Lie Algbraic Methods for Treating

Lattice Parameter Errors in Particle Accelerators

by

Liam Michael Healy

Dissertation submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1986

APPROVAL SHEET

Title of Dissertation: Lie Algebraic Methods for Treating Lattice Parameter Errors in Particle Accelerators

Name of Candidate: Liam Michael Healy
Doctor of Philosophy, 1986

Dissertation and Abstract Approved: Alex J. Dragt
Alex J. Dragt
Professor
Department of Physics and Astronomy

Date Approved: 8/16/86

ABSTRACT

Title of Dissertation: Lie Algebraic Methods for Treating Lattice Parameter Errors in Particle Accelerators

Liam Michael Healy, Doctor of Philosophy, 1986

Dissertation Directed By: Dr. Alex J. Dragt, Professor
Department of Physics and Astronomy
University of Maryland

Orbital dynamics in particle accelerators, and ray tracing in light optics, are examples of Hamiltonian systems. The transformation from initial to final phase space coordinates in such systems is a symplectic map. Lie algebraic techniques have been used with great success in the case of idealized systems to represent symplectic maps by Lie transformations. These techniques allow rapid computation in tracking particles while maintaining complete symplecticity, and easy extraction of analytical quantities such as chromaticities and aberrations.

Real accelerators differ from ideal ones in a number of ways. Magnetic or electric devices, designed to guide and focus the beam, may be in the wrong place or have the wrong orientation, and they may not have the intended field strengths. The purpose of this dissertation is to extend the Lie algebraic techniques to treat these misplacement, misalignment and mispowering errors.

Symplectic maps describing accelerators with errors typically have first-order terms. There are two major aspects to creating a Lie algebraic theory of accelerator errors: creation of appropriate maps and their subsequent manipulation and use.

There are several aspects to the manipulation and use of symplectic maps. A first aspect is particle tracking. That is, one must find how particle positions are transformed by a map. A second is concatenation, the combining of several maps into a single map including nonlinear feed-down effects from high-order elements. A third aspect is the computation of the fixed point of a map, and the expansion of a map about its fixed point. For the case of a map representing a full turn in a circular accelerator, the fixed point corresponds to the closed orbit.

The creation of a map for an element with errors requires the integration of a Hamiltonian with first-order terms to obtain the corresponding Lie transformation. It also involves a procedure for the complete specification of errors, and the generation of the map for an element with errors from the map of an ideal element.

The methods described are expected to be applicable to other electromagnetic systems such as electron microscopes, and also to light optics systems.

Acknowledgments

There are many people who made this effort not only possible but enjoyable as well. First, of course, is my advisor Alex Dragt, whose understanding of physics and patience with me made the whole process very rewarding. In addition, the people with whom I've worked at Maryland have been invaluable in gaining an understanding of Lie algebraic methods and of accelerator physics: David Douglas, who got me interested in the subject in the first place, and whose name rightfully appears in many places in this dissertation; Etienne Forest, who was of great assistance in the difficult early stage of learning the subject and who saved me from much embarassment by pointing out the gaps in my half-baked understanding of many topics; Filippo Neri and Robert Ryne with whom discussions on these methods and other topics were very helpful. Lastly, Rachel Needle typed this whole manuscript from illegible handwriting, with good-natured efficiency, and kept me on schedule in the crucial final months.

Although a dissertation is about one very narrow topic within a subfield, graduate education involves a wide range of physics and how to go about doing it. In this regard, I have learned far more from my fellow graduate students than all the courses and books that were part of graduate school. It is impossible to name all these many friends but I would in particular like to thank Fernando Pineda and Parney Albright not only contributing to my knowledge but also for being good friends and providing necessary diversions.

I would remiss if I did not acknowledge the finanical support in this endeavor. A research assistantship under the U.S. Department of

Energy contract DE-AS05-80ER-10666 provided much of this support, with a Graduate School Fellowship providing the balance. In addition, two summers at the Los Alamos National Laboratory were not only of financial benefit but more importantly gave me another view of accelerator physics and interaction with knowledgeable people. For this I would like to thank Richard Cooper and the others at Los Alamos.

Pat Francis deserves much gratitude for companionship, moral support and many hours of proofreading. She knows more about accelerator physics and Lie algebras than she cares to!

Finally, I would like to thank the people that started it all, my parents Edward and Helen Healy who instilled in me a love of learning and the dedication to complete a task. The other member of my family, my sister Beth, taught me how to read, a skill which turned out to be crucial in doing this dissertation.

Table of Contents

List of Tables

List of Figures

Introduction

The work presented here is part of an ongoing effort in the application of Lie algebraic techniques to particle accelerators and related areas such as light optics (Dragt and Finn [1976], Dragt [1982], Douglas [1982], Dragt and Forest [1983], Forest [1984]). In particular, I treat the problem of lattice parameter errors, especially beam element alignment, positioning and powering errors.

Such errors will generally introduce a first-order term into the factorized Lie transformation, i.e., a particle on the design trajectory, once it passes through one of these erroneous beamline elements, will no longer be on the design trajectory. There thus need to be the mathematical tools available to work with these maps: concatenation, tracking and finding the fixed point (closed orbit) in particular. Part I covers these mathematical tools.

Part II then treats a problem which perhaps conceptually comes before Part I: how the maps of erroneous elements are computed in the first place. What is described here is an extension of the methods developed previously in the references given above for ideal elements, together with some computation of actual elements.

Finally, the appendices cover various topics of related interest: the beginnings of a method for treating random distributions of errors; description, examples, testing and listing of MARYLIE 3.1, the computer code that embodies the work here by extending the Lie algebraic particle tracking code MARYLIE 3.0 (Dragt et. al. [1985]) to include errors; a description of ANNALIE, the code written in the language SMP to assist with the analytical computations needed to write MARYLIE.

Part I: General Lie Algebraic and Group Theoretical Tools

This part deals with the mathematics necessary to treat beamline element errors, which produce first-order terms in the factored Lie transformations. It is an extension of the methods without first-order terms developed by Dragt and Finn [1976], Dragt [1982], and Douglas [1982]. Chapter 1 is an introduction to the mathematics, showing how Lie algebras play a role in Hamiltonian systems. Much of the information comes from the references above, but is repeated for the sake of completeness. Chapter 2 deals with the tracking of particles through the maps, and how a first-order term affects this process. Chapter 3 describes, in mathematical terms, the various Lie algebras implied by possible approximation schemes, and shows a particular one as natural for concatenation. Chapter 4 is a computation of the concatenation rules. Chapter 5 shows how to handle the symplectification of matrices, necessary when a first-order term is concatenated with higher order terms. Chapter 6 is a description of a method for finding the fixed point, or closed orbit, of a map, and the map around it. This is extremely important because in the presence of machine errors, one almost always wants to find the new fixed point and the map around it. Finally, Chapter 7 deals with the Euclidean group, the group of rigid body motions, which we shall need for description of element alignment errors.

1. Introduction

a. The Motion of Charged Particles in Accelerators

A charged particle moving in an accelerator is subject to electromagnetic forces of various origins. In most cases, the predominant force is from the magnets, radio frequency cavities, and perhaps electrostatic elements installed as part of the accelerator to guide and accelerate the beam. Other possible sources include space-charge forces, that is, the force of other charged particles in the bunch, and wake-field forces, the electromagnetic force reflected off the walls of a cavity from the earlier passage of particles. In addition, synchrotron radiation plays a significiant role in some machines, and minor effects may be caused by collision of the accelerated particles with residual gas in the beam pipe.

A major task of accelerator physics is to simulate the motion of particles in accelerators to insure proper behavior and to understand what magnet arrangements and strengths - the lattice - will produce desirable behavior, and what arrangments will produce undesirable behavior. If we are lucky, we may find some entity that represents the lattice, and from which we may extract the potential for good or ill behavior directly, or use this entity for simulation of particle motion.

For simplicity, I shall assume the only significant effects in beam motion arise from the external magnetic or electric forces, that is, those forces coming from fixed elements such as bending magnets, focusing magnets (quadrupoles), and so on. We then choose a set of coordinates: the z direction will be along the direction of a particle

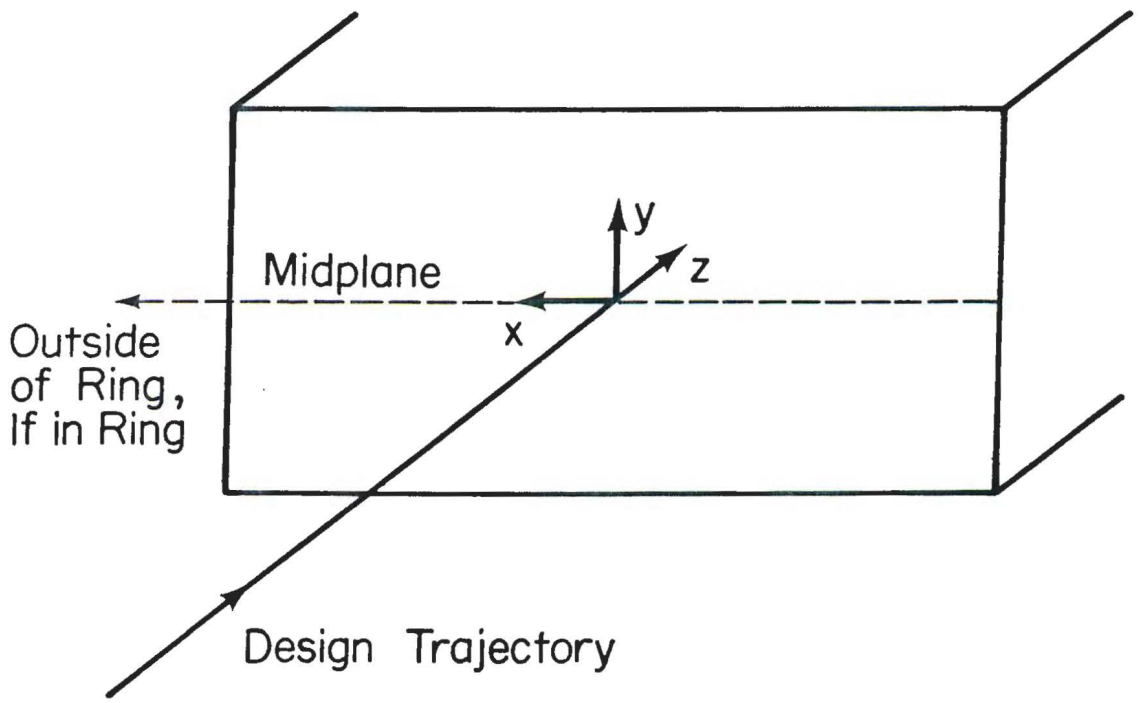


Figure 1.1 Coordinates

following the design trajectory ("design particle"), the x direction will be in the midplane of the bending magnets, or horizontal and perpendicular to z, y will be perpendicular to both, and t will be flight time (see Figure 1.1). Both x and y are measured from the design trajectory. Each of these has a conjugate momentum p_x, p_y, p_z, or p_t. Six of these quantities (three pairs) form the phase space, and the other pair become the independent variable and the negative of the Hamiltonian. Given a particle's initial position in phase space,

$$v^o \equiv (x,p_x,y,p_y,z,p_z)\,|_{t=t_o}, \tag{1.1}$$

one way to analyze particle dynamics is to study six dependent variables, denoted by ζ, as a function of time, say

$$\zeta(v^o,t) \equiv x(v^o,t),\ p_x(v^o,t),\ \ldots\ p_z(v^o,t). \tag{1.2}$$

A simple representation of particle behavior would be to give these functions for the whole time a particle would be in a machine. Certainly, undesirable properties would become obvious - if a beam were doomed to head for a wall, this would be indicated in the function. Unfortunately, these functions are difficult to calculate in general, due to non-linearities that arise from the kinematics and from the lattice elements, and we would be mathematically unable to make use of an obvious property of circular machines: as far as the forces are concerned, one turn is like another.

We shall make an important change that remedies this problem: all phase space variables will be measured as deviations from the design

values. This allows us to approximate the motion by Taylor expansion of the functions:

$$\zeta_i(v_o,t) = \sum_j m_{ij}(t)\, v_j^o + \sum_{j,k} t_{ijk}(t)\, v_j^o v_k^o + \ldots \qquad (1.3)$$

where v^o are the coordinates at $t = 0$ and m_{ij}, t_{ijk}, ... are real coefficients. For convenience, we may take a fixed section of the accelerator, say one turn, as implicit, and drop the t. This corresponds to a Poincare surface of section; we give the coordinates of a particular particle only at a particular position on the ring at each pass and do not care what happens to it elsewhere. These functions ζ_i together form what is called a **transfer map** and shall usually be represented with script letters $\mathcal{M},\mathcal{N}$ etc.

Generally, this is an effective method, because most accelerators are quite linear. That is, each term in the expansion is much larger than the next, so that truncation of the series after two or three turns gives reasonable answers.

The quantities $M \equiv \{m_{ij}\}$, $T \equiv \{t_{ijk}\}$, ... are determined solely by the machine construction, and not at all by the initial (or any) conditions of the particle. This is the representation of the lattice we sought. We may track particles through a lattice by repeatedly applying (1.2), or we may extract useful information directly from the coefficients m_{ij}, t_{ijk}, ... Further, we may concatenate: determine the matrices M, T, ... for a section from two pieces M_1,T_1, and M_2,T_2, ... that make it up, e.g., obtaining the lattice matrices for two turns from those for one. If the nonlinearities are not too great, the terms that have been eliminated in truncation will not be significant.

So far, I have assumed that all elements are perfectly positioned and powered: there are no constant terms in the expansion (1.3), so a design particle, with coordinates (0,0,0,0,0,0), maintains those coordinates. This need not be the case, of course; magnets, as well as particles, may fail to be in the design position.

The introduction of constant terms represents no major problem until we try to combine maps. Suppose we have two maps, from time t_o to t_1 and from t_1 to t_2.

$$v^1 = M^{t_1 \leftarrow t_o}(v^o) \tag{1.4a}$$

and

$$v^2 = M^{t_2 \leftarrow t_1}(v^2) \tag{1.4b}$$

and we wish to combine these into a single map

$$v^2 = M^{t_2 \leftarrow t_o}(v^o). \tag{1.5}$$

Then, with the truncation of the Taylor series at each step, we may introduce "feed down" errors: a particular map's fourth-order term, when concatenated with a first-order term, generates a third-, second- and first-order term. If the fourth-order term had been neglected, the resultant third-order term would be wrong. The solution to this, which I shall discuss in greater detail later, is to assume that the constant terms are small in the sense that the phase space coordinates are small, and may be similarly truncated.

b. Hamiltonian Systems and Lie Groups

i. Hamilton's Equations

The motion of a charged particle in an accelerator, assuming no synchrotron radiation effects, is a Hamiltonian system. The description is given with a set of 2n coordinates ζ, which form n groups of canonical pairs, and there is a function $H(\zeta,t)$, the Hamiltonian, such that Hamilton's equations hold,

$$\dot{\zeta} = J \cdot \nabla_{\zeta} H. \tag{1.6}$$

Here the matrix J and the generalized gradient are defined by

$$J = \begin{bmatrix} B & 0 & & \\ 0 & B & & \\ & & \ddots & \\ & & & \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{1.7}$$

and

$$\nabla_{\zeta} = \begin{bmatrix} \frac{\partial}{\partial \zeta_1} \\ \vdots \\ \frac{\partial}{\partial \zeta_{2n}} \end{bmatrix} . \tag{1.8}$$

On the face of it, there is nothing special about a Hamiltonian system as opposed to a non-Hamiltonian system. However, the motion possible from a Hamiltonian system, a Hamiltonian flow, is more restricted than an arbitrary flow. In essence, some of the coefficients M, T, ... of Section 1a are redundant. It will be possible to recast the description of an accelerator section with fewer numbers. This is

done by means of a Lie transformation, a method describing a symplectic map. All Hamiltonian flows give rise to symplectic transformations on phase space, as I shall show later.

ii. Groups

To study Hamiltonian systems, we shall need the concept of a group. A **group** is a set, together with a "multiplication" operation which will be denoted by juxtaposition, satisfying the following four axioms:

1) Closure: if A and B are in the group, so is AB;
2) Identity: there is an identity element I such that $AI = IA = A$;
3) Inversion: for every element A there is an inverse A^{-1} such that $AA^{-1} = A^{-1}A = I$;
4) Associativity: group multiplication obeys the relation $A(BC) = (AB)C$.

All the groups that will be introduced are **Lie groups.** Lie groups are groups that are also manifolds (have a differentiable structure) such that the group operation and inversion are both C^{∞} (infinitely differentiable).

A mapping $\rho: G \to H$ from a group G to a group H is a group **homomorphism** if the group operation is preserved,

$$\rho(g_1)\,\rho(g_2) = \rho(g_1 g_2). \tag{1.9}$$

If ρ is injective (one-to-one) and surjective (onto), i.e., is a one-to-one correspondence, then it is an **isomorphism** and the groups are

isomorphic.

Groups of transformations frequently appear in physics and are of particular importance to Hamiltonian dynamics. For example, the set of all possible rotations of a rigid body in space (or transformation of the coordinate axes) forms a group. A particular kind of transformation, the **linear transformation**, acts on a vector space V. A transformation $T : V \to V$ is linear if

$$T(\alpha_1 v_1 + \alpha_2 v_2) = \alpha_1 T(v_1) + \alpha_2 T(v_2) \qquad (1.10)$$

where $v_1, v_2 \in V$ and $\alpha_1, \alpha_2 \in \mathbf{R}$. If a set of basis vectors has been picked for V, there is a 1-1 correspondence between the set of linear transformations on V and the set of n by n matrices, where n is the dimension of V. For this reason, the distinction between these will be blurred.

A homomorphism from a group to a group of transformations is called a **realization** of the first group. We shall see that a particular realization is useful for computing the effects of misalignments (see Chapter 12). If the homomorphism maps to a group of linear transformations, the realization is called a **representation.** If a realization or a representation is an isomorphism it is called **faithful.**

A **subgroup** is a subset of a group that is also a group in itself, under the same operation. An **invariant** or **normal** subgroup H of G is one where for all $h \in H$, $g \in G$, $ghg^{-1} \in H$.

iii. The Symplectic Group and the Group of Symplectic Maps

A linear transformation or matrix M is **symplectic** if it satisfies

the property

$$M\tilde{J}\tilde{M} = J, \tag{1.11}$$

where $\tilde{M}$ denotes the transpose of M. These transformations or matrices form a group under matrix multipliation called the **symplectic group,** designated Sp(2n).

