tau + XP_y^2P_\tau + P_xY^2\tau - 2P_xYP_yP_\tau - P_xP_y^2\tau$

DR Index	Designation	Element
176	R100210	$XY^2\tau + 2XYP_yP_\tau - XP_y^2\tau - P_xY^2P_\tau + 2P_xYP_y\tau + P_xP_y^2P_\tau$
177	I100210	$XY^2P_\tau - 2XYP_y\tau - XP_y^2P_\tau + P_xY^2\tau + 2P_xYP_yP_\tau - P_xP_y^2\tau$
178	R310000	$X^4 - P_x^4$
179	I310000	$2X^3P_x + 2XP_x^3$
180	R003100	$Y^4 - P_y^4$
181	I003100	$2Y^3P_y + 2YP_y^3$
182	R201100	$X^2Y^2 + X^2P_y^2 - P_x^2Y^2 - P_x^2P_y^2$
183	I201100	$2XP_xY^2 + 2XP_xP_y^2$
184	R112000	$X^2Y^2 - X^2P_y^2 + P_x^2Y^2 - P_x^2P_y^2$
185	I112000	$2X^2YP_y + 2P_x^2YP_y$
186	R211000	$X^3Y - X^2P_xP_y + XP_x^2Y - P_x^3P_y$
187	I211000	$X^3P_y + X^2P_xY + XP_x^2P_y + P_x^3Y$
188	R102100	$XY^3 + XYP_y^2 - P_xY^2P_y - P_xP_y^3$
189	I102100	$XY^2P_y + XP_y^3 + P_xY^3 + P_xYP_y^2$
190	R210100	$X^3Y + X^2P_xP_y + XP_x^2Y + P_x^3P_y$
191	I210100	$-X^3P_y + X^2P_xY - XP_x^2P_y + P_x^3Y$
192	R101200	$XY^3 + XYP_y^2 + P_xY^2P_y + P_xP_y^3$
193	I101200	$-XY^2P_y - XP_y^3 + P_xY^3 + P_xYP_y^2$
194	R400000	$X^4 - 6X^2P_x^2 + P_x^4$
195	I400000	$4X^3P_x - 4XP_x^3$
196	R004000	$Y^4 - 6Y^2P_y^2 + P_y^4$
197	I004000	$4Y^3P_y - 4YP_y^3$
198	R301000	$X^3Y - 3X^2P_xP_y - 3XP_x^2Y + P_x^3P_y$
199	I301000	$X^3P_y + 3X^2P_xY - 3XP_x^2P_y - P_x^3Y$
200	R103000	$XY^3 - 3XYP_y^2 - 3P_xY^2P_y + P_xP_y^3$
201	I103000	$3XY^2P_y - XP_y^3 + P_xY^3 - 3P_xYP_y^2$
202	R300100	$X^3Y + 3X^2P_xP_y - 3XP_x^2Y - P_x^3P_y$
203	I300100	$-X^3P_y + 3X^2P_xY + 3XP_x^2P_y - P_x^3Y$
204	R100300	$XY^3 - 3XYP_y^2 + 3P_xY^2P_y - P_xP_y^3$
205	I100300	$-3XY^2P_y + XP_y^3 + P_xY^3 - 3P_xYP_y^2$
206	R202000	$X^2Y^2 - X^2P_y^2 - 4XP_xYP_y - P_x^2Y^2 + P_x^2P_y^2$
207	I202000	$2X^2YP_y + 2XP_xY^2 - 2XP_xP_y^2 - 2P_x^2YP_y$
208	R200200	$X^2Y^2 - X^2P_y^2 + 4XP_xYP_y - P_x^2Y^2 + P_x^2P_y^2$
209	I200200	$-2X^2YP_y + 2XP_xY^2 - 2XP_xP_y^2 + 2P_x^2YP_y$

Chapter 15

Index

Index

aberrations, 15, 250, 590, 642
achromat, 678, 688
add, 460
advanced commands, 349
aim, 479, 526
amplitude, 365, 382
analysis subroutines, 91
anharmonicity, 365, 382
aperture, 310–312, 343
apply map, 450
apply script N, 481
arc, 281
array, 347, 517, 527
augmented phase space, 80
axial rotation, 218

bases, 403
basis elements, 881
beam, 98, 108, 109, 488
beam betatron functions, 494
beam moment plots, 800
beam moment plots with geometry, 838
beam parameters, 336
beam-line elements, 139
bell, 325
bending magnet, 143, 159, 163, 171, 182
beta, 94, 109, 337, 353
betatron, 355
betatron amplitude, 365, 382
betatron factor, 368
betatron motion, 222
brho, 109, 337
buffers, 447
buncher, 49