A map M from R^{2n} to R^{2n}, $\bar{\zeta} = M\zeta$, is symplectic if its Jacobian matrix defined by

$$M_{ij} = \frac{\partial \bar{\zeta}_i}{\partial \zeta_j} \tag{1.12}$$

is symplectic for all ζ. These maps form a group under composition, the **group of symplectic maps.** The symplectic group is a subgroup of it.

iv. The Poisson Bracket and Canonical Transformations

The Hamiltonian evolution

$$\dot{\zeta} = J \cdot \nabla_\zeta H \tag{1.13}$$

can be used to study an arbitrary function on phase space $f(\zeta,t)$. Then the time dependence is

$$\frac{df}{dt} = \nabla_\zeta f \, \dot{\zeta} + \frac{\partial f}{\partial t} . \tag{1.14}$$

Hamilton's equations allows us to substitute for $\dot{\zeta}$:

$$\frac{df}{dt} = \nabla_\zeta f \cdot J \cdot \nabla_\zeta H + \frac{\partial f}{\partial t} \tag{1.15}$$

It is useful to define the **Poisson Bracket** for two functions on phase space

$$[f,g]_\zeta = \nabla_\zeta f \cdot J \cdot \nabla_\zeta g \tag{1.16}$$

so that

$$\frac{df}{dt} = [f,H] + \frac{\partial f}{\partial t} \, . \tag{1.17}$$

If there is no explicit time dependence in the function f, then

$$\frac{df}{dt} = [f,H] = \nabla_\zeta f \cdot J \cdot \nabla_\zeta H. \tag{1.18}$$

Closely related to the concept of a Poisson Bracket is the important concept of a **canonical transformation.** A canonical transformation is a set of functions $\bar{\zeta}(\zeta,t)$ which preserve the Poisson Bracket:

$$[\bar{\zeta}_i,\bar{\zeta}_j]_\zeta = [\zeta_i,\zeta_j]_\zeta = J_{ij}. \tag{1.19}$$

I have used the subscript ζ to indicate the derivative ∇_ζ to be used, because we now have another set of coordinates $\bar{\zeta}$ that could also be used as canonical coordinates. In fact, if we invert the transformation $\zeta \to \bar{\zeta}$ locally around the image $\bar{\zeta}$ to form $\zeta(\bar{\zeta},t)$, we find

$$[\zeta_i,\zeta_j]_{\bar{\zeta}} = [\bar{\zeta}_i,\bar{\zeta}_j]_{\bar{\zeta}} = J_{ij}, \tag{1.20}$$

so it too is canonical.

Suppose we make two canonical transformations in succession $\zeta \to \bar{\zeta} \to \bar{\bar{\zeta}}$. Then

$$[\bar{\bar{\zeta}}_i, \bar{\bar{\zeta}}_j]_\zeta = \nabla_\zeta \bar{\bar{\zeta}}_i \cdot J \cdot \nabla_\zeta \bar{\bar{\zeta}}_j \tag{1.21}$$

$$= \sum_{m,n} \frac{\partial \bar{\bar{\zeta}}_i}{\partial \zeta_m} \cdot J_{mn} \cdot \frac{\partial \bar{\bar{\zeta}}_j}{\partial \zeta_n}$$

$$= \sum_{k,\ell,m,n} \frac{\partial \bar{\bar{\zeta}}_i}{\partial \bar{\zeta}_k} \frac{\partial \bar{\zeta}_k}{\partial \zeta_m} J_{mn} \frac{\partial \bar{\zeta}_\ell}{\partial \zeta_n} \frac{\partial \bar{\bar{\zeta}}_j}{\partial \bar{\zeta}_\ell}$$

because $[\bar{\zeta}_k, \bar{\zeta}_\ell]_\zeta = J_{k\ell}$, this is

$$= \frac{\partial \bar{\bar{\zeta}}_i}{\partial \bar{\zeta}_k} J_{k\ell} \frac{\partial \bar{\bar{\zeta}}_j}{\partial \bar{\zeta}_\ell} = [\bar{\zeta}_i, \bar{\zeta}_j]_{\bar{\zeta}} = J_{ij}. \tag{1.22}$$

So the composition of two canonical transformations $\zeta \to \bar{\bar{\zeta}}$ is canonical. Therefore, the composition of an arbitrary number of canonical transformations is canonical.

Finally, canonical transformations are associative because all transformations are. For these reasons, canonical transformations on phase space form a group under composition.

One may ask what the connection is between this group and the group of symplectic maps. Specifically, are they the same? The answer is yes, if canonical transformations are defined, as above, as those transformations that preserve the Poisson bracket. However, it is possible to define them as transformations that preserve the Hamiltonian. Then

the symplectic group is a subgroup of the canonical group. What is not included are the scaling transformations: if we allow transformations with Jacobian M such that

$$\tilde{M}JM = \lambda J \tag{1.23}$$

then this extended group is the canonical group.

Proof
$$[\bar{\zeta}_i, \bar{\zeta}_j] = \nabla\bar{\zeta}_i \cdot J \cdot \nabla\bar{\zeta}_j \tag{1.24}$$

$$= \sum_{k,\ell} M_{ik} J_{k\ell} M_{j\ell}$$

$$= (MJ\tilde{M})_{ij} = \lambda J_{ij}$$

because $\tilde{M}JM = \lambda J \Leftrightarrow MJ\tilde{M} = \lambda J$.

v. Hamiltonian Flows and Symplectic Maps

Now that we have the canonical group and the symplectic maps identified, we would like to relate Hamiltonian flows to them. Thus we have

Theorem: A Hamiltonian flow that takes the coordinates ζ^o at time 0 to ζ at time t gives rise to a symplectic transformation from ζ to ζ^o.

Proof: The Jacobian matrix M is defined by

$$M_{ij} = \frac{\partial\zeta_i}{\partial\zeta_j^o} . \tag{1.25}$$

Then

$$\dot{M}_{ij} = \frac{\partial}{\partial \zeta_j^o} \dot{\zeta}_i = \frac{\partial}{\partial \zeta_j^o} [\zeta_i, H]_\zeta \tag{1.26}$$

$$= \sum_k \frac{\partial}{\partial \zeta_j^o} \left(J_{ik} \frac{\partial H}{\partial \zeta_k} \right)$$

$$= \sum_k J_{ik} \frac{\partial}{\partial \zeta_j^o} \frac{\partial H}{\partial \zeta_k} = \sum_{k,\ell} J_{ik} \frac{\partial \zeta_\ell}{\partial \zeta_j^o} \frac{\partial^2 H}{\partial \zeta_\ell \partial \zeta_k}$$

$$= \sum_{k,\ell} J_{ik} S_{k\ell} M_{\ell j}$$

so

$$\dot{M} = JSM \tag{1.27}$$

where

$$S_{k\ell}(\zeta) = \frac{\partial^2 H}{\partial \zeta_\ell \partial \zeta_k} . \tag{1.28}$$

Now suppose t is divided up into N equal intervals of length ε. If $\bar{M}$ is the Jacobian matrix at the end and M is the Jacobian matrix at the beginning of one of these intervals

$$\bar{M} = M + \varepsilon \dot{M} + O(\varepsilon^2) = M + \varepsilon JSM + O(\varepsilon^2). \tag{1.29}$$

If we assume M is symplectic to order ε

$$MJ\tilde{M} = J + O(\varepsilon^2) \tag{1.30}$$

we may check the symplecticity of $\bar{M}$

$$\bar{M}J\tilde{\bar{M}} = (M + \varepsilon JSM + O(\varepsilon^2))\ J(M + \varepsilon JSM + O(\varepsilon^2)) \qquad (1.31)$$

$$= (1 + \varepsilon JS + O(\varepsilon^2))\ MJ\tilde{M}\ (1 + \varepsilon JS + O(\varepsilon^2))$$

$$= (1 + \varepsilon JS + O(\varepsilon^2))(J + O(\varepsilon^2))(1 - \varepsilon SJ + O(\varepsilon^2))$$

$$= J + \varepsilon JS - \varepsilon JS + O(\varepsilon^2) = J + O(\varepsilon^2).$$

At time 0, M is the identity, which is symplectic. Each of the N transformations is symplectic through order ε. By letting $\varepsilon \to 0$, we get exact symplecticity at the end; since $N = t/\varepsilon$, the remainder term $O(\varepsilon^2)$ goes to zero faster than the number of intervals increases.

c. Lie Algebras and Operators

We have seen the symplectic mapping that governs particle behavior in an accelerator can be viewed as an element of a Lie group. We now wish to look at this Lie group from a differential view: if a particle is at a particular set of coordinates at a particular time, what are its coordinates a short time later? This information is given by the Lie algebra.

An **algebra** over the reals is a vector space S, with a multiplication rule $A: S \times S \to S$ satisfying the bilinearity properties:

1) $A(\alpha s,t) = A(s,\alpha t) = \alpha A(s,t)$ where $\alpha \in \mathbf{R}$; $s,t \in S$;

2) $A(s,t + v) = A(s,t) + A(s,v)$

$A(s + t,v) = A(s,v) + A(t,v)$.

The algebra will be called **associative** if $A(s,A(t,v)) = A(A(s,t),v)$. The algebra is a **Lie algebra** if the multiplication satisfies **antisymmetry** and the **Jacobi identity**:

3) $A(s,t) = -A(t,s)$

4) $A(s,A(t,u)) + A(t,A(u,s)) + A(u,A(s,t)) = 0$

A Lie algebra multiplication is usually indicated with brackets [,].

A **linear operator** u on a vector space S is a mapping from the vector space to itself, $u:S \to S$. Call the space of all linear operators S*. It has a vector space structure adopted from S. If $u,v \in S^*$, α,β scalars, and $s \in S$, then

$$(\alpha u + \beta v)\ s = \alpha u(s) + \beta v(s).$$

The composition of operators makes S* into an associative algebra.

Repeated composition of operators shall be indicated with a superscript, by analogy with real numbers:

$$u^2 = uu\ ,\ u^3 = uuu\ ,\ \text{etc.} \qquad (1.32)$$

An operator with the superscript 0 is the identity.

A **derivation** D of an algebra S is a linear operator that satisfies

$$DA(s,t) = A(Ds,t) + A(s,Dt). \tag{1.33}$$

One may verify by induction that

$$D^n A(s,t) = \sum_{m=0}^{n} \binom{n}{m} A(D^m s,\ D^{n-m} t), \tag{1.34}$$

where $\binom{n}{m} = \frac{n!}{m!(n-m)!}$ is the binomial coefficient.

An associative algebra can be made into a Lie algebra by defining the Lie product via the operation

$$[s,t] = st - ts, \tag{1.35}$$

which the reader may verify gives a Lie algebraic structure. It is called the **commutator Lie algebra.**

The derivations do not form a subalgebra of S* under composition. That is, the composition of two derivations is not in general a derivation. However, by forming the commutator Lie algebra

$$[D_1,D_2] = D_1D_2 - D_2D_1 \tag{1.36}$$

we can make the space of derivations a subalgebra, because $[D_1,D_2]$ will always be a derivation if D_1 and D_2 are, as may be easily verified.

A useful concept when dealing with Lie algebras is that of the **adjoint map.** An adjoint map of S gives, for each element of S, a linear operator on S,

$$Ad_A: S \to S^* \tag{1.37}$$

in the following way:

$$Ad_A(f) = A(f,\cdot). \tag{1.38}$$

That is, multiplication by a fixed element of the algebra gives an operator, and the map Ad_A promotes that fixed element to that operator.

Since S* is an associative algebra, it can be made into a commutator Lie algebra.

The map Ad from the underlying Lie algebra S to S* is a (Lie algebra) homomorphism, that is, it preserves the Lie structure:

$$[Ad\ s,\ Ad\ t] = Ad[s,t], \tag{1.39}$$

which may easily be verified from the Jacobi identity and the anti-symmetry property:

$$\begin{aligned}
[Ad\ s,\ Ad\ t] &= Ad\ s\ Ad\ t - Ad\ t\ Ad\ s \qquad (1.40)\\
&= [s,[t,\cdot]] - [t,[s,\cdot]]\\
&= [s[t,\cdot]] + [t,[\cdot,s]]\\
&= -[\cdot,[s,t]]\\
&= [[s,t],\cdot] = Ad[s,t],
\end{aligned}$$

where • indicates an unspecified arbitrary argument of S upon which the operators are to act.

An operator in the image of Ad is called a **Lie operator.** If the Lie algebra is a commutator Lie algebra, such an operator is a derivation on the associative algebra:

$$(\mathrm{Ad}\ s)\ tu = [s,tu] = stu - tus \tag{1.41}$$

$$= stu - tsu + tsu - tus$$

$$= [s,t]\ u + t[s,u]$$

$$= ((\mathrm{ad}\ s)\ t)\ u + (t(\mathrm{ad}\ s)\ u).$$

Also, the Jacobi identity, together with the antisymmetry condition, means that a Lie operator is a derivation on the underlying Lie algebra, whether or not it is a commutator Lie algebra

$$(\mathrm{Ad}\ s)\ [t,u] = [s,[t,u]] = -\ [u,[s,t]] - [t,[u,s]] \tag{1.42}$$

$$= [(\mathrm{Ad}\ s)t,u] + [t,(\mathrm{Ad}\ s)u].$$

Lie operators are sometimes called **inner derivations.**

Now we may apply these results. Let S be the space of continuous functions on phase space with at least one derivative,

$$S = \{f:\mathbf{R}^6 \to \mathbf{R} \mid f \in C^1\}. \tag{1.43}$$

Consider the algebra given by pointwise multiplication on this space

$$A(f,g) \equiv fg \equiv \{h \mid [h(x) = f(x)\ g(x)\ \ \forall_{x \in \mathbf{R}^n}\}. \qquad (1.44)$$

Consider another multiplication operation [,] that makes this space a Lie algebra, whose adjoint is a derivation in A

$$[f,gh] = [f,g]h + g[f,h] \qquad (1.45)$$

Furthermore, let the values on the phase space coordinates be

$$[\zeta_i,\zeta_j] = J_{ij} \qquad (1.46)$$

where we are considering the phase space variables ζ as functions.

These rules uniquely define the Poisson Bracket Lie Algebra multiplication, which we indicate by [,]. The reader should convince himself that the rules imply the relation given before:

$$[f,g] = \nabla_\zeta f \cdot J \cdot \nabla_\zeta g, \qquad f,\ g \in S. \qquad (1.47)$$

In the Poisson Bracket Lie Algebra we indicate the adjoint with a pair of colons

$$:f:g = [f,g]. \qquad (1.48)$$

Because Ad is not a bijection (one-to-one and onto map), however, it is not an isomorphism. For example :f + c: = :f:, where c = constant, be-

cause :c: = 0. The kernel of the adjoint, that is the set of all points that map to zero, is called the **center** of the algebra; clearly, then, in this Lie algebra, the functions constant on phase space are the center.

The time evolution of phase space functions in a Hamiltonian system is governed by this Lie algebra:

$$\dot{\zeta} = -[H,\zeta] \tag{1.49a}$$

or

$$\dot{\zeta} = - :H:\zeta. \tag{1.49b}$$

Obviously, the Lie operator :H: is very important in Hamiltonian systems, and one may reasonably expect that Lie algebras can play a significant role in analyzing these systems. Despite its importance, it is not practical in exactly this form. In accelerator physics, we usually want to find the coordinates after a finite time (or axial position) rather than the instantaneous rate of change. In other words, we need the integral rather than the differential form of the dynamical equations. In this case, Lie transformations are more useful.

d. Lie Transformations

We have thus far seen that Hamiltonian flows give rise to symplectic maps or canonical transformations which form a Lie group; we also have seen that the differential form of a Hamiltonian flow is governed by Lie operators which form a Lie algebra. One may conclude that Lie got his name on everything. One may also wonder what the

relation is between the two; it is given by the exponential of a Lie operator, called a (yes) Lie transformation.

As motivation for the use of a Lie transformation, consider the dynamical differential equation

$$\dot{\zeta} = -:H(t):\zeta. \tag{1.50}$$

This reminds one of the ordinary differential equation

$$f'(x) = g(x)\, f(x). \tag{1.51}$$

The solution to this, of course, is

$$f(x) = b\, e^{\int_a^x g(x')\, dx'}, \tag{1.52}$$

where $b = f(a)$ is the initial condition. Thus we might propose that the solution of (1.50) is, if $:H(t):$ commutes with itself at different times,

$$\zeta(t) = e^{-\int_o^t :H(t'): dt'}\, \zeta(0). \tag{1.53}$$

Here we must define the exponential of a Lie operator suitably. If we define it with the Taylor expansion that the ordinary exponential has,

$$e^{:f:} = \sum_{n=0}^{\infty} \frac{:f:^n}{n!}, \tag{1.54}$$

then it can be shown that (1.53) solves the differential equation

(1.50).

These exponentials of Lie operators give elements of the Lie group of symplectic maps and are called **Lie transformations.** We say that the Lie algebra **generates** the Lie transformation (group). The Lie transformations have the remarkable property

$$e^{:f:}[g,h] = [e^{:f:}g, e^{:f:}h]. \tag{1.55}$$

This may be shown as follows. Using the derivation property (1.34)

$$e^{:f:}[g,h] = \sum_{n=0}^{\infty} \frac{1}{n!} :f^n: [g,h] \tag{1.56}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} \sum_{m=0}^{n} \frac{n!}{m!(n-m)!} [(:f:^m g,),(:f:^{n-m} h)]$$

$$= \sum_{n=0}^{\infty} \sum_{m=0}^{n} [\frac{:f:^m g}{m!}, \frac{:f:^{n-m} h}{(n-m)!}]$$

$$= \sum_{m=0}^{\infty} \sum_{\ell=0}^{\infty} [\frac{:f:^m g}{m!}, \frac{:f:^{\ell} h}{\ell!}]$$

$$= [e^{:f:}g, e^{:f:}h].$$

An identical calculation also based on the derivation rule yields

$$e^{:f:}(gh) = (e^{:f:}g)(e^{:f:}h). \tag{1.57}$$

This means that the transformation of a polynomial may be done on the coordinates. A polynomial is just a sum of monomials of the form

$$\alpha\, \zeta_1^{m_1}\, \zeta_2^{m_2} \cdots \zeta_{2n}^{m_{2n}} \tag{1.58}$$

Then because of the relation (1.57) the Lie transformation may be distributed across the product,

$$e^{:f(\zeta):}\, (\alpha\, \zeta_1^{m_1} \zeta_2^{m_2} \cdots \zeta_{2n}^{m_{2n}})$$

$$= \alpha\, (e^{:f(\zeta):}\, \zeta_1^{m_1})(e^{:f(\zeta):}\, \zeta_2^{m_2}) \cdots (e^{:f(\zeta):}\, \zeta_{2n}^{m_{2n}})$$

and by linearity for any polynomial,

$$e^{:f(\zeta):} g(\zeta) = = g(e^{:f(\zeta):}\zeta). \tag{1.59}$$

It is clear by the above relations that transformations of the form $e^{:f:}$ are canonical for any f: if $\bar{\zeta} = e^{:f:}\zeta$, then

$$[\bar{\zeta}_i, \bar{\zeta}_j] = [e^{:f:}\zeta_i, e^{:f:}\zeta_j] \tag{1.60}$$

$$= e^{:f:}[\zeta_i, \zeta_j] = e^{:f:} J_{ij} = J_{ij}.$$

The converse is more interesting: given a canonical transformation, $\zeta \to \bar{\zeta}$ is there an f such that $\bar{\zeta} = e^{:f:}\zeta$? In general the answer is no. However, as we saw at the beginning of the section, the canonical transformation occurring under the flow of a Hamiltonian that commutes with itself at different times can be represented this way. Note that real accelerators have time-dependent Hamiltonians that do not commute with themselves, because the electromagnetic field seen by a given particle

depends on the time. In general it will not be possible to make a single Lie transformation representing the Hamiltonian flow. However, we can approximate the transformation by a finite series of transformations I shall discuss in section f.

From the Poisson bracket Lie algebra A of functions on phase space S, we used the map Ad to induce the adjoint Lie Algebra A*, or Commutator Lie Algebra S*. Since this is itself a Lie Algebra, it is possible to get <u>its</u> adjoint. Define

$$Ad_{A^*}: S^* \to S^{**} \tag{1.61}$$

where $S^{**}:S^* \to S^*$ is the space of operators on S^*. We denote the map Ad_{A^*} by surrounding with '#', that is

$$\#:f:\#:g: = [:f:,:g:] = :f::g: - :g::f:. \tag{1.62}$$

Douglas [1982] uses '^' as an abbreviation for #: :#, e.g. $\hat{f} \equiv \#:f:\#$. The Lie Algebra in S** is a commutator, as in S*

$$[\hat{f},\hat{g}] = \hat{\hat{f}}\hat{g} - \hat{\hat{g}}\hat{f} . \tag{1.63}$$

The following theorem is useful for exchanging the order of two Lie transformations

<u>Theorem</u> (Douglas [1982], Dragt and Finn [1976]). If $f,g \in F$ with $n \in \mathbf{Z}$, then

(a) $$e^{\hat{f}}:g: = :e^{:f:}g: \tag{1.64}$$

(b) $$e^{:f:}:g:^n e^{-:f:} = (e^{\hat{f}}:g:)^n = :e^{:f:}g:^n \tag{1.65}$$

(c) $$e^{:f:}e^{:g:}e^{-:f:} = \exp(e^{\hat{f}}:g:) = \exp(:e^{:f:}g:). \tag{1.66}$$

Here $e^{\hat{f}}$ is defined as

$$e^{\hat{f}} = \sum_{n=0}^{\infty} \frac{1}{n!} \hat{f}^n . \tag{1.67}$$

Proof

(a) $$e^{\hat{f}}:g: = \sum_{n=0}^{\infty} \frac{\hat{f}^n}{n!} :g: = \sum_{n=0}^{\infty} \frac{1}{n!} ::f:^n g: \tag{1.68}$$

$$= \sum_{n=0}^{\infty} : \frac{:f:^n}{n!} g: = :e^{:f:}g:.$$

To show (b) start with n=1. Let τ be a real parameter, and define

$$:h(\tau): = e^{\tau:f:}:f:e^{-\tau:f:}. \tag{1.69}$$

Differentiation gives

$$\frac{d:h:}{d\tau} = :f::h: - :h::f: = \hat{f}:h:, \tag{1.70}$$

which has the solution

$$:h(\tau) = e^{\tau\hat{f}}:g:. \tag{1.71}$$

Setting $\tau=1$ yields

$$e^{:f:}:g:e^{-:f:} = e^{\hat{f}}:g:. \tag{1.72}$$

For $n > 1$, note $e^{-:f:}e^{:f:}$ is the identity so

$$e^{:f:}:g:^n e^{-:f:} = e^{:f:}:g:e^{-:f:}e^{:f:}:g:e^{-:f:} \cdots e^{:f:}:g:e^{-:f:}$$

$$= (e^{\hat{f}}:g:)^n = :e^{:f:}g:^n. \tag{1.73}$$

Finally, we have

$$e^{:f:}e^{:g:}e^{-:f:} = e^{:f:}\left(\sum_{n=0}^{\infty} \frac{:g:^n}{n!}\right) e^{-:f:} \tag{1.74}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} e^{:f:}:g:^n e^{-:f:}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!} :e^{:f:}g:^n$$

$$= e^{:e^{:f:}g:} = e^{e^{\hat{f}}:g:} .$$

This relation is useful if we have, say,

$$e^{:f:}e^{:g:}, \tag{1.75}$$

and want the transformation $e^{:f:}$ on the right. Then, using $e^{-:f:}e^{:f:} =$ identity, we have the result

$$e^{:f:}e^{:g:} = e^{:f:}e^{:g:}e^{-:f:}e^{:f:} \tag{1.76}$$

$$= e^{:e^{:f:}g:}e^{:f:} = e^{:g^T:}e^{:f:}$$

where

$$g^T = e^{:f:}g. \tag{1.77}$$

This will be called the **transformation rule.**

e. Canonical Transformations to Convenient Coordinates

In this section, following Douglas [1982] and Dragt [1981], I shall make certain coordinate changes to facilitate the use of Lie algebraic methods in particle accelerator physics.

The first step is to make a canonical transformation so that the independent variable is no longer flight time t, but distance along the flight path z. Generally speaking, a dependent variable Q_i in a Hamiltonian system may be made into the independent variable if

$$\dot{Q}_i = \frac{\partial H}{\partial P_i} \neq 0. \tag{1.78}$$

Formally, the theorem is (Douglas [1982], p. 94, Dragt [1981], p. 151)

Theorem Let H(z,t) be the Hamiltonian for a system with n degrees of freedom $(Q_1,P_1, \ldots, Q_n,P_n)$. Suppose

$$\dot{Q}_i = \frac{\partial H}{\partial P_i} \neq 0 \tag{1.79}$$

in some region of state space. Then within this region, Q_i can be introduced as the independent variable replacing the time t. Moreover, the equations of motion with Q_i as independent variable may be obtained by using $K = -P_l$ as an effective Hamiltonian.

For a proof, which is based on the implicit function theorem, the reader is urged to consult the references given above.

It should be clear that for any reasonable motion of particles in an accelerator, this condition holds for z:

$$\dot{z} \neq 0 \tag{1.80}$$

since z(t) is, we hope, a monotonically increasing function of time. Thus we may take as the Hamiltonian the quantity

$$\kappa(x,p_x,y,p_y,t,p_t;z) = -p_z. \tag{1.81}$$

The second step is to make a canonical transformation measuring time and its momentum as deviations from the design trajectory

$$t^*(z) = t(z) - t^o(z) \tag{1.82a}$$

$$p_t^*(z) = p_t(z) - p_t^o(z) \tag{1.82b}$$

where the superscript o indicates the value of the quantity on the design trajectory, and the superscript * indicates the new coordinates. This is a canonical transformation. We may use a generating function of type 2 (Goldstein [1950], p. 240)

$$F(t,p_t^*;z) = (t - t^o(z))\ (p_t^* + p_t^o(z)). \qquad (1.83)$$

Then the new Hamiltonian is

$$\kappa^{NEW}(x,_x,y,p_y,t^*,p_t^*;z) = K(x,p_x,y,p_y,t,p_t;z) + \frac{\partial F}{\partial z}\ . \qquad (1.84)$$

In this system of coordinates, the design trajectory has the value (0,0,0,0,0,0) and particles "near" the design trajectory will be described by "small" values of the phase space variables. Thus the motion is amenable to a perturbation description, and the Taylor series described in section a will have validity.

The notions of smallness and nearness can be given precision by scaling these variables so that the result is dimensionless. This scaling preserves the Hamiltonian form of the equations of motion. Choose an arbitrary scale length ℓ; it could be, for instance, the bending radius of the machine. Choose as the scale momentum p_o the design momentum. Then

$$X = \frac{x}{\ell} \qquad (1.85a)$$

$$P_X = \frac{p_x}{p_o} \qquad (1.85b)$$

$$Y = \frac{y}{\ell} \qquad (1.85c)$$

$$P_Y = \frac{p_y}{p_o} \qquad (1.85d)$$

$$T = \frac{t^*}{(\ell/c)} \qquad (1.85e)$$

$$P_T = \frac{P_t^*}{p_o c} \ . \tag{1.85f}$$

These imply that the Hamiltonian must be scaled

$$K = \frac{\kappa}{p_o} \ . \tag{1.85g}$$

f. The Relation Between Lie Transformations and Symplectic Maps

Lie operators and transformations are more general than the corresponding map and symplectic maps on phase space. A symplectic map is a map M: $S \to S$, that is, M maps phase space into itself. A Lie transformation, $e^{:f:}$, or series of the form $e^{:f:}e^{:g:} \ldots$, on the other hand, acts on real functions of phase space $\bar{S}$: $S \to \mathbf{R}$.

$$e^{:f:}:\bar{S} \to \bar{S} \ , \tag{1.86}$$

or sets of them, $\bar{S}^m$: $S \to \mathbf{R}$,

$$e^{:f:}:\bar{S}^m \to \bar{S}^m \ . \tag{1.87}$$

In particular, we may consider the set of 2n functions ζ which give each of the 2n canonical coordinates X, P_x, etc., in succession. If we apply a Lie transformation or series, then substitute specific values v_o for ζ, we will have a symplectic map:

$$v = e^{:f:}e^{:g:} \ldots \zeta \,|_{\zeta = v_o} . \tag{1.88}$$

We then say $e^{:f:}e^{:g:}$ **... corresponds** to M, and vice versa.

At this point a brief explanation of the notation and terminology used in this thesis is in order. As I have used the symbols above, the Greek letters ζ, ξ, ... will stand for the dependent phase space variables

$$\zeta = (X, P_X, \ldots), \tag{1.89}$$

either as abstract symbols or as functions on phase space where now the new variables are used. For example,

$$\bar{\zeta} = e^{:f_2(\zeta):}\zeta, \tag{1.90}$$

means to apply the transformation formed from the homogeneous second-order polynomial f_2 of the phase space variables to each of the phase space variables separately, i.e.

$$\bar{X} = e^{:f_2(\zeta):}X, \tag{1.91}$$

$$\bar{P}_X = e^{:f_2(\zeta):}P_x,$$

etc.

To indicate specific values of these variables - i.e., a point in $\mathbf{R}^6$ phase space, I shall use lower case Latin letters v,w, ... For example,

$$v = e^{:f_2(\zeta):}\, \zeta\,|_{\zeta=v_o}\,, \tag{1.92}$$

means to make the transformation above, then substitute the values $X = v_{oX}$, $P_x = v_{oP_x}$, etc., in order to get 6 values corresponding to a new point in R^6.

A convenient shortcut will be to leave off the symbol ζ for the phase space variables. Where its presence will be missed, a bullet ($\bullet$) will be used. For example, the equation (1.92) above can be written as

$$v = e^{:f_2:}\,\bullet\,|_{v_o}\,\bullet \tag{1.93}$$

The distinction made above between the Lie transformations and symplectic maps based on the objects on which they act may seem like an irrelevant technicality until one considers a series of Lie transformations.

Consider just two Lie transformations, and their effect on phase phase: let $M_f = e^{:f:}$, and $M_g = e^{:g:}$. Suppose we look at their successive effect on phase space,

$$e^{:f(\zeta):}e^{:g(\zeta):}\,\zeta \tag{1.94}$$

$$= e^{:f(\zeta):}e^{:g(\zeta):}e^{-:f(\zeta):}e^{:f(\zeta):}\,\zeta$$

$$= e^{:e^{:f:}g(\zeta):}e^{:f(\zeta):}\,\zeta$$

$$= e^{:g(e^{:f:}\zeta):}e^{:f(\zeta):}\,\zeta$$

$$= e^{:g(\xi):}\,\xi, \text{ where } \xi = e^{:f(\zeta):}\,\zeta$$

or

$$e^{:f:}e^{:g:} \cdot \big|_v = e^{:g:} \cdot \big|_{e^{:f:} \cdot \big|_v} \cdot \tag{1.95}$$

We see that the corresponding symplectic transformations must be applied in reverse order:

$$M_f\, M_g \cdot\big|_v = e^{:f:}e^{:g:} \cdot \big|_v = M_g \cdot\big|_{M_f} \cdot\big|_v \tag{1.96}$$

In the future, I shall occasionally abuse notation and write v for $M \cdot\big|_v$, when doing so will not cause confusion.

Now that we have a correspondence between the Lie transformations and symplectic maps, we shall take a look at the subgroup of linear transformations and their Lie algebras.

For a homogeneous second-order polynomial, f_2, the Lie transformation

$$e^{:f_2:} \tag{1.97}$$

is a linear transformation, because each application of the operator $:f_2:$ leaves the order of its argument unchanged. Thus, the corresponding symplectic map is linear and we may represent it with the symplectic matrix M,

$$\bar{v} = e^{:f_2:} \cdot \big|_v = Mv = e^{JS}\, v. \tag{1.98}$$

In this case, S is a symmetric matrix obtained from the coefficients of f_2. Thus we have a mapping of the set of Lie transformations of the form $e^{:f_2:}$ into Sp(6). It is an injective but not surjective mapping; there are symplectic matrices to which no $e^{:f_2:}$ corresponds (see Dragt and Finn [1976]).

This mapping has a corresponding homomorphism in the Lie algebra, which is easily computed. If $c_{\zeta_i\zeta_j}$ is the coefficent of $\zeta_i\zeta_j$ in f_2, then the matrix JS is given by

$$\begin{bmatrix}
-c_{XP_X} & -2c_{P_XP_X} & -c_{P_XY} & -c_{P_XP_Y} & -c_{P_XT} & -c_{P_XP_T} \\
2c_{XX} & c_{XP_X} & c_{XY} & c_{XP_Y} & c_{XT} & c_{XP_T} \\
-c_{XP_Y} & -c_{P_XP_Y} & -c_{YP_Y} & -2c_{P_YP_Y} & -c_{P_YT} & -c_{P_YP_T} \\
c_{XY} & c_{P_XY} & 2c_{YY} & c_{YP_Y} & c_{YT} & c_{YP_T} \\
-c_{XP_T} & -c_{P_XP_T} & -c_{YP_T} & -c_{P_YP_T} & -c_{TP_T} & -2c_{P_TP_T} \\
c_{XT} & c_{P_XT} & c_{YT} & c_{P_YT} & 2c_{TT} & c_{TP_T}
\end{bmatrix}. \tag{1.99}$$

g. The Factorization Theorem

Representing the Hamiltonian flow as $e^{-\int_o^z :H(z'): dz'}$ is awkward for combining and tracking from a computational standpoint because the exponential series will in general never terminate, and there will be no

reasonable basis for truncating it.

For this reason, the factorization theorem is very powerful. It allows us to split up any analytic symplectic map, order by order, and stop at any point.

Before giving the theorem, let us take a look at the effect of the Poisson bracket operation on the order of polynomials in Lie operators. Let a subscript n on a polynomial indicate that it is homogeneous in nth order in the phase space variables; e.g. $f_3 = 5X^2P_Y - YP_T^2$ is homogeneous third-order. In a Poisson bracket of homogeneous polynomials, the resultant order is two less than the sum of the orders:

$$[f_n,g_m] = h_{n+m-2} \quad (n+m \geq 2) \tag{1.100}$$

because there are two derivatives, and a multiplication. Thus, a Lie operator $:f_n:$ raises the order of its argument by n-2. In particular, $:f_1:$ lowers it by 1, $:f_2:$ does not change the order, $:f_3:$ increases it by one, etc.

As Lie transformations, $e^{:f_2:}$ corresponds to a linear map, $e^{:f_3:}$ in general corresponds to quadratic and all higher orders, $e^{:f_4:}$ corresponds to only higher even orders, and so on. I shall refer to $e^{:f_n:}$ as an nth-order transformation.

Now we are ready for the factorization theorem, an extension of the theorem and proof in Dragt and Finn [1976], Dragt [1981].

Theorem (Factorization) Let M be an analytic symplectic map. That is, suppose the relation

$$\bar{z} = Mz$$

can be written as a Taylor series in the form

$$\bar{z}_i = F_i(z) = \sum_{|\sigma| > 0} a_i(\sigma)\, z^\sigma \qquad (1.101)$$

where σ is a collection of exponents $\sigma_1, \sigma_2, \ldots \sigma_{2n}$ and

$$|\sigma| = \sum_1^{2n} \sigma_i \quad , \quad z^\sigma = z_1^{\sigma_1} \cdots z_{2n}^{\sigma_{2n}} \, . \qquad (1.102)$$

Then there exist homogeneous polynomials f_1, f_2^c, f_2^a, f_3, $\ldots$. of degree 1, 2, 2, 3, $\ldots$. such that the map (1.101) can be written in the form

$$\bar{z} = [e^{:f_2^c:} e^{:f_2^a:} e^{:e_3:} \cdots e^{:f_1:}] \cdot |_z . \qquad (1.103)$$

These polynomials are unique.

Proof First, split off the constant terms c_i

$$\bar{z}_i = c_i + z_i^* = c_i + F_i^*(z) = c_i + \sum_{|\sigma| > 0} a_i(\sigma)\, z^\sigma \qquad (1.104)$$

and put them aside.

Let M(z) be the Jacobian matrix of the functions F*(z). Then

$$M(0) = L \qquad (1.105)$$

where the matrix L is defined as being the coefficients of the linear

part of F(z):

$$L_{ij} = a_i(\rho_j), \tag{1.106}$$

where ρ_j is a collection of 2n integers, the jth integer being 1 and the rest 0.