canonical, 94
canonical transformation, 81
capture parameter set, 551
cartesian basis, 403, 881
catalog, 139, 283
cavity, 49
cell, 67
change tune range, 480
chaos, 761
charge, 109, 337
check symplectic condition, 484
chromatic expansion, 467
chromaticity, 365, 564
circulate, 291
circumference, 352
close files, 316
closed orbit, 351
closed-orbit transfer matrix, 354
combined function bend, 182
combined function quadrupole, 251
commands, 283, 349
comment, 98, 105
comment out, 106
complete achromat, 688
composite plots, 815
compressed multipole, 230
concatenation, 89, 131
constants, 93, 854
constraints, 544
contents of arrays, 347
copy file to working array, 517
corrector, 590, 642
COSY Infinity, 854
Courant-Snyder invariants, 365, 383, 424, 430

Darboux, 306
data point, 256, 508
defaults, 557
delta, 94
design energy, 94, 109, 337
design momentum, 94, 109, 337
design orbit, 77, 94

design velocity, 344
determinant, 333
deviation variables, 94
dimensionless variables, 94
dimensions, 340
dipole, 143, 159, 163, 171, 182
dispersion function, 352
dispersion matrix, 240
dispersion polynomial, 335
distortion, 15
distribution, 488
do loop, 522
doublet, 12, 263
drawings, 850
drift, 142
drop file, 324
dynamic aperture, 728
dynamic map, 381
dynamic resonance basis, 403, 882

eigen emittances, 496
eigen moments, 496
eigenvalues, 880
eigenvectors, 367, 383
electron microscope, 642
element by element, 292, 591, 604, 625
element library, 87, 139
emittance, 496
end, 286
energy deviation, 94
envelope coefficients, 367, 383
envelope matrix, 495
envelopes, 367, 383
error messages, 880
Euler angles, 508
evaluate a polynomial, 465
exponential, 404
exponents, 881
extra array, 347, 527

factorization theorem, 84
field index, 187
field profile, 245, 260
file 11, 97
files, 95, 316, 317, 517, 518, 878, 879
filter, 314
final conditions, 99, 101
final focus test beam facility, 704
fit, 92, 526, 532, 535, 564, 590, 605, 642
fitting at multiple locations, 667
fitting commands, 520
flags, 557
flat file, 327
floor plan, 850
foot print, 584
format, 96, 878, 879
Fourier analyze, 448, 449
fractional part of tune, 364, 382
free format, 96
free parameter set, 556
fringe field, 179, 250, 259
full tune, 574
function, 450
fundamental constants, 93, 854

gamma, 93, 109, 337, 353
general optimization, 540
generate beam, 488
generator, 82
GENMAP, 3, 242, 259
geometric coordinates, 355
geometry, 340, 815, 828, 838
geometry of a loop, 508
get buffer contents, 447
get transfer map, 309
gradient matrix, 546

Hamilton's Equations, 76
Hamiltonian flow, 80
history, 313
history of lost particle, 761

identity map, 300
imaging system, 12, 245
index, 86, 100, 187, 881
infinities, 332
initial conditions, 99, 491
initial conditions buffer, 288
inner procedure, 522, 524
input, 100, 878

input transfer map, 294
interspersed comments, 106
invariants, 423, 429, 496
invert, 301
isochronous, 688

J matrix, 81, 234

KAM tori, 58
kicks, 728

labor, 98, 131
lattice function plots, 773, 815
layout drawings, 850
length, 344
library, 87
Lie generator, 82
Lie operator, 82
Lie transformation, 83
linear part, 84
lines, 98, 119
logarithm of normal form, 519
logical do loop, 522
loops, 98, 127, 327, 508
lost particles, 313
lumps, 98, 122, 728

MAD, 3, 854
map input, 100
map output, 101
marker, 255
MaryLie index, 881
mask, 305
mass, 93, 109, 337
master input file, 7, 96, 97, 102, 293
matched beam, 705, 729
matrix, 297
matrix norm, 487
menu, 98, 110, 559
merge files, 518
merit function, 326, 542, 543
messages, 880
microprobe, 5, 590
microscope, 642
minimize, 538
mixed invariants, 423, 429
mnemonic, 110, 858
moment, 450, 475, 800, 838
moment analysis, 493
moment matrix, 494
momentum compaction, 354
momentum deviation, 94
monomial index, 86
MSS optimizer, 538, 625
multiple locations, 667
multiplet, 257
multiplicative notation, 121, 126, 129, 134
multiply, 461
multipole, 226, 230, 251, 257

name, 118
nested procedures, 523
Newton's method, 290, 536
nonlinear instability, 42
nonlinear part, 84, 289, 483
norm, 487
normal form, 356, 368, 405, 413, 519
normal form factorization, 382
normalizing map, 356, 368, 384, 405, 413
number, 321

octupole, 208
octupole correction, 590, 605
open files, 317
operator, 82
optimization commands, 520
optimize, 92, 526, 532, 538, 540, 605, 625
optimizing at multiple locations, 667
orbit, 77, 351
outer procedure, 523, 525
output, 101, 879
output transfer map, 296