Since M(z) is symplectic for all values of z, so is L. Thus, by a theorem of Dragt [1982], it can be written in the form

$$L = e^{JS^a} e^{JS^c}, \tag{1.107}$$

where S^a is a symmetric matrix that anticommutes with J and S^c is a symmetric matrix that commutes with J. Because of the Lie algebra isomorphism between the matrix Lie algebra and the polynomial Poisson bracket Lie algebra, we may find the second-order polynomials that are the image under S^a and S^c. We shall call them f_2^a and f_2^c. The isomorphism between the corresponding Lie groups therefore gives the appropriate maps $e^{:f_2^a:}$ and $e^{:f_2^c:}$, and their product $e^{:f_2^c:} e^{:f_2^a:}$ for L.

Now we note that the action of the linear transformations on z_i^* is

$$e^{-:f_2^a:} e^{-:f_2^c:} \cdot|_{z_i^*} = z_i + r(> 1), \tag{1.108}$$

where r(> 1) is a generic symbol for a polynomial of the phase space variables consisting of terms higher than first degree. To show this, use the expansion for F* written as

$$z_i^* = F_i^*(z) = \sum_{|\sigma|=1} a_i(\sigma)\, z^\sigma + \sum_{|\sigma|>1} a_i(\sigma)\, z^\sigma \qquad (1.109)$$

apply $e^{-:f_2^a:}\, e^{-:f_2^c:}$ to both sides to get

$$e^{-:f_2^a:}\, e^{-:f_2^c:} \cdot |_{z_i^*} = \sum_{|\sigma|=1} a_i(\sigma)\, e^{-:f_2^a:}\, e^{-:f_2^c:} \cdot |_{z^\sigma} + r(>1) \qquad (1.110)$$

which can be written

$$e^{-:f_2^a:}\, e^{-:f_2^c:} \cdot |_{z_i^*} = \sum_j L_{ij}\, e^{-:f_2^a:}\, e^{-:f_2^c:} \cdot |_{z_j} + r(>1). \qquad (1.111)$$

The correspondence between L and $e^{:f_2^c:}\, e^{:f_2^a:}$ gives

$$e^{-:f_2^a:}\, e^{-:f_2^c:} \cdot |_{z_j} = \sum_k (L^{-1})_{jk}\, z_k. \qquad (1.112)$$

Thus we obtain the desired result

$$e^{-:f_2^a:}\, e^{-:f_2^c:} \cdot |_{z_i^*} = z_i + r(>1). \qquad (1.113)$$

Next, we assume the series (1.103) exists to order n, and extend it to order n+1. Assuming the series

$$f_2^c,\ f_2^a,\ \ldots,\ f_n \qquad (n \geq 2) \qquad (1.114)$$

is known, so that F* may be written

$$F^*(z) = e^{:f_2^c:}\, e^{:f_2^a:} \cdots e^{:f_n:}\, z + r(>n-1), \qquad (1.115)$$

a term f_{n+1} may be obtained such that the remainder term is of order n:

$$z^* = e^{:f_2^c:} e^{:f_2^a:} \cdots e^{:f_n:} e^{:f_{n+1}:} \cdot |_z + r(> n). \qquad (1.116)$$

Because we have assumed that the series (1.114) has been carried through order n, there is some homogeneous polynomial of order n, g_n, such that

$$e^{-:f_n:} \cdots e^{-:f_2^c:} z_i \cdot |_{z_i^*} = z_i + g_n^i(z) + r(> n). \qquad (1.117)$$

By forming the Poisson Bracket of this with j replacing i in it, we obtain

$$J_{ij} = [z_i + g_n^i(z) + r(> n),\ z_j + g_n^i(z) + r(> n)] \qquad (1.118)$$

$$= J_{ij} + [z_i,\ g_n^i(z)] + [z_j,\ g_n^i(z)] + r(> n - 1).$$

Looking at the terms of order n-1 in z, we see

$$[z_i,\ g_n^j] + [g_n^i,\ z_j] = 0. \qquad (1.119)$$

This implies there exists a function f_{n+1}

$$g_n^i(z) = :f_{n+1}:z_i \qquad (1.120)$$

which may be concluded by noting that (1.119) may be rephrased as

$$\frac{\partial g_n^j}{\partial z_i^\dagger} = \frac{\partial g_n^i}{\partial z_j^\dagger} \qquad (1.121)$$

where $z_i^\dagger = \sum_j J_{ij} z_j$. Thus $\sum_i g_n \, dz_i^\dagger$ is an exact differential, so

$$f_{n+1}(z) = - \int^z \sum_{i,j} f_i(z') \, J_{ij} dz'_j \tag{1.122}$$

is the function desired.

Now

$$e^{-:f_n:} \cdots e^{-:f_2^c:} \cdot |_{z_i} = z_i + :f_{n+1}: z_i + r \; (> n) \tag{1.123}$$

so

$$e^{-:f_{n+1}:} e^{-:f_n:} \cdots e^{-:f_2^c:} \cdot |_{z_i^*} = z_i + r \; (> n). \tag{1.124}$$

Inverting the maps on left-hand side, and applying them to both sides,

$$z_i^* = e^{:f_2^c:} e^{:f_2^a:} \cdots e^{:f_{n+1}:} \cdot |_{z_i} + r \; (> n). \tag{1.125}$$

This gives us an inductive rule for carrying the series to any order.

The one task that remains is to account for the constant term; this is now easy to do. Recall

$$\bar{z} = F(z) = c + F^*(z), \tag{1.126}$$

where the component notation has been left off. Find an f_1 such that

$$c = [f_1, \zeta]. \tag{1.128}$$

This is solved by

$$f_1(\zeta) = -\sum_{k,\ell} c'_k J_{k\ell} \zeta_\ell . \tag{1.129}$$

Finally, we have a complete factorization,

$$F(z) = e^{:f_2^c:} e^{:f_2^a:} \cdots e^{:f_n:} e^{:f_1:} \cdot|_z + r(> n-1), \tag{1.130}$$

which may be verified by applying the map to a specific point in phase space. By applying the transformations as in (1.95), we have

$$F(z) = e^{:f_2^c:} e^{:f_2^a:} \cdots e^{:f_n:} e^{:f_1:} \cdot|_z + r(> n-1) \tag{1.131}$$

$$= e^{:f_1:} \cdot| \qquad\qquad + r(>n-1) .$$
$$e^{:f_2^c:} e^{:f_2^a:} \cdots e^{:f_n:} \cdot|_z$$

Then observe that $e^{:f_1:}$ evaluated at z is just $z + c$,

$$e^{:f_1:} \cdot|_z = (\zeta + [f_1,\zeta])\,|_{\zeta=z} \tag{1.132}$$

$$= z + [f_1,\zeta]$$

$$= z + c,$$

and so

$$F(z) = e^{:f_2^c:} e^{:f_2^a:} \cdots e^{:f_n:} \cdot|_z + c + r(> n-1) = F^*(z) + c + r(> n-1) \tag{1.133}$$

as desired.

In practice we would want to cut off any of these series after a certain number of terms, say four:

$$e^{:f_2:} e^{:f_3:} e^{:f_4:} e^{:f_1:}. \tag{1.134}$$

Although this would be symplectic and in general would give terms of all orders, it would be accurate only through third order. Thus, as with a Taylor expansion, we are using a perturbation method. Therefore, we must know that the remaining terms are insignificant. This will be true if the dimensionless momenta are small, and if the coordinates are small with respect to the scale of non-linearities in the system.

It will be more useful for our purposes to factorize with the first-order term on the left

$$M = e^{:f_1:} e^{:f_2^c:} e^{:f_2^a:} e^{:f_3:} e^{:f_4:} \cdots . \tag{1.135}$$

This is a much more difficult problem. If, however, we keep terms in the Taylor series (1.101) only through order n, which we have to do in practice anyway, we may use the factorization theorem to write this as

$$\bar{z} = e^{:f_2^c:} e^{:f_2^a:} e^{:f_3:} \cdots e^{:f_{n+1}:} e^{:f_1:} \cdot|_z + r(> n). \tag{1.136}$$

The concatenation techniques of Chapter 4 will show us how to rewrite this with the first-order term on the left,

$$\bar{z} = e^{:g_1:} e^{:g_2^c:} e^{:g_2^a:} e^{:g_3:} \cdots e^{:g_{n+1}:} \cdot|_z + r_{c,z}(> n). \tag{1.137}$$

Notice how the remainder polynomial $r_{c,z}$ is now a function of both z and c, the constants in F. Thus, this factorization is only accurate through a set order in the constant terms. This restriction shall be no great burden, as we shall assume the machine errors (misalignment, mispowering, etc.) which give rise to the constant terms are small.

The form (1.137) in ascending order shall be the standard factorization we shall use. However, we shall occasionally need the descending-order factorization

$$M = \ldots e^{:f_4:} e^{:f_3:} e^{:f_2^c:} e^{:f_2^a:} e^{:f_1:} . \qquad (1.138)$$

It is possible to write this from the form (1.135) given truncation at a certain order.

2. Ray Tracing

In tracing rays we are given a set of phase space variable values v_o at a particular time (or axial position z) and ask what the values v are after having been transformed by the map M:

$$v = Mv_o. \tag{2.1}$$

In a circular accelerator, for instance, M could be the map for one turn around the machine. Then if v_o is the phase space position of a particle at a particular turn, v will be the position at the next turn. By applying M to v, the position at a subsequent turn will be obtained. This process may be repeated for any number of turns to find the long-term behavior of the machine.

With the Lie transformations the relation (2.1) would be written as

$$v = e^{:f_1(\zeta):}_e e^{:f_2(\zeta):}_e e^{:f_3(\zeta):}_e e^{:f_4(\zeta):} \cdots \zeta\,|_{\zeta=v_o} . \tag{2.2}$$

These Lie transformations may be rearranged using the transformation rule:

$$v = e^{:f_2(e^{:f_1(\zeta):}\zeta):}_e e^{:f_3(e^{:f_1(\zeta):}\zeta):}_e e^{:f_4(e^{:f_1(\zeta):}\zeta):} \cdots e^{:f_1(\zeta):}\zeta\,|_{\zeta=v_o} \tag{2.3}$$

In this equation, the first-order transformation of phase space

$$e^{:f_1(\zeta):}{}_{\zeta} \tag{2.4}$$

occurs often, so let us call it ξ. Then (2.3) can be shortened to

$$v = e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdots \xi \Big|_{\xi = e^{:f_1:}\zeta} \Big|_{\zeta = v_o} \tag{2.5}$$

$$= e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdots \Big|_{v_o + [f_1, \zeta]} .$$

If we call the coordinates shifted by the operator $:f_1:$ w,

$$w = v_o + [f_1, \zeta] \tag{2.6}$$

then v becomes

$$v = e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdots \Big|_w . \tag{2.7}$$

A similar process with the transformation $e^{:f_2:}$ gives

$$v = e^{:f_3:} e^{:f_4:} \cdots \Big|_{e^{:f_2:} \cdot |_w} . \tag{2.8}$$

It is more convenient from a practical standpoint to keep $e^{:f_2:}$ as simply the matrix of the linear transformation M. In this case,

$$v = e^{:f_3:} e^{:f_4:} \cdots \Big|_{Mw} \tag{2.9}$$

To compute the effect of the remaining terms we use a different tactic,

making use of the higher orders of the polynomials which increase the order with each term in the Taylor case.

The point at which the exponential series are truncated determines the order of the transformation. Since each homogeneous Lie operator $:f_n:$ of order n changes the order of its operand by n-2, and we wish to keep terms through order p, we may truncate the exponential of the Lie transformation $e^{:f_n:}$ after $[\frac{p-1}{n-2}]$ terms, where [x] means greatest integer less than x, and the identity is the zeroth term.

For example, if the terms in the ellipses are disregarded,

$$v = (1 + :f_3: + \frac{1}{2} :f_3:^2 + \ldots)(1 + :f_4: + \ldots) \ldots \cdot |_{Mw} \quad (2.10)$$

will be accurate through order 3. The end result can be given as a transfer map for each phase space variable

$$v = (\zeta + [f_3,\zeta] + [f_4,\zeta] + \frac{1}{2} [f_3,[f_3,\zeta]] + \ldots) |_{\zeta=Mw} \quad (2.11)$$

Once the terms in the exponential are truncated, the map is no longer symplectic. MARYLIE (Dragt et. al. [1985]) has an alternative method of tracking that symplectifies the truncated series, but I shall not go into it here, except to say that it may be applied to w to include the effects of f_1.

When factorized in the reverse order, the process is similar, but the linear and higher-order parts of the transformation do not act on the constant term

$$v = \ldots e^{:g_3:} e^{:g_2:} e^{:g_1:} \cdot |_{v_o} \qquad (2.12)$$

$$= Mw + c$$

where now

$$w = \ldots e^{:g_3:} \cdot |_{v_o} \qquad (2.13)$$

$$c = [g_1, \zeta]$$

The truncation point of the exponential series was determined above by the highest final order desired. This is a reasonable criterion for phase space variables small, so that the truncated series is close to the limit value. It assumed, however, that the function on which the transformation acted was just a single phase space variable. This would not be true if we had more than one map in succession.

Suppose the final value z is related to the initial z_o by two successive transformations M_f and M_g where M_f is represented by $e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:}$ and M_g by a similar series in $g_1, g_2, \ldots$ Then

$$z = e^{:g_1:} e^{:g_2:} e^{:g_3:} e^{:g_4:} e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdot |_{z_o} \qquad (2.14)$$

The g transformations yield the result above. However, we may no longer cut off the exponential series at any point, even if in the end we only desire a transformation of order p. This is because the succeeding transformation $e^{:f_1:}$ <u>decreases</u> the order of its operand by one for each term in the exponential. Thus, even if we need only third-order terms,

we would have to keep higher terms in the g, because the $e^{:f_1:}$ would bring it back below 3 in the end.

The resolution of this problem is to assume the first-order transformations are small in the same way the phase space variables are. Then we are supplied with a natural guide for truncation again. This is discussed in more detail in Chapters 3 and 4.

3. Ideal Structure of the Lie Algebra

The truncation of the Lie transformation series (1.135) or (1.137) both with and without a first-order term deserves closer investigation. It is of particular interest in concatenation. Given two maps in the standard factorization

$$M_f = e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdots \tag{3.1a}$$

$$M_g = e^{:g_1:} e^{:g_2:} e^{:g_3:} e^{:g_4:} \cdots \tag{3.1b}$$

What are the polynomials h_n such that

$$M_f M_g = M_h, \tag{3.1c}$$

where

$$M_h = e^{:h_1:} e^{:h_2:} e^{:h_3:} e^{:h_4:} \cdots ? \tag{3.1d}$$

If we truncate each series M_f, M_g at a fixed value of n, say n = 4, then it is also reasonable to truncate the M_h series at that value of n. In combining the polynomials, however, higher-order terms will be produced, as we shall see in Chapter 4. The question we need to answer is: by what standard are we permitted to ignore or choose arbitrarily the Poisson bracket of two polynomials $[f_n,g_n]$? The answer is divided into two parts: without a first-order term present, as is computed in MARYLIE 3.0 (Douglas [1982], Dragt et. al. [1985]), and with first-order

terms present, as we will need to handle misalignment and other machine errors.

a. First-Order Terms Absent

The presumption of the Lie transformation series (1.135) or (1.137) is that the correspondng Taylor series is convergent (see Chapter 1). Since we are truncating this series, we want the remainder term that is left off to be so small that it can be safely dropped. In order to do this, we take the values of each of the phase space variables ζ to be small so that sufficiently high orders may be ignored. Specifically, let each of the phase space variables carry the small factor δ, so that powers of δ count the order of these variables. Polynomials homogeneous of order n, f_n have a factor δ^n, and will be said to have δ-rank n.

The behavior of the δ-rank of polynomials in a Poisson bracket has been explored (1.100) in a slightly different guise; the δ-rank of the Poisson bracket is 2 less than the sum of the participants' δ-rank, provided each was at least 1.

Since δ is small we may, when taking Poisson brackets, neglect terms of a given order or higher. Thus, for example, if we choose to neglect terms of fourth order and higher, the Poisson bracket $[f_3,g_4]$ may be ignored.

We now give this process some rigor; before doing this however, it is necessary to introduce some new definitions.

A subset $S' \subseteq S$ is a **subalgebra** if $A(S',S') \subseteq S'$, where $A(S',S')$ is the image of the multiplication restricted to S'. A subset $S' \subseteq S$ is an **ideal** if $A(S',S) \subseteq S'$, that is, for $s' \in S'$, $s \in S$, $A(s',s) \in S'$. (For

a Lie algebra this is equivalent to if $A(S,S') \subseteq S'$.) Clearly, an ideal is also a subalgebra.

An algebra S is **graded** if S is the direct sum of subspaces S_i ($i = 0,1, \ldots, \infty$) and $A(S_i,S_j) \subseteq S_{i+j}$. An algebra S is **filtered** if for each non-negative integer i, there is a subspace $S^{(i)}$ such that

1) $S^{(i)} \subseteq S^{(j)}$ for $i \leq j$;
2) $\bigcup S^{(i)} = S$;
3) $A(S^{(i)},S^{(j)}) \subseteq S^{(i+j)}$.

If S is graded, then it is filtered by the rule

$$S^{(i)} = \bigoplus_{j \leq i} S_j. \tag{3.2}$$

Similarly, an algebra S is **complementary filtered*** if for each non-negative integer i, there is a subspace $S^{(i)}$ such that

1) $S^{(i)} \supseteq S^{(j)}$ for $i \leq j$;
2) $\bigcup S^{(i)} = S$;
3) $A(S^{(i)},S^{(j)}) \subseteq S^{(i+j)}$.

If S is graded, then it is complementary filtered by the rule

$$S^{(i)} = \bigoplus_{j \geq i} S_j. \tag{3.3}$$

*This is my own terminology.

It is clear from the above definition that each of the members $S^{(i)}$ of a complementary filter is an ideal. Let $s^{(i)} \in S^{(i)}$, $s \in S$. Therefore, there is a j such that $s \in S^{(j)}$. Then $A(s,s^{(i)}) \in S^{(i+j)} \subseteq S^{(i)}$, and also $A(s^{(i)},s) \in S^{(i+j)} \subseteq S^{(i)}$. Since s and $s^{(i)}$ were arbitrary within their respective sets, $S^{(i)}$ is an ideal in S.

Let S be a (Lie) algebra, I an ideal in S. Define S/I to be the set of equivalence classes given by the equivalence relation: $s_1 \cong s_2$ if $s_1 = s_2 + i$, for some $i \in I$. We denote these classes by $s + I$, where $s \in S$. Since an ideal is a subalgebra, S/I is a (Lie) algebra with the rules

$$(s_1 + I) + (s_2 + I) = (s_1 + s_2) + I, \tag{3.4a}$$

$$c(s_1 + I) = cs_1 + I, \; c \in \mathbf{R}, \tag{3.4b}$$

$$A(s_1 + I, \; s_2 + I) = A(s_1,s_2) + I. \tag{3.4c}$$

These rules are easily seen to be consistent. If i_1, $i_2 \in I$ are arbitrary, the left side of (3.4a) is

$$(s_1 + i_1) + (s_2 + i_2) = (s_1 + s_2) + (i_1 + i_2). \tag{3.5}$$

$$\in (s_1 + s_2) + I,$$

since I is a vector subspace of S. A similar argument holds for (3.4b). The left side of (3.4c) is

$$A(s_1 + i_1, s_2 + i_2) = A(s_1, s_2) + A(s_1, i_2) + A(i_1, s_2) + A(i_1, i_2). \quad (3.6)$$

From the definition of an ideal, the last three terms on the right side are in I, thus upholding (3.4c).

Let us now apply this to our particular problem: let S be the space of functions on phase space that have power series expansions with no first-order terms (note the more restricted definition than before). Grade it with subspaces of polynomials homogeneous in a particular order of the phase space variables: let

$$S_0 = \{\text{constants}\} \cup \{\text{homogeneous polynomials of order 2}\}, \quad (3.6a)$$

$$S_i = \{\text{homogeneous polynomials of order } i+2\} \text{ for } i > 0. \quad (3.6b)$$

One may easily verify that this is a grading on S under the Poisson Bracket. (Note that the constants are not relevant; they could have been included with any S_i). If a particular polynomial (or Lie operator) belongs to the subspace S_i (or S_i^*), then I will say it has δ-rank i.

Given this grading S_i by polynomial order, we have the corresponding complementary filter given by (3.3). This gives us a series of ideals $S^{(i)}$, and a series of quotient algebras $S/S^{(i)}$. The ideal $S^{(i)}$ consists of all power series with coefficients zero for the terms of order 1 through i+1. The quotient algebra $Q^{(i)} = S/S^{(i+1)}$ is a rigorous way of describing the algebra S but "neglecting terms of order i+3 and greater."

MARYLIE 3.0 (Douglas [1982], Dragt et al. [1985]) which does Lie algebraic computations through fourth order and has no first-order terms, is actually computing in the algebra $Q^{(2)}$. If f_n, g_n, and h_n are homogeneous n^{th} order polynomials,

$$[f_2,g_3] \cong g_3, \tag{3.7a}$$

$$[f_3,f_4] \cong 0, \tag{3.7b}$$

$$[f_4,g_4] \cong 0, \tag{3.7c}$$

etc.

The results that are in $S^{(3)}$ are taken to be 0, although any element of $S^{(3)}$ would be acceptable. In applying the Lie transformations to particle coordinates, portions of the result that are in $S^{(3)}$ may be chosen non-zero to insure that the overall result is symplectic.

The homomorphism Ad carries all these definitions to the adjoint algebra S*. S_i^* is all the Lie operators that are of order i, i.e., as images of S_i, and $S^{(i)*}$ are the direct sum of S_i^* or the image of $S^{(i)}$. The $S^{(i)*}$ are ideals, so the $Q^{(i)*} = S^*/S^{i+1)*}$ are quotient algebras. Thus the adjoint algebra has the same ideal and quotient structure as the underlying algebra, as we expect, and commutators of Lie operators are set to zero (or to an arbitrary value) when the Poisson Bracket of the corresponding polynomials would be.

The Lie groups generated by these Lie algebras and composition are as we would expect: the symplectic maps on the coordinates that result,

while containing terms of all orders, are accurate only through order i+1 for $Q^{(i)}$. Specifically, let G be the group of symplectic maps on phase space and $G^{(i)}$ (i = 0,1,...) be the subgroup of these maps that has a power series expansion consisting of terms only order i+1 and higher plus the identity. Then $G^{(i)}$ is a normal or invariant subgroup of G, that is, $ghg^{-1} \in G^{(i)}$ for all $g \in G$, $h \in G^{(i)}$. The quotient group $H^{(i)} = G/G^{(i+1)}$ is defined as the equivalence classes given by $g_1 \cong g_2$ if $g_1 = g_2 + h$, where $h \in G^{(i)}$. That is, two elements are the same if they differ only by terms of order i+1 or higher. That this is in fact a group may be easily verified. These groups $H^{(i)}$ are generated by the algebras $Q^{(i)}*$ and products.

b. With First-Order Transformations

If a first-order transformation is present, the truncation by δ-rank given in part (a) will not be sufficient because the δ-rank will be lowered by a first-order term. Suppose we keep through δ-rank 4, and discard anything higher. Then forming, for instance, $[f_1,[g_3,h_4]]$ would yield the wrong answer: in the first place $[g_3,h_4]$, we would take the answer as 0 because the δ-rank is 5, and so our overall answer would be zero. But this is not correct; even though the first Poisson bracket is δ-rank 5, the Poisson bracket with f_1 subsequently lowers the δ-rank to an acceptable 4.

These kinds of troubles may be avoided by considering the source of the first-order terms. As we shall see in part II, the first-order terms are proportional to a machine error, either a misalignment or a mispowering. Presuming these are small, the first order will also be small. To be specific, say that a factor of the small parameter ε

multiplies each first-order polynomial. We may now consider how this changes the analysis of the algebra and the corresponding group in the last section.

The polynomial spaces S_i now need to be supplemented. Let S_{-1} be the space of first-order polynomials. The S_i ($i = -1,0,1,\ldots$) is still a grading. However, we can not construct a corresponding complementary filter according to (3.3), because we now have a negative i. Thus the $S^{(i)}$ are no longer ideals and we cannot form the quotient algebra. There is a corresponding destruction of the normal subgroups and quotient groups of symplectic maps.

Instead of using a grading, let us try to create a complementary filter, and thus a series of ideals and a series of quotient algebras, in another way. Let us define a second index j ($j = 0,1,\ldots$) on the S_i that is equal to the ε order. Thus S_{ij} is a subset of S that is homogeneous of order i+2 in the phase space coordinates, and homogeneous of order j in ε. The index i ranges from $-1,0,1,\ldots\infty$, and the index j ranges from $0,1,\ldots,\infty$. I will say that a polynomial (or Lie operator) has δ-rank i and ε-rank j if it belongs to S_{ij} (or S^*_{ij}). The only combination of i and j within these ranges that is prohibited is i= -1, j=0, the smallness requirement on first-order terms discussed above. We now seek a complementary filter constructed from the S_{ij}.

Let $\mathbf{Z}^{2*} = \{-1,0,1,\ldots\} \times \{0,1,\ldots\} - \{(-1,0)\}$, the pairs of allowed coefficients. Let $\mathbf{Z}^+$ be the non-negative integers. Let ν be a function $\nu:\mathbf{Z}^{2*} \to \mathbf{Z}^+$ with the property

$$\nu(i,j) + \nu(k,\ell) \leq \nu(i+k,\ j+\ell). \tag{3.8}$$

Now define the sequence of subspaces $S^{(i)}$, $i \in \mathbf{Z}^+$ by

$$S^{(i)} = \bigoplus_{\nu(j,k) \geq i} S_{jk}. \tag{3.9}$$

This sequence is a complementary filter, which may be verified using the fact

$$A(S_{ij}, S_{k\ell}) \subseteq S_{i+k,j+\ell}. \tag{3.10}$$

Then the product of an element in $S^{(\nu(i,j))}$ with one in $S^{(\nu(k,\ell))}$ is in $S^{(\nu(i+k,j+\ell))}$,

$$A(S^{(\nu(i,j))}, S^{(\nu(k,\ell))}) \subseteq S^{(\nu(i+k) + \nu(j+\ell))} \tag{3.11}$$

$$\subseteq S^{(\nu(i+k,j+\ell))}$$

because of the rule (3) of the definition of the complementary filter, and the relation (3.8).

With this complementary filter, we have a sequence of ideals $S^{(i)}$ in the algebra which may be used to define quotient algebras.

Note that ν is undetermined except for the condition (3.8). Consider two examples, $\nu(i,j) = \min(i,j)$ and $\nu(i,j) = \alpha i + \beta j$ where $\alpha, \beta \in \mathbf{Z}^+$ are constants. The former case corresponds to keeping all terms except those whose δ-rank and whose ε-rank each exceed a certain value. The latter exclude those whose weighted sum exceeds a certain value. This form of ν satisfies a stricter condition than (3.8), in fact

$$\nu(i,j) + \nu(k,\ell) = \nu(i+k,\ j+\ell) \tag{3.12}$$

and so we have a grading $S_{\nu(i,j)}$, which may take a complementary filter by (3.3). This complementary filter is the same as the one defined by (3.9).

Normal subgroups $G^{(\nu(i,j))}$ of the group of symplectic maps G including constant terms may be defined by analogy to the case without constant terms. The quotient group $H^{(\nu(i,j))} = G/G^{(\nu(i,j))}$ is the group of transformations that will actually be used in computations.

The function ν that I shall actually use, and the one that makes the most intuitive sense, is

$$\nu(i,j) = i + j.$$

This will be called the total rank or total order. In terms of ε-rank and δ-rank, this criterion says that we restrict terms to $O(\delta) + O(\varepsilon) \leq N$ for some N, specifically, N=4 for MARYLIE 3.1. For MARYLIE 3.0, where we keep up through and including fourth-order polynomials, we would choose N=4. Physically, this is a realistic criterion, because it means for example that the misaligned element is off-center by an amount that is the same order as the typical deviation of the particles when entering an aligned element. Thus, we can expect the same accuracy in the result. In this calculation, it should not matter whether the magnet is moved or the particles are moved.

4. Concatenation of Factored Maps

The description of an accelerator lattice or section as a transfer map has utility for tracking and for determination of analytic quantities such as tunes and chromaticity. What adds greatly to this utility, however is the ability to combine or concatenate maps to show the effect of two or more sections in succession. Then a library of common beamline elements may be maintained, and for a particular lattice, maps are concatenated to form an overall transfer map. This is an essential part of MARYLIE 3.0 (Dragt [1985] et al., and Douglas [1982]), which contains Lie transformations second through fourth order. This chapter describes an extension to include first-order transformations arising from misalignments or mispowering. How the maps including misalignment and mispowering arise is the subject of Part II; for now, we shall assume they exist.

The task of this chapter is as follows. Consider two maps in the standard factorization that includes a first-order polynomial

$$M_f = e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:}, \tag{4.1a}$$

$$M_g = e^{:g_1:} e^{:g_2:} e^{:g_3:} e^{:g_4:}. \tag{4.1b}$$

We wish to find the polynomials h_n such that

$$M_f M_g = M_h \tag{4.1c}$$

where

$$M_h = e^{:h_1:}e^{:h_2:}e^{:h_3:}e^{:h_4:}. \qquad (4.1d)$$

Additionally, we would like to convert to a descending factorization; given M_f as above what are the h_n such that

$$e^{:h_4:}e^{:h_3:}e^{:h_2:}e^{:h_1:} = M_f = e^{:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:} \qquad (4.2)$$

This computation is divided into two parts: the hard part and the easy part. The hard part is moving the first-order term to the left. This is covered in section a. The easy part is concatenating the transformations second order and higher that are left behind. This is covered in section b.

Both these parts use the same two tools. One tool is the transformation rule,

$$e^{:f:}e^{:g:} = e^{:e^{:f:}g:}e^{:f:} \qquad (4.3)$$

proved in Chapter 1. The second tool is the Baker-Campbell-Hausdorff theorem (BCH) and its inverse, the Zassenhaus formula. The BCH Theorem tells us how to bring the product of exponentials of non-commuting objects into one exponent, and the Zassenhaus formula tells us how to take a single exponential and break it apart into the product of several exponentials. The BCH theorem says that if

$$e^A e^B = e^C \qquad (4.4)$$

then

$$C = A + B + \frac{1}{2}[A,B] + \frac{1}{12}[A,[A,B]] + \frac{1}{12}[B,[B,A]] + \ldots \qquad (4.5)$$

where [,] is the commutator Lie algebra multiplication. The factorization will be changed by first using the BCH formula to combine terms into a single exponent, and then using the Zassenhaus formula to pull them apart in the order desired.

At the end of section b the goal of concatentation (4.1) will have been achieved. In section c, the problem of writing the factorization in descending form (4.2) is addressed; the computation is essentially moving the first-order term to the right and is similar to that of moving a first-order term to the left (section a). Section d shows that when tracking a particular initial condition, the results agree exactly with the first-order term from concatenation, and could serve as an alternate derivation of that term. Finally, section e serves as an appendix showing the details of one of the calculations in section a.

a. Moving the First-Order Term Left

In order to simplify moving the first-order term to the left we may drop the term $e^{:g_2:}e^{:g_3:}e^{:g_4:}$ temporarily and concentrate on writing

$$e^{:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:} \qquad (4.6a)$$

as

$$e^{:h_1':}e^{:h_2^{(1)'}:}\ldots e^{:h_2^{(n)'}:}e^{:h_3':}e^{:h_4':}. \qquad (4.6b)$$

The possibility is left open that there are more than one second-order

(linear) transformations $h_2^{(1)'}$, ..., $h_2^{(n)'}$. It may not be possible to combine them into a single transformation, and in any case it is not necessary, since they are kept as matrices.

The factorization change (4.6) will be performed by successively moving $e^{:g_1:}$ left past each term. First, we shall compute k_n, where

$$e^{:f_4:} e^{:g_1:} = e^{:k_1:} e^{:k_2:} e^{:k_3:} e^{:k_4:} \tag{4.7}$$

Next, leaving behind the k_2, k_3, and k_4 terms, we shall compute m_n where

$$e^{:f_3:} e^{:k_1:} = e^{:m_1:} e^{:m_2:} e^{:m_3:} e^{:m_4:} \tag{4.8}$$

Again leaving behind the higher-order terms, we will find m_1^T such that

$$e^{:f_2:} e^{:m_1:} = e^{:m_1^T:} e^{:f_2:} \tag{4.9}$$

and thus, putting it all back together, (4.6a) will be

$$e^{:f_1:} e^{:m_1^T:} e^{:f_2:} e^{:m_2:} e^{m_3:} e^{:m_4:} e^{:k_2:} e^{:k_3:} e^{:k_4:} \tag{4.10}$$

With the first-order terms on the left, section b will be concerned with continuing the other terms into the form (4.6b)

Before embarking on these calculations, the reader is reminded of the criterion by which we truncate the BCH series. In Chapter 3 it was noted that certain Poisson brackets are to be discarded because their total order, the sum of the δ order (phase space variables) and ε order (count of first-order polynomials), is too high. Thus, the BCH series

(4.5) will be stopped at some point by this criterion. The actual number of terms used varies with each calculation.

i. Moving g_1 past f_4

The first step in concatenating is to get $e^{:g_1:}$ past $e^{:f_4:}$. We want to find k_1, k_2, k_3 and k_4 such that

$$e^{:f_4:}e^{:g_1:} = e^{:k_1:}\, e^{:k_2:}\, e^{:k_3:}\, e^{:k_4:}. \tag{4.11}$$

Note that in general, one would expect in moving g_1 past f_4 that terms of all orders would appear in the standard factorization. Using the transformation rule, we find the result

$$e^{:f_4:}e^{:g_1:} = e^{:g_1:}\, e^{:e^{-:g_1:}f_4:} \tag{4.12}$$

$$= e^{:g_1:}\, e^{:f_4 - :g_1:f_4 + \frac{1}{2}:g_1:^2 f_4 - \frac{1}{6}:g_1:^3 f_4:}.$$

This expression is exact because the next term in the series is a constant. We may say the second exponential on the right side is

$$e^{:j_1+j_2+j_3+j_4:} \tag{4.13}$$

where, with their orders in ε and δ,

$$j_1 = -\frac{1}{6}:g_1:^3 f_4 \quad , \quad O(\delta) = 1 \quad , \quad O(\varepsilon) = 3 \tag{4.14a}$$

$$j_2 = \frac{1}{2}:g_1:^2 f_4 \quad , \quad O(\delta) = 2 \quad , \quad O(\varepsilon) = 2 \tag{4.14b}$$

$$j_3 = -:g_1:f_4 \quad , \quad O(\delta) = 3 \quad , \quad O(\varepsilon) = 1 \tag{4.14c}$$

$$j_4 = f_4 \quad , \quad O(\delta) = 4 \quad , \quad O(\varepsilon) = 0 \tag{4.14d}$$

Note the total order, $O(\delta) + O(\varepsilon)$ is always 4. The next step is to split up this sum in the exponent into a standard factorization of the form

$$e^{:j_1+j_2+j_3+j_4:} = e^{:\ell_1:} e^{k_2:} e^{:k_3:} e^{:k_4:} \tag{4.15}$$

This is accomplished with the Zassenhaus formula: essentially, solve the BCH formula using the unknown quantities ℓ_1 and k_n. First, bring the ℓ_1 to the left

$$e^{-:\ell_1:} e^{:j_1+j_2+j_3+j_4:} = e^{:k_2:} e^{:k_3:} e^{:k_4:} \tag{4.16}$$

then apply the BCH formula to the left side:

$$e^{-:\ell_1:} e^{:j_1+j_2+j_3+j_4:} = e^{:p:} \tag{4.17}$$

where

$$p = -\ell_1 + j_1 + j_3 + j_4 + \frac{1}{2}[-\ell_1, j_1 + j_2 + j_3 + j_4] + \tag{4.18}$$

$$+ \frac{1}{12}[-\ell_1[-\ell_1, j_1 + j_2 + j_3 + j_4]] + \ldots$$

and solve for ℓ_1 iteratively. The BCH series for p has been cut off

because either the order in ℓ_1 becomes too high yielding constants or the total order becomes too high (each j_n is total order 4). Note that p is a polynomial with no terms first-order in δ. Therefore, we may rewrite ℓ_1, as

$$\ell_1 = j_1 + r_1 \tag{4.19}$$

and attempt to solve r_1. So now, in terms of r_1,

$$\begin{aligned} p &= -r_1 + j_2 + j_3 + j_4 + \tfrac{1}{2}\,[-r_1 - j_1, j_1 + j_2 + j_3 + j_4] + \\ &\quad + \tfrac{1}{12}\,[-r_1 - j_1, [-r_1 - j_1, j_1 + j_2 + j_3 + j_4]] + \cdots \\ &= r_1 - \tfrac{1}{2}\,[r_1, j_2] + j_2 - \tfrac{1}{2}\,[r_1, j_3] + j_3 - \tfrac{1}{2}\,[r_1, j_4] + j_4 \\ &\quad + \tfrac{1}{12}\,[r_1, r_1, j_2] + \tfrac{1}{12}\,[r_1, [r_1, j_3]] + \tfrac{1}{12}\,[r_1, [r_1, j_4]] + \cdots \end{aligned} \tag{4.20}$$

Again, each j_n has a total order of 4, so a Poisson bracket of one with another has a total order 6. Since this exceeds the limit of 4 for retention, all the terms that contain nothing but Poisson brackets of j's may be dropped.