P_τ, 94
parameter set, 319, 320, 322, 551, 556
particle loss, 761
path length, 344
pause, 331
phase slip, 352
phase space, 79
phase trombone, 66

phase-space distribution, 488
phase-space variables, 94
plane rotation, 169
plots, 773, 790, 800, 815, 828, 838
Poincare surface, 352
Poisson bracket, 79, 462
Poisson matrix, 81, 234
polar decomposition, 463
pole face rotation, 169
polynomial, 459–462, 465
polynomial labeling, 85
polynomial scalar product, 485
POSTER, 3, 340, 815, 830, 840
power of dynamic normal form, 445
power of nonlinear part, 483
power of static normal form, 443
precision, 289
PREP, 3, 96, 102
principal planes, 492
print master input file, 293
print transfer map, 297
procedure, 520, 522–525
profile, 245, 260
pseudo hamiltonian, 405, 413

quadratic moments, 475
quadrupole, 194, 251, 257
quantities, 476, 478
quick reference guide, 858

R matrix, 82, 84, 297
random elements, 235
random map, 278
random number, 278, 490
random parameter set, 320
ray plots, 790
ray plots with geometry, 828
ray trace, 90, 99, 287
REC quadrupole, 257
references, 854
reset menu entries, 559, 643
resonance analysis, 390, 396
resonance bases, 403, 881
reverse, 303
reverse factorize, 304
rewind, 295
RF cavity, 49, 217
rigidity, 109
ring, 564
rms merit function, 542
rotation, 169, 218

scalar multiplication, 459
scalar product, 485
scale length, 94, 109, 337
scan, 532, 662
scan parameter space, 550
select quantities, 476, 478
septum, 310
sextupole, 200
sextupole correction, 642
simple commands, 283
singlet, 260
skew, 218
slicing a lattice or ring, 574
slicing elements, 773
solenoid, 241, 642
space, 276
spectrometer, 19
spherical aberration, 15, 590, 642
splitting elements, 773
spot forming system, 5, 590, 605, 642, 704, 790, 800, 828
square, 307
Stanford Linear Accelerator Center, 704
static, 352
static map, 363
static resonance basis, 403, 881
storage ring, 29, 49, 66, 564, 728
store transfer map, 308
suggestion form, 857
sum of squares merit function, 542
sum of squares optimization, 538
Superconducting Super Collider, 66
symplectic condition, 484
symplectic map, 81
symplectic matrix, 81
symplectic ray trace, 90, 289
symplectically congruent, 496

symplectify a matrix, 306
synchrotron tune, 382

T matrix, 82, 84, 297
target values, 526
tau, 94
Taylor expansion, 82
temporal dispersion, 352
temporal tune, 382, 480
Tevatron, 729
thin multipole, 226
time, 323
time of flight, 94, 344, 352
torus, 58
total transfer map, 131
tracking, 99
transfer map, 81, 131, 294, 296, 297
transfer matrix, 82
transfer quadratic moments, 475
transformation function, 289
transition energy, 217
transition gamma, 354
translate initial conditions, 491
TRANSPORT, 95, 854
transpose, 302
triplet, 5, 263
tune, 364, 382, 480, 564, 574
tune expansions, 364
tune foot print, 584
tune range, 480
Twiss analyze, 363, 381
Twiss invariants, 365, 383
Twiss matrix, 222
Twiss parameter expansions, 366
Twiss parameters, 365, 382
Twiss polynomial, 334
type code, 110, 858

U matrix, 82, 84, 297
ucalc, 341, 479
units, 93
user calculated array, 341
user specified subroutine, 238, 239
user written constraints, 544
user written merit function, 543
variables, 93
vary, 479, 532
velocity, 344, 353

warning messages, 880
wiggler, 163
window, 312, 343
working array, 517
write contents of loop, 327
write out parameter values, 322
write selected quantities, 478

zblock z(i,j), 288, 527, 705
zeroes, 333