Taking the first-order in ε part of both sides of (4.2), we have an implicit equation in r_1,

$$0 = r_1 + \tfrac{1}{2}\,:r_1:j_2 - \tfrac{1}{12}\,:r_1:^2 j_3. \tag{4.21}$$

The only reasonable solution to this is $r_1 = 0$, as is demonstrated in detail in section e of this chapter. Thus, (4.19) gives $\ell_1 = j_1$.

From ℓ_1 we may move on to solve for k_2. The equation to solve, (4.17), is now:

$$e^{-:k_2:} e^{:j_2+j_3+j_4:} = e^{:p:}, \tag{4.22}$$

where p is now different. Applying BCH:

$$p = -k_2 + j_2 + j_3 + j_4 + \frac{1}{2}[-k_2, j_2 + j_3 + j_4] + \ldots \tag{4.23}$$

Let

$$k_2 = j_2 + r_2 \tag{4.24}$$

so that

$$p = -r_2 + j_3 + j_4 - \frac{1}{2}[r_2, j_2 + j_3 + j_4] - \frac{1}{2}[j_2, j_2 + j_3 + j_4] + \ldots \tag{4.25}$$

Again, we find that all Poisson brackets involving nothing but j's may be dropped. The requirement that terms of p second-order in δ be zero dictates that $r_2=0$.

Next consider k_3. We are left with, in (4.16)

$$e^{:k_3:} e^{:j_3+j_4:} = e^{:k_4:} \tag{4.26}$$

or

$$k_4 = -k_3 + j_3 + j_4 + \frac{1}{2}[-k_3, j_3 + j_4] \ldots \tag{4.27}$$

Let

$$k_3 = j_3 + r_3 \tag{4.28}$$

and by the reasoning above,

$$r_3 = 0 \quad , \quad k_4 = j_4. \tag{4.29}$$

In summary,

$$e^{:f_4:} e^{:g_1:} = e^{:g_1:} e^{:j_1+j_2+j_3+j_4:} \tag{4.30}$$

$$= e^{:g_1:} e^{:\ell_1:} e^{:k_2:} e^{:k_3:} e^{:k_4:}$$

where

$$\ell_1 = -\frac{1}{6} :g_1:^3 f_4 \tag{4.31a}$$

$$k_2 = \frac{1}{2} :g_1:^2 f_4 \tag{4.31b}$$

$$k_3 = -:g_1:f_4 \tag{4.31c}$$

$$k_4 = f_4 \tag{4.31d}$$

to total order 4. Since the Poisson bracket of two first order operators is a constant, g_1 and ℓ_1 may be combined to form k_1 so that

$$k_1 = g_1 + \ell_1 = g_1 - \frac{1}{6} :g_1:^3 f_4. \tag{4.32}$$

Thus we have a solution to (4.11)

ii. Moving k_1 past f_3

The terms second and higher order present no problems in concatenating; they will be dealt with later. We thus may leave them behind temporarily and concentrate on moving the first-order term past the next obstacle, $e^{:f_3:}$;

$$e^{:f_3:}\, e^{:k_1:} = e^{:n_1:}\, e^{:n_2:}\, e^{:n_3:}\, e^{:n_4:}, \tag{4.33}$$

analagous to (4.8). We shall see that the fourth-order term n_4 does not appear.

As with the f_4, we use the transformation property to obtain

$$e^{:f_3:}\, e^{:k_1:} = e^{:k_1:}\, e^{:e^{-:k_1:} f_3:} \tag{4.34}$$

$$= e^{:k_1:}\, e^{:f_3 - :k_1:f_3 + \frac{1}{2}:k_1:^2 f_3:},$$

which is exact. We represent the second exponential as

$$e^{:m_1+m_2+m_3:} \tag{4.35}$$

where

$$m_1 = \tfrac{1}{2}\,:k_1:^2 f_3 \quad , \quad O(\delta) = 1 \quad , \quad O(\varepsilon) = 2 \tag{4.36a}$$

$$m_2 = - :k_1:f_3 \quad , \quad O(\delta) = 2 \quad , \quad O(\varepsilon) = 1 \tag{4.36b}$$

$$m_3 = f_3 \quad , \quad O(\delta) = 3 \quad , \quad O(\varepsilon) = 0. \tag{4.36c}$$

The total order in all three cases is now 3 instead of 4. This makes the calculation more interesting; we will obtain one Poisson bracket before the series is terminated due to excessive total order. To get the expression (4.35) to factored form,

$$e^{:m_1+m_2+m_3:} = e^{:n_1:} e^{:n_2:} e^{:n_3:} e^{:n_4:}, \tag{4.37}$$

the procedure is as before. We solve implicitly for n_1:

$$e^{-:n_1:} e^{:m_1+m_2+m_3:} = e^{:p:} = e^{:n_2:} e^{:n_3:} e^{:n_4:}, \tag{4.38}$$

where p is not as before. Knowing that p should have no first-order term, we may solve (4.38) for n_1. Using BCH, this is

$$p = -n_1 + m_1 + m_2 + m_3 + \frac{1}{2}[-n_1, m_1 + m_2 \quad m_3] \tag{4.39}$$

$$+ \frac{1}{12}[-n_1, [-n_1, m_1 + m_2 + m_3]]$$

$$\frac{1}{12}[m_1 + m_2 + m_3, [m_1 + m_2 + m_3, -n_1]] + \ldots$$

In the first step we take, with a new r_1 but analogous to (4.19),

$$n_1 = m_1 + r_1 \tag{4.40}$$

so we now must solve

$$p = -r_1 + m_2 + m_3 + \frac{1}{2}[-m_1, m_1 + m_2 + m_3] + \frac{1}{2}[-r_1, m_1 + m_2 + m_3]$$

$$+ \frac{1}{12}[-m_1 - r_1, [-m_1 - r_1, m_1 + m_2 + m_3]] \qquad (4.41)$$

$$+ \frac{1}{12}[m_1 + m_2 + m_3, [m_1 + m_2 + m_3, -m_1 - r_1]] + \ldots$$

for r_1. The fourth term of this,

$$\frac{1}{2}[-m_1, m_1 + m_2 + m_3] = -\frac{1}{2}[m_1, m_2] - \frac{1}{2}[m_1, m_3], \qquad (4.42)$$

is of total order 4, since each argument of each Poisson bracket is of total order 3. However, brackets involving 3 or more m's may be dropped because the total order will exceed 4. So

$$p = -r_1 - \frac{1}{2}[m_1, m_2] + m_2 - \frac{1}{2}[m_1, m_3] + m_3 \qquad (4.43)$$

$$- \frac{1}{2}[r_1, m_1 + m_2 + m_3]$$

$$- \frac{1}{12}[r_1, [m_1, m_1 + m_2 + m_3]] - \frac{1}{12}[m_1, [r_1, m_1 + m_2 + m_3]]$$

$$- \frac{1}{12}[r_1, [r_1, m_1 + m_2 + m_3]]$$

$$- \frac{1}{12}[m_1 + m_2 + m_3, [m_1 + m_2 + m_3, r_1]] + \ldots$$

To cancel off the first-order part again, we choose

$$r_1 = -\frac{1}{2}[m_1,m_2] + s_1 \tag{4.44}$$

where s_1 is yet to be found, and the term $[m_1,m_2]$ is of total order 4. Note that all terms in the nested Poisson brackets involving this term can be taken as zero. To get the first-order part of p to zero, we may self-consistently take $s_1 = 0$. Then

$$p = m_2 - \frac{1}{2}[m_1,m_3] + m_3 \tag{4.45}$$

and

$$n_1 = m_1 - \frac{1}{2}[m_1,m_2]. \tag{4.46}$$

The next step is to solve for n_2 using the result for p, (4.45) and (4.38):

$$e^{-:n_2:}\, e^{:m_2+m_3-\frac{1}{2}[m_1,m_3]:} = e^{:n_3:} e^{:n_4:} = e^{:q:}. \tag{4.47}$$

Using BCH, this must be solved so that the second-order part of q is zero.

$$q = -n_2 + m_2 + m_3 - \frac{1}{2}[m_1,m_3] \tag{4.48}$$

$$+ \frac{1}{2}[-n_2, m_2 + m_3 - \frac{1}{2}[m_1,m_3]] + \cdots$$

Say

$$n_2 = m_2 - \frac{1}{2}[m_1,m_3] + r_2 \tag{4.49}$$

with r_2 to be determined, then

$$q = m_3 - \frac{1}{2}[r_2,m_2 + m_3 - \frac{1}{2}[m_1,m_3]] \tag{4.50}$$

$$+ \frac{1}{2}[-m_2,m_2 + m_3 - \frac{1}{2}[m_1,m_3]] + \cdots$$

$$= m_3 - \frac{1}{2}[r_2,m_2 + m_3 - \frac{1}{2}[m_1,m_3]] - \frac{1}{2}[m_2,m_3],$$

where terms of excessive total order have been dropped. To get rid of the terms second order in δ we may take $r_2 = 0$ to yield

$$q = m_3 - \frac{1}{2}[m_2,m_3] \tag{4.51}$$

and

$$n_2 = m_2 - \frac{1}{2}[m_1,m_3]. \tag{4.52}$$

Finally, given (4.51), the right-hand of (4.47),

$$e^{:n_3:}e^{:n_4:} = e^{:q:} \tag{4.53}$$

is solved by

$$n_3 = m_3 - \frac{1}{2}[m_2,m_3], \tag{4.54}$$

$$n_4 = 0. \tag{4.55}$$

In summary

$$e^{:f_3:}\, e^{:k_1:} = e^{:k_1:}\, e^{:m_1 - \frac{1}{2}[m_1,m_2]:}\, e^{:m_2 - \frac{1}{2}[m_1,m_3]:}\, e^{:m_3 - \frac{1}{2}[m_2,m_3]:} \tag{4.56}$$

where

$$m_1 = \frac{1}{2} :k_1:^2 f_3 \tag{4.57a}$$

$$m_2 = - :k_1: f_3 \tag{4.57b}$$

$$m_3 = f_3 \tag{4.57c}$$

Again we may join the first-order terms

$$e^{:k_1:}\, e^{:m_1 - \frac{1}{2}[m_1,m_2]:} = e^{:k_1 + m_1 - \frac{1}{2}[m_1,m_2]:} \tag{4.58}$$

so (4.33) is solved by

$$n_1 = k_1 + m_1 - \frac{1}{2}[m_1,m_2] \tag{4.59a}$$

$$n_2 = m_2 - \frac{1}{2}[m_1,m_3] \tag{4.59b}$$

$$n_3 = m_3 - \frac{1}{2}[m_2,m_3] \tag{4.59c}$$

$$n_4 = 0 \tag{4.59d}$$

iii. Moving n_1 past f_2

Once again, we may concentrate on moving the first-order term and leave the rest for later. The next term to pass is $e^{:f_2:}$. This is easy to overcome, for we may apply the transformation rule in a straightfoward manner. It is

$$e^{:f_2:} e^{:n_1:} = e^{:n_1^T:} e^{:f_2:} \tag{4.60}$$

where T indicates the transformed polynomial:

$$n_1^T = n_1(e^{:f_2:}\cdot). \tag{4.61}$$

iv. Picking up the Pieces

The final step is to concatenate the terms second order and higher that we have left behind. The expression we started with, (4.6a), now looks like

$$e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:} e^{:g_1:} \tag{4.62}$$

$$= e^{:f_1:} e^{:n_1^T:} e^{:f_2:} e^{:n_2:} e^{:n_3:} e^{:k_2:} e^{:k_3:} e^{:k_4:} \tag{4.63}$$

The new terms n_1^T, n_2, n_3, k_2, k_3, and k_4 have been calculated above.

First, n_1^T is, using (4.61), (4.59a), (4.57) and (4.32),

$$n_1^T = (k_1 + m_1 - \tfrac{1}{2}\,[m_1,m_2])^T \tag{4.63}$$

$$= (k_1 + \tfrac{1}{2}\,:k_1:^2 f_3 - \tfrac{1}{2}\,[\tfrac{1}{2}\,:k_1:^2 f_3,\ -:k_1:f_3)]^T$$

$$= (g_1 - \tfrac{1}{6}\,:g_1:^3 f_4 + \tfrac{1}{2}\,:g_1:^2 f_3 + \tfrac{1}{4}\,[:g_1:^2 f_3,:g_1:f_3])^T$$

$$= e^{:f_2:}(g_1 - \tfrac{1}{6}\,:g_1:^3 f_4 + \tfrac{1}{2}\,:g_1:^2 f_3 + \tfrac{1}{4}\,[:g_1:^2 f_3,:g_1:f_2])$$

The quantity $\ell_1 = g_1 - \frac{1}{6}\,:g_1:^3 f_4$ may be taken as g_1 in all but the first term, because the total order is too high in all the other terms. This is also done in computing n_2, using (4.59b), (4.57), and (4.32),

$$n_2 = m_2 - \tfrac{1}{2}\,[m_1,m_3] = -\,:k_1:f_3 - \tfrac{1}{2}\,[\tfrac{1}{2}\,:k_1:^2 f_3,f_3] \tag{4.64}$$

$$= -\,:g_1:f_3 - \tfrac{1}{4}\,[:g_1:^2 f_3,f_3].$$

The polynomial n_3 is found from (4.59c), (4.57), and (4.32)

$$n_3 = m_3 - \tfrac{1}{2}\,[m_2,m_3] = f_3 - \tfrac{1}{2}\,[-\,:\ell_1:f_3,f_3] = f_3 + \tfrac{1}{2}\,[:g_1:f_3,f_3] \tag{4.65}$$

$$= f_3 + \tfrac{1}{2}\,:f_3:^2 g_1.$$

The polynomials k_2 and k_3 are given in (4.31b) and (4.31c)

$$k_2 = \tfrac{1}{2}\,:g_1:^2 f_4, \tag{4.31b}$$

$$k_3 = - :g_1:f_4. \tag{4.31c}$$

To put these transformations in ascending order, we note the terms involving n_3 and k_2 are in the wrong order, so use the transformation rule to write

$$e^{:n_3:}e^{:k_2:} = e^{:k_2:}e^{e^{-:k_2:}n_3:} \tag{4.66}$$

$$= e^{:k_2:}e^{:n_3:}$$

because k_2 has total order 4, $e^{-:k_2:}$ acts as the identity on n_3, so (4.62) becomes

$$e^{:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:} \tag{4.67}$$

$$= e^{:f_1:}e^{:n_1^T:}e^{:f_2:}e^{:n_2:}e^{:k_2:}e^{:n_3:}e^{:k_3:}e^{:f_4:}$$

We may now combine some terms of like order as follows. The first-order terms f_1 and n_1^T may be combined into a single exponent

$$e^{:f_1:}e^{:n_1^T:} = e^{:f_1+ n_1^T:} \tag{4.68}$$

because their Poisson bracket is a constant, so all Poisson bracket terms in their BCH formula are zero. The second-order terms d_2 and c_2 may be combined

$$e^{:n_2:}e^{:k_2:} = e^{:n_2+k_2:} \tag{4.69}$$

because d_2 has total order 3 and k_2 has total order 4, so a Poisson bracket will have total order 5, higher than the cut-off of 4. Thus any Poisson bracket terms in the BCH series may be taken to be zero. A similar argument allows us to combine the third-order terms

$$e^{:d_3:}e^{:k_3:} = e^{:d_3+k_3:} \tag{4.70}$$

Then

$$e^{:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:} = e^{:f_1+n_1^T:}e^{:n_2+k_2:}e^{:n_3+k_3:}e^{:f_4:}. \tag{4.71}$$

To summarize with the f's and g's in place, (4.6) is

$$e^{:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:} \tag{4.72}$$

$$= e^{:f_1+e^{:f_2:}(g_1 + \frac{1}{2}:g_1:^2f_3 - \frac{1}{6}:g_1:^3f_4 - \frac{1}{4}[:g_1:f_3,:g_1:^2f_3]):}$$

$$e^{:f_2:}e^{:-:g_1:f_3 + \frac{1}{4}[f_3,:g_1:^2f_3]: + \frac{1}{2}:g_1:^2f_4:}$$

$$e^{:f_3 + \frac{1}{2}:f_3:^2g_1 + :f_4:g_1:}e^{:f_4:}.$$

b. Concatenation of Terms Second-Order and Higher

With g_1 taken care of, we may now concatenate with the transformations $e^{:g_2:}e^{:g_3:}e^{:g_4:}$. Fortunately, these are easier. We have

$$e^{:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:}e^{:g_2:}e^{:g_3:}e^{:g_4:} \tag{4.73}$$

$$= e^{:h_1':}\, e^{:h_2':}\, e^{:h_3':}\, e^{:f_4:}\, e^{:g_2:}\, e^{:g_3:}\, e^{:g_4:}$$

By the transformation rule, the term $e^{:g_2:}$ may be brought to the left.

$$e^{:h_1':}\, e^{:h_2^{(1)'}:}\, e^{:h_2^{(2)'}:}\, e^{:g_2:}\, e^{:f_3'^T:}\, e^{:f_4^T:}\, e^{:g_3:}\, e^{:g_4:} \tag{4.74}$$

where the superscript T now indicates that the polynomial is to be transformed by $e^{-:g_2:}$:

$$p^T(\zeta) = p(e^{-:g_2:}\zeta). \tag{4.75}$$

We now concentrate on the third and fourth order terms. We may make the exchange

$$e^{:f_4^T:}\, e^{:g_3:} = e^{:g_3:}\, e^{:f_4^T:} \tag{4.76}$$

because the Poisson bracket of a third-order polynomial and a fourth-order polynomial is a fifth-order polynomial, which we are excluding. There are now two adjacent third-order terms and two adjacent fourth-order terms:

$$e^{:h_3'^T:}\, e^{:g_3:}\, e^{:f_4^T:}\, e^{:g_4:}. \tag{4.77}$$

The first two terms may be combined using the BCH formula

$$e^{:h_3'^T:}\, e^{:g_3:} = e^{:h_3'^T+g_3+\frac{1}{2}[h_3'^T,g_3]:} \tag{4.78}$$

$$= e^{:h_3'^T + g_3:} e^{:\frac{1}{2}[h_3'^T, g_3]:},$$

where the fourth-order term formed by the Poisson bracket of the third order term may be put in a separate exponential, because all higher terms are being neglected. For the same reason, all fourth-order terms may be combined in a single exponential, giving

$$e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:} e^{:g_1:} e^{:g_2:} e^{:g_3:} e^{:g_4:} \tag{4.79}$$

$$= e^{:h_1':} e^{:h_2^{(1)}:} e^{:h_2^{(2)}:} e^{:g_2:} e^{:h_3'^T + g_3:} e^{:\frac{1}{2}[h_3'^T, g_3] + h_4' + g_4:}$$

Thus, the answer to the original question (4.1) is, to the proper total order

$$h_1 = f_1 + e^{:f_2:}(g_1 - \tfrac{1}{6}:g_1:^3 f_4 + \tfrac{1}{2}:g_1:^2 f_3 + \tfrac{1}{4}[:g_1:^2 f_3, :g_1:f_3]) \tag{4.80a}$$

$$h_2^{(1)} = f_2 \tag{4.80b}$$

$$h_2^{(2)} = \tfrac{1}{2}:g_1:^2 f_4 - :g_1:f_3 - \tfrac{1}{4}[:g_1:^2 f_3, f_3] \tag{4.80c}$$

$$h_2^{(3)} = g_2 \tag{4.80d}$$

$$h_3 = g_3 + e^{-:g_2:}(-:g_1:f_4 + f_3 + \tfrac{1}{2}:f_3:^2 g_1) \tag{4.80e}$$

$$h_4 = \tfrac{1}{2}[e^{-:g_2:}(-:g_1 f_4 + f_3 + \tfrac{1}{2}:f_3:^2), g_3] + e^{-:g_2:} f_4 + g_4. \tag{4.80f}$$

c. Factorization in Descending Order and Inversion

In addition to concatenation of maps, the BCH and Zassenhaus formulas may be used to find a map factorization in descending order given its factorization in ascending order, that is, to find the h_n such that

$$e^{:f_1:}\, e^{:f_2:}\, e^{:f_3:}\, e^{:f_4:} = e^{:h_4:}\, e^{:h_3:}\, e^{:h_2^{(1)}:} \cdots e^{:h_2^{(n)}:}\, e^{:h_1:} \tag{4.81}$$

This is useful for inverting a map: we first reverse the factorization, and then change the sign of all polynomials.

As with the concatenation, we concentrate first on moving the term $e^{:f_1:}$ only this time to the right instead of the left. By a derivation almost identical to that of moving the first-order form left (section b), we obtain

$$e^{:f_1:}\, e^{:f_2:}\, e^{:f_3:}\, e^{:f_4:} = e^{:f_2:}\, e^{::f_1^T:f_3 - \frac{1}{4}[f_3,:f_1^T:f_3] + \frac{1}{2}:f_1^T:^2 f_4:} \tag{4.82}$$

$$e^{:f_3 - \frac{1}{2}:f_3:^2 f_1^T - :f_1^T:f_4:}\, e^{:f_4:}$$

$$e^{:f_1^T + \frac{1}{2}: f_1^T:^2 f_3 + \frac{1}{6}:f_1^T:^3 f_4 - \frac{1}{6}[:f_1^T:f_3,:f_1^T:^2 f_3]:}$$

where

$$f_1^T(\zeta) = f_1(e^{-:f_2:}\zeta) \tag{4.83}$$

By using the transformation rule with the two second-order transforma-

tions, we obtain

$$h_1 = f_1^T + \frac{1}{2} + :f_1^T: f_3 + \frac{1}{6} :f_1^T:^3 f_4 - \frac{1}{6} [:f_1^T:f_3, :f_1^T:^2 f_3] \quad (4.84a)$$

$$h_2^{(1)} = f_2 \quad (4.84b)$$

$$h_2^{(2)} = :f_1^T:f_3 - \frac{1}{4} [f_3, :f_1^T:f_3] + \frac{1}{2} :f_1^T:^2 f_4 \quad (4.84c)$$

$$h_3 = (f_3 - \frac{1}{2} :f_3:^2 f_1^T - :f_1^T:f_4)^t \quad (4.84d)$$

$$h_4 = f_4^t , \quad (4.84e)$$

where the superscript t indicates the transformation of the polynomial

$$p^t(\zeta) = p(e^{:h_2^{(1)}:} e^{:h_2^{(2)}:} \zeta). \quad (4.85)$$

Note that these formulas are also useful in concatenation of maps factorized in descending order, or in ascending order except with the first-order term on the right.

d. Relation to Ray Tracing

An alternative process may be used to determine the concatenation formula for h_1, or to understand why certain tracking results calculated using either a concatenated map or using the maps separately agree exactly (see Appendix B).

To simplify the problem, take $f_1 = g_2 = g_3 = g_4 = 0$, so that

$$e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:} = e^{:h_1:}e^{:h_2^{(1)}:}e^{:h_2^{(2)}:}e^{:h_3:}e^{:h_4:} \tag{4.86}$$

with

$$h_1 = e^{:f_2:}(g_1 - \tfrac{1}{6}:g_1:^3 f_4 + \tfrac{1}{2}:g_1:^2 f_3 + \tfrac{1}{4}[:g_1:^2 f_3, :g_1:f_3]), \tag{4.87a}$$

$$h_2^{(1)} = f_2, \tag{4.87b}$$

$$h_2^{(2)} = \tfrac{1}{2}:g_1:f_4 - :g_1:f_3 - \tfrac{1}{4}[:g_1:^2 f_3, f_3], \tag{4.87c}$$

$$h_3 = -:g_1:f_4 + f_3 + \tfrac{1}{2}:f_3:^2 g_1, \tag{4.87d}$$

$$h_4 = f_4. \tag{4.87e}$$

Consider the phase space points

$$v_b = e^{-:h_1:} \cdot |_0 = [-h_1, \cdot] \tag{4.88}$$

and

$$v_a = e^{:g_1:} \cdot |_0 = [g_1, \cdot]. \tag{4.89}$$

Using the formula for h_1 above, I shall show that $M_f v_b = -v_a$ exactly when M_f is the map expanded for non-symplectified tracking (see Chapter 2).

In order to see that both $M_f v_b$ and $-v_a$ are analytically the same when tracked nonsymplectically, rephrase the equation as

$$v_b = M_f^{-1}\,(-v_a) \tag{4.90}$$

or

$$e^{-:f_4:}\,e^{-:f_3:}\,e^{-:f_2:}\,\cdot\,|_{-v_a} = v_b = [-h_1,\cdot] \tag{4.91}$$

$$e^{-:g_1:}\,e^{-:f_4:}\,e^{-f_3:}\,e^{-f_2:}\,\cdot\,|_0 = -\,[h_1,\cdot] \tag{4.92}$$

$$e^{-:g_1:}\,e^{-:f_2:}\,e^{-:f_4^T:}\,e^{-:f_3^T:}\,\cdot\,|_0 = -\,[h_1,\cdot] \tag{4.93}$$

where

$$f_n^T = f_n(e^{:f_2:}\,\cdot) \quad , \quad n = 3,4. \tag{4.94}$$

We may exchange f_3^T and f_4^T since we are keeping only terms up through fourth order

$$e^{-:g_1:}\,e^{-:f_2:}\,e^{-:f_3^T:}\,e^{-:f_4^T:}\,\cdot\,|_0 = -\,[h_1,\,\cdot] \tag{4.95}$$

or

$$e^{-:f_2:}\,e^{-:g_1^T:}\,e^{-:f_3^T:}\,e^{-:f_4^T:}\,\cdot\,|_0 = -\,[h_1,\,\cdot] \tag{4.96}$$

$$e^{-:f_3^T:}\,e^{-:f_4^T:}\,\cdot\,|_{-\,[g_1^T,\cdot]} = -\,[h_1,\,\cdot]. \tag{4.97}$$

The expansion of the Lie transformation exponentials non-symplectically through fourth order gives, with h_1 given by (4.87a)

$$(\cdot - [f_3^T, \cdot] + \tfrac{1}{2}\,[f_3^T,[f_3^T, \cdot]] - [f_4^T, \cdot])\,\Big|_{-\,[g_1^T,\cdot]} \tag{4.98}$$

$$= [-\,g_1^T - \tfrac{1}{6}\,{:}g_1^T{:}^2\, f_3^T + \tfrac{1}{4}\,[[g_1^T,f_3^T],[g_1^T,g_1^T,f_3^T]] + \tfrac{1}{6}\,{:}g_1^T{:}^3 f_4^T\,,\,\cdot]$$

I shall now show that, term by term, the two sides are equal. The first terms are obviously the same, $-\,[g_1^T, \cdot]$.

To show the equality of the remaining terms, let

$$g_1^T = g_X X + g_{P_X} P_X + \ldots + g_{P_T} P_T \tag{4.99}$$

and

$$g \equiv \begin{bmatrix} g_X \\ g_{P_X} \\ \vdots \\ g_{P_T} \end{bmatrix}. \tag{4.100}$$

Let $\partial_i f \equiv \frac{\partial f}{\partial \zeta_i}$. Then $[g_1^T, \cdot] = g \cdot J$ and $[g_1^T,f] = \sum_i (g \cdot J)_i\, \partial_i f$. Now consider the second term on the right side of (4.98), acting on the t^{th} phase space variable, ζ_t:

$$[-\,\tfrac{1}{2}\,{:}g_1^T{:}^2\, f_3^T, \zeta_t] = -\,\tfrac{1}{2}\,[[g_1^T,[g_1^T,f_3^T]],\zeta_t] \tag{4.101}$$

$$= -\,\tfrac{1}{2}\,[[g_1^T, \textstyle\sum_i (g \cdot J)_i\, \partial_i f_3^T], \zeta_t]$$

$$= -\,\tfrac{1}{2} \sum_{i,j} [(g \cdot J)_i (g \cdot J)_j\, \partial_i \partial_j f_3^T, \zeta_t]$$

$$= -\,\tfrac{1}{2} \sum_{i,j} (g \cdot J)_i (g \cdot J)_j\, (\partial_s \partial_i \partial_j f_3^T)\, J_{st}$$

$$= -\frac{1}{2} \sum_{i,j} \zeta_i \zeta_j \, (\partial_i \partial_j \partial_s f_3^T) \, J_{st} \, |_{\zeta = g \cdot J}$$

$$= -\sum \, (\partial_s f_3^T(\zeta)) \, J_{st} |_{\zeta = g \cdot J} = - \, [f_3^T, \zeta_t] |_{\zeta = g \cdot J}$$

where use has been made of Taylor's theorem on the homogeneous second-order polynomial $\partial_i f_3^T$. Since this is true for all components t of ζ, we may say

$$[-\frac{1}{2} \, :g_1^T:^2 \, f_3^T, \, \cdot] = - \, [f_3^T, \, \cdot] \, |_{[-g_1^T, \cdot]} \tag{4.102}$$

The third terms of (4.98) may be shown equal by a similar process:

$$\frac{1}{4} \, [[[g_1^T, f_3^T], \, [g_1^T, [g_1^T, f_3^T]]], \, \zeta_t] \tag{4.103}$$

$$= \frac{1}{4} \sum_{\substack{i,j,k, \\ m,n,s}} \partial_s \{(g \cdot J)_i (\partial_m \partial_i f_3^T) \, J_{mn} \, (g \cdot J)_j (g \cdot J)_k \, (\partial_n \partial_j \partial_k f_3^T)\} \, J_{st}$$

$$= \frac{1}{4} \sum_{\substack{i,j,k \\ m,n,s}} (g \cdot J)_i (\partial_s \partial_m \partial_i f_3^T) \, J_{mn} \, (g \cdot J)_j (g \cdot J)_k \, (\partial_n \partial_j \partial_k f_3^T) \, J_{st}$$

$$= \frac{1}{4} \sum \zeta_i (\partial_i \partial_j \partial_s f_3^T) \, J_{mn} \, \zeta_j \zeta_k (\partial_j \partial_k \partial_n f_3^T) \, J_{st} \, |_{\zeta = g \cdot J}$$

$$= \frac{1}{2} \sum \partial_m \partial_s f_3^T \, (\zeta) \, J_{mn} \, \partial_n f_3^T \, (\zeta) \, J_{st} \, |_{\zeta = g \cdot J}$$

$$= \frac{1}{2} \sum \, [\partial_s f_3^T(\zeta), \, f_3^T(\zeta)] \, J_{st} \, |_{\zeta = g \cdot J}$$

$$= -\frac{1}{2} \sum \, [f_3^T, \partial_s f_3^T] \, J_{st} \, |_{g \cdot J}$$

$$= -\frac{1}{2} \, [f_3^T, [f_3^T, \zeta_t]] \, |_{\zeta = g \cdot J},$$

where the second step occurs because $\partial_n \partial_j \partial_k f_3^T$ is a constant. We may conclude the third terms are equal,

$$\frac{1}{4}\,[[[g_1^T, f_3^T], [g_1^T, [g_1^T, f_3^T]], \cdot] = \frac{1}{2}\,[f_3^T, [f_3^T, \cdot]]\,\Big|_{-[g_1^T, \cdot]} . \qquad (4.104)$$

Finally, the fourth term on the right side of (4.98) is

$$\frac{1}{6}\,[:g_1^T:^3 f_4^T, \zeta_t] = \frac{1}{6} \sum_{i,j,k,s} (g \cdot J)_i (g \cdot J)_j (\partial_i \partial_j \partial_k \partial_s f_4^T)\, J_{st} \qquad (4.105)$$

$$= \frac{1}{6} \sum \zeta_i \zeta_j \zeta_k (\partial_i \partial_j \partial_k \partial_s f_4^T)\, J_{st}\,\Big|_{\zeta = g \cdot J}$$

$$= \sum \partial_s f_4(\zeta)\, J_{st}\,\Big|_{\zeta = g \cdot J} = [f_4, \zeta_t]\,\Big|_{\zeta = g \cdot J}$$

so

$$\frac{1}{6}\,[:g_1^T:^3 f_4^T, \cdot] = -\,[f_4, \cdot]\,\Big|_{-[g_1^T, \cdot]} . \qquad (4.106)$$

Thus, term by term, the two sides of (4.95) are equal. Clearly, had we not known h_1, we could have run each of these calculations backward to obtain it, and we could do so for higher orders, since the Taylor expansion yields an easy sequence of terms for the left side.

e. Uniqueness of the Solution for r_1

In section a, we needed to solve equation (4.21) for r_1:

$$0 = r_1 + \frac{1}{2}\,:r_1:j_2 - \frac{1}{12}\,:r_1:^2 j_3 . \qquad (4.21)$$

The purpose of this section is to explore the possible solutions of this equation.

Although an obvious solution may immediately occur to the reader, and this is in fact correct, it is worthwhile seeing that this is the only solution. First, we shall assume that r_1 is of order at least one in ε. This is necesssary to be consistent with the previous truncation of the BCH series, and is crucial to give a unique answer.

Let us investigate this equation by taking a particular example for the polynomials. Suppose

$$f_4 = -3XP_X^3 \tag{4.107a}$$

$$g_1 = \frac{\varepsilon}{3} X \tag{4.107b}$$

so that

$$j_4 = f_4 = -3XP_X^3 \tag{4.108a}$$

$$j_3 = -:g_1:f_4 = 3\varepsilon XP_X^2 \tag{4.108b}$$

$$j_2 = \frac{1}{2} :g_1:^2 f_4 = -\varepsilon^2 XP_X \tag{4.108c}$$

$$j_1 = -\frac{1}{6} :g_1:^3 f_4 = \frac{\varepsilon^3}{9} X. \tag{4.108d}$$

Now consider the possible solution $r_1 = \kappa X$, where κ is some, as yet undetermined, real constant. The second term of (4.21) becomes

$$\frac{1}{2} :r_1:j_2 = \frac{1}{2} [\kappa X, \varepsilon^2 X P_X] = -\frac{1}{2} \kappa \varepsilon^2 X \tag{4.109}$$

and the third term becomes

$$\frac{1}{12} :r_1:^2 j_3 = \frac{1}{12} [\kappa X, [\kappa X, 3\varepsilon\, X P_X^2]] \tag{4.110}$$

$$= \frac{1}{12} [\kappa X, 6\kappa\varepsilon X P_X] = \frac{1}{2} \kappa^2 \varepsilon X.$$

Then the right-hand side of (4.21) is

$$r_1 + \frac{1}{2} :r_1:j_2 - \frac{1}{12} :r_1:^2 j_3 = \kappa X - \frac{1}{2} \kappa \varepsilon^2 X - \frac{1}{2} \kappa^2 \varepsilon X \tag{4.111}$$

$$= (1 - \frac{1}{2} \varepsilon^2 - \frac{1}{2} \kappa\varepsilon)\, \kappa X.$$

In order for this to be 0, we demand either

$$\kappa = 0 \tag{4.112a}$$

or

$$1 - \frac{1}{2} \varepsilon^2 - \frac{1}{2} \kappa\varepsilon = 0. \tag{4.112b}$$

The former gives $r_1 = 0$ and is the obvious solution spoken of above. The latter has the solution

$$\kappa = \frac{2}{\varepsilon} - \varepsilon, \tag{4.113}$$

which gives $r_1 = (\frac{2}{\varepsilon} - \varepsilon)X$. This solution however, is prohibited by our assumption: it contains a term proportional to ε^{-1}, and we assumed that r_1 would only have terms of first order and higher in ε. Not making this assumption would clearly negate the BCH-series terminating benefits of regulating the first-order terms with a small parameter ε.

All possible solutions for r_1 will either be 0, or will be ill behaved with respect to ε. Thus we are forced to take $r_1 = 0$ and discard the alternate solution. This provides us with a unique answer, which we expect because we expect a unique final answer.

The other equations similar to (4.21) in section 1, such as (4.43) with the value of r_1 given by (4.44), will have similar solutions.

5. Symplectification of Matrices

As shown in Chapter 4, a second-order map is created when transporting g_1 past other maps. Unlike the other linear transformations (second-order operators) of MARYLIE, this one is not born symplectic. In treating $e^{:f_2:}$ we actually compute, store and manipulate it as a matrix on phase space variables. These matrices when computed and stored are symplectic to machine precision. When more than one of these has to be concatenated, conventional matrix multiplication insures that the result will also be symplectic. In this case, however, we generate a second-order operator which is to be exponentiated. The δ-order of the terms does not provide a stopping point in the Taylor expansion of the exponential, and truncation of the series may leave a non-symplectic result.

There are several ways to deal with this problem. In each case, the operators are first made into matrices on phase space, so that instead of dealing with $:f_2:$ we are dealing with a corresponding matrix JS, where S is a symmetric matrix, as shown in part 1f. A first and most obvious method is to carry the Taylor series of the exponential to a point where additional terms are beyond machine precision. The advantage of this method is that it is not ony exactly symplectic, but exactly correct. The disadvantage is that it may require too much computation.

A second method is an iterative one of Furman [1985]. Recall the definition that a matrix M is a symplectic if $MJ\tilde{M} = J$, where $\tilde{M}$ is the transpose of M. Equivalently, $-MJ\tilde{M}J = I$ where I is the identity. If we study the matrix function $F(M) = -I-MJ\tilde{M}J$, we note that its deviation

from the zero matrix indicates how far M is from symplecticity. We then seek a correction matrix C such that M'=C M is more symplectic than M, based on the value of F(M). There is some arbitrariness in the choice of C within this constraint. One choice is $C=(I+F(M))^{-1/2}$. We can approximate for F(M) small, $C \approx I - F(M)/2$. Then $M' = \frac{1}{2}(3+MJ\tilde{M}J)M$. With this approximation for C, the matrix M' will not be exactly symplectic, but will be closer to being symplectic than M.

Specifically, I shall show that $\|F(M')\| \leq \|F(M)\|$ for any suitable matrix norm where $\|I\| = 1$. Let M be nearly symplectic, its deviation from symplecticity measured by $\|F(M)\|$, with

$$0 \leq \|F(M)\| << 1. \tag{5.1}$$

Now let $M' = (I - \frac{1}{2} F(M))\, M$. The deviation from symplecticity is

$$F(M') = -I - M'J\tilde{M}'J \tag{5.2}$$

$$= -I - (I - \tfrac{1}{2} F(M))\, MJ\tilde{M}\, (I - \tfrac{1}{2} F(M))\, J$$

$$= -I - MJ\tilde{M}J + \tfrac{1}{2} F(M)\, MJ\tilde{M}J + \tfrac{1}{2} MJ\tilde{M}\, F(M)\, J - \tfrac{1}{4} F(M)\, MJ\tilde{M}\, \tilde{F}(M)\, J$$

$$= F(M) + \tfrac{1}{2} F(M)\, (-I - F(M)) + \tfrac{1}{2} (-I - F(M))\, J^{-1}\, \tilde{F}(M)\, J$$

$$- \tfrac{1}{4} F(M)\, (-I - F(M))\, J^{-1}\, \tilde{F}(M)\, J.$$

To calculate the relative symplecticity of M', we will need the transpose, of F(M),

$$\tilde{F}(M) = -(I + \tilde{J}M\tilde{J}\tilde{M}) = -(I + JMJ\tilde{M}), \tag{5.3}$$

so that

$$J^{-1}\,\tilde{F}(M)\,J = J(I + JMJ\tilde{M})\,J = -I - MJ\tilde{M}J = F(M) \tag{5.4}$$

where the relation $J^2 = -I$ has been used. Then

$$F(M') = F(M) - \frac{1}{2}F(M)\,(I + F(M)) - \frac{1}{2}(I + F(M))\,F(M)$$

$$+ \frac{1}{4}F(M)\,(I + F(M))\,F(M) = -\frac{3}{4}F^2(M) + \frac{1}{4}F^3(M). \tag{5.5}$$

Thus the deviation is $\|F(M')\| \approx \|F^2(M)\| = \|F(M)\|^2$. This shows that the process is quadratically convergent when iterating. If $\|F(M)\| < 1$, $\|F(M')\| \leq \|F(M)\|$, equality only if $F(M) = 0$, for some suitable matrix norm. Thus we may iterate: M' may be used to calculate a more symplectic M", and so on, until we are satisfied with the degree of symplecticity.

A third method uses the Cayley representation of symplectic matrices. If a symplectic matrix M can be written $M = \exp(JS)$ with S symmetric, we may rewrite it as

$$M = \frac{I + \tanh(JS/2)}{I - \tanh(JS/2)} = \frac{I + JW}{I - JW} \tag{5.6}$$

where $W = -J\tanh(\varepsilon JS/2)$ is symmetric if and only if M is symplectic. Now run this backwards: we start with a matrix M that is nearly symplectic. Define the matrix V by

$$V = J \frac{I - M}{I + M} \tag{5.7}$$

which will be an approximate W. We create the actual W by symmetrizing V: $W = (V+\tilde{V})/2$. We may now use the formula $\frac{I + JW}{I - JW}$ above to create a new matrix M'. We are assured that M' is near M and is exactly symplectic.

This method may be extended so that we can exponentiate and symplectify in one step. We start with the representation of the matrix we wish to calculate as $M = \exp(\varepsilon JS)$, where S is symmetric and known. Then we may calculate M by the formula (5.6), using for W the approximation of $\tanh(\varepsilon \frac{JS}{2})$ by its Taylor series truncated at some suitable point:

$$W \approx W_a = \sum_{n=0}^{m} a_n \varepsilon^n \left(\frac{JS}{2}\right)^n. \tag{5.8}$$

This truncated series is automatically symmetric, so we need not symmetrize. The first few coefficients a_n for the hyperbolic tangent are

$$a_o = 0, \quad a_1 = 1, \quad a_2 = 0, \quad a_3 = -\frac{1}{3}. \tag{5.9}$$

The validity of the approximation of the hyperbolic tangent in the Cayley method may be verified in the following manner. Let

$$W = \tanh\left(\frac{JS}{2}\right) = \sum_{n=0}^{\infty} a_n \varepsilon^n \left(\frac{JS}{2}\right)^n, \tag{5.10}$$

where only odd n enter into the summation because the hyperbolic tangent

is an odd function. Let us write the truncated series as

$$\Sigma = JW_a = J \sum_{n=0}^{m} a_n \varepsilon^n \left(\frac{JS}{2}\right)^n \tag{5.11}$$

the remainder term being dropped is

$$\Sigma_r = JW - \Sigma = J \sum_{n=m+1}^{n} a_n \varepsilon^n \left(\frac{JS}{2}\right)^n. \tag{5.12}$$

The approximated matrix M is

$$M_a = \frac{1 + JW_a}{1 - JW_a} = \frac{1 + \Sigma}{1 - \Sigma} \tag{5.13}$$

the actual matrix is

$$M = \frac{1 + JW}{1 - JW} = \frac{1 + \Sigma + \Sigma_r}{1 - \Sigma - \Sigma_r} = \frac{1 + \Sigma + \Sigma_r}{1 - \Sigma} \frac{1}{1 - \frac{\Sigma_r}{1 - \Sigma}} \tag{5.14}$$

$$= \left(\frac{1 + \Sigma}{1 - \Sigma} + \frac{\Sigma_r}{1 - \Sigma}\right) \frac{1}{1 - \frac{\Sigma_r}{1 - \Sigma}}$$

$$= \left(M_a + \frac{\Sigma_r}{1 - \Sigma}\right) \left[1 + \frac{\Sigma_r}{1 - \Sigma} - \left(\frac{\Sigma_r}{1 - \Sigma}\right)^2 + \cdots\right]$$

$$= M_a + \frac{\Sigma_r}{1 - \Sigma} + M_a \frac{\Sigma_r}{1 - \Sigma} + \left(\frac{\Sigma_r}{1 - \Sigma}\right)^2 - M_a \left(\frac{\Sigma_r}{1 - \Sigma}\right)^2 - \left(\frac{\Sigma_r}{1 - \Sigma}\right)^3 + \cdots$$

$$= M_a + O(\varepsilon^{m+2})$$

so

$$M - M_a = O(\varepsilon^{m+2}). \tag{5.15}$$

Because S contains terms in ε of order 1 through N-2, M will not be accurate in ε beyond order N-2. Therefore, we may truncate at m = N-3. For N=4, we need keep just one term.

6. Determination of the Fixed Point

Once we have computed the transformation for a section of the accelerator,

$$M = e^{:g_1:} e^{:g_2:} e^{:g_3:} e^{:g_4:}, \tag{6.1}$$

a desirable thing to know would be the fixed point(s) v: the points that satisfy

$$v = Mv. \tag{6.2}$$

If the transfer map M represents one complete turn of a circular accelerator, the fixed point will be the closed orbit. A particle starting with those phase space coordinates will return to those coordinates in each successive turn. Once the closed orbit has been found it may be corrected to zero by a variety of methods, some of which are outlined in Appendix B.

If the closed orbit remains uncorrected, we will want to extract information about the behavior around the closed orbit, such as tune, chromaticity, and so on. In terms of the Lie series, if

$$\bar{w} = e^{:g_1:} e^{:g_2:} e^{:g_3:} e^{:g_4:} \cdot|_w, \tag{6.3}$$

then the map around the fixed point is given by the polynomials f_n where

$$\bar{w} - v = e^{:f_2:} e^{:f_3:} e^{:f_4:} \cdot|_{w-v}. \tag{6.4}$$

$$f_1^{(0)},\ f_1^{(1)},\ \cdots \tag{6.8}$$

be a sequence of first-order polynomials yet to be determined. Define a sequence of maps

$$N^{(0)},\quad N^{(1)},\ \cdots \tag{6.9}$$

by the rule

$$N^{(n)} = e^{-:f_1^{(n)}:} \underset{M}{} e^{+:f_1^{(n)}:} \tag{6.10}$$

for a non-negative integer n where $f_1^{(n)}$ is defined below. By application of the concatenation scheme above, $N^{(n)}$ may be put into the standard factorization

$$N^{(n)} = e^{:d_1^{(n)}:} \underset{e}{} e^{:d_2^{(n)}:} \underset{e}{} e^{:d_3^{(n)}:} \underset{e}{} e^{:d_4^{(n)}:} \tag{6.11}$$

Now define the sequence $\{f_1^n\}_{n=0,1,\ \cdots}$ by

$$f_1^{(0)} = 0, \tag{6.12a}$$

$$f_1^{(n+1)} = f_1^{(n)} + (1 - e^{:d_2^{(n)}:})^{-1}\, d_1^{(n)}\ , \tag{6.12b}$$

$$n = 0,1,2,\cdots$$

As these maps are iterated, $d_1^{(n)}$ will approach 0, and $d_m^{(n)}$ will approach f_m, $m \geq 2$.

This procedure is, in principle, quadratically convergent. That is, $d_1^{(n+1)}$, which measures how far $f_1^{(n+1)}$ is from the actual f_1. will be order ε^2 if $d_1^{(n)}$ is of order ε. Let

$$r_1^{(n+1)} = (1 - e^{:d_2^{(n)}:})^{-1} d_1^{(n)}, \tag{6.13}$$

which will be $O(\varepsilon)$ if $d_1^{(n)}$ is. Then

$$f_1^{(n+1)} = f_1^{(n)} + r_1^{(n+1)}, \tag{6.14}$$

and

$$N^{(n+1)} = e^{-:r_1^{(n+1)}:} N^{(n)} e^{:r_1^{(n+1)}:} \tag{6.15}$$

$$= e^{-:r_1^{(n+1)}:} e^{:d_1^{(n)}:} e^{:d_2^{(n)}:} e^{:d_3^{(n)}:} e^{:d_4^{(n)}:} e^{:r_1^{(n+1)}:}$$

$$= e^{-:r_1^{(n+1)}:} e^{:d_1^{(n)}:} e^{:d_2^{(n)}:} e^{:r_1^{(n+1)}:} e^{-:r_1^{(n+1)}:} e^{:d_3^{(n)}:} e^{:d_4^{(n)}:} e^{:r_1^{(n+1)}:}$$

$$= e^{:d_1^{(n)} + (e^{:d_2^{(n)}:} - 1) r_1^{(n+1)}:} e^{:d_2^{(n)}:} e^{-:r_1^{(n+1)}:} e^{:d_3^{(n)}:} e^{:d_4^{(n)}:} e^{:r_1^{(n+1)}:}$$

$$= e^{:d_2^{(n)}:} e^{-:r_1^{(n+1)}:} e^{:d_3^{(n)}:} e^{:d_4^{(n)}:} e^{:r_1^{(n+1)}:}$$

by the definition of $r_1^{(n+1)}$.

Comparing this with (6.11) at iteration n+1, and recalling the rule for moving first-order polynomials, we see $d_1^{(n+1)}$ will be $O(\varepsilon^2)$.

In the above analysis, $1 - e^{:d_2:}$ was assumed to be invertible.

This is not always the case. Wherever a tune is 0 or 1, this quantity will not be invertible. Of course, a realistic accelerator lattice will not have such a tune in the horizontal or vertical degrees of freedom but the third degree of freedom is flight time, and in such a case the tune will be 0 or 1 if the Hamiltonian is time-independent, i.e. if the particle's energy is not changed.

If the Hamiltonian is not explicitly dependent on time, and consequently $1 - e^{:d_2:}$ is not invertible, the process described above may still be used with slight modification. Consider a particle starting with coordinates $u_\perp$ in the transverse part of phase space (X,P_X,Y,P_Y). Since the Hamiltonian is time-independent, it does not change the energy P_T. Therefore, any value is suitable for the P_T component of the fixed point. On the other hand, the flight time T will be changed by a fixed amount τ that is independent of the initial value of T, and dependent only on $u_\perp$. Consequently, if we compute the fixed point $u_\perp$ by the process above working only on the transverse part of phase space, we may say that

$$u = (u_{\perp X},u_{\perp P_X},u_{\perp Y},u_{\perp P_Y},T,0), \tag{6.16}$$

where T is arbitrary, is a fixed point of the 4-dimensional subspace (X,P_X,Y,P_Y) when the Hamiltonian is time-independent.

7. The Euclidean Group

So far, we have seen groups associated with dynamical evolution in a Hamiltonian system. We now turn our attention to a different kind of Lie group: the Euclidean group. This is the group of rigid body motions in space, which we shall need to describe the misalignment of beamline elements.

There are six degrees of freedom for rigid body motion (Goldstein [1950]); three translational, which form the subgroup T^3, and three rotational, which form the subgroup SO(3). The Euclidean group is the semi-direct product of T^3 and SO(3); if $\alpha_i \in T^3$, $R_i \in SO(3)$ for $i = 1,2$, the multiplication in the Euclidean group is

$$E(\alpha_1;R_1)\, E(\alpha_2;R_2) = E(\alpha_1 + \tau(R_1)(\alpha_2);\ R_1R_2) \qquad (7.1)$$

where τ: SO(3) $\to \mathbf{R}^{3*}$ is the representation of SO(3) as a linear transformation in $\mathbf{R}^3$, given in (7.4). The identity is the semi-direct product of the identities of the two subgroups, E(0;I), and the inverse is

$$E^{-1}(\alpha;R) = E(-\tau(R^{-1})(\alpha),\ R^{-1}) \qquad (7.2)$$

which may be checked using (7.1).

The action of an element α of T^3 on a rigid body is straightforward. If we pick a set of coordinates, and give some fiducial point of the body relative to these coordinates, α need only be the vector from the origin to the fiducial point.

The rotation group SO(3) does not have as natural a parameterization. As a start, note that every rotation leaves some axis (one dimensional subspace) in $\mathbb{R}^3$ unchanged; one way to parameterize rotations is to give the axis and the angle with direction of rotation given by the right-hand rule. More useful for our purposes, however, is the Euler angle parameterization, which is easily explicated in terms of the axis-angle parameterization.

The Euler angles ϕ, θ, ψ, specify three successive rotations about the coordinate axes (Goldstein [1950], Dragt [1986]).* In the first step, we rotate about the Z axis by an angle ϕ. In the second we rotate about the Y axis by an angle θ. Finally, in the third step we rotate about the Z axis by an angle ϕ. The only disadvantage of the Euler angles is that they are many-to-one, but we may make special allowance for this.

In order to determine the multiplication and inversion rules under the Euler angle parameterization, we shall pick a representation in which we can do the multiplication. The result will then be compared with the general form of the representation to extract the Euler angles. For this method to work, we need a faithful representation; once again we turn to the natural representation of 3 by 3 matrices. Although the parameterization in terms of Euler angles is many-to-one, it will yield a useful answer.

Writing out the three axis rotation,

*Note Goldstein uses passive rotations, and Dragt uses active rotations. I shall follow Dragt, because he must approve this dissertation.

$$R(\phi,\theta,\psi) = R_Z(\phi)\, R_Y(\theta)\, R_Z(\psi). \tag{7.3}$$

In terms of the matrix representation, the Euler angle parameterization looks like

$$M(\phi,\theta,\psi) = \tau((R\phi,\theta,\psi)) \tag{7.4}$$

$$= \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\phi\cos\theta\cos\psi - \sin\phi\sin\psi & -\cos\phi\cos\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\sin\theta \\ \sin\phi\cos\theta\cos\psi + \cos\phi\sin\psi & -\sin\phi\cos\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\sin\theta \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta \end{bmatrix}$$

We may multiply two of these matrices to get the product

$$M(\bar{\phi},\bar{\theta},\bar{\psi}) = M(\phi_2,\theta_2,\psi_2)\, M(\phi_1,\theta_1,\psi_1), \tag{7.5}$$

and now solve for $\bar{\phi}$, $\bar{\theta}$, $\bar{\psi}$ by looking at the form of the matrix above and extracting the quantities. That is, we know that

$$M_{33}(\bar{\phi},\bar{\theta},\bar{\psi}) = \cos\bar{\theta}\ , \tag{7.6}$$

so we find

i)
$$\bar{\theta} = \arccos\,(M_{33}(\bar{\phi},\bar{\theta},\bar{\psi})). \tag{7.7}$$

Also, from elements 1,3 and 2,3

ii) $$\bar{\phi} = \arctan\left(\frac{M_{23}(\bar{\phi},\bar{\theta},\bar{\psi})}{M_{13}(\bar{\phi},\bar{\theta},\bar{\psi})}\right), \tag{7.8}$$

and from elements 3,1 and 3,2

iii) $$\bar{\psi} = \arctan\left(-\frac{M_{31}(\bar{\phi},\bar{\theta},\bar{\psi})}{M_{32}(\bar{\phi},\bar{\theta},\bar{\psi})}\right). \tag{7.9}$$

Formulas ii) and iii) may give the wrong answer or no answer under some circumstances; we need to be careful in two respects. First, the function used should be a two-argument function, numerator and denominator, that returns values on the full circle $[-\pi,\pi]$ rather than on the right half $[-\frac{\pi}{2},\frac{\pi}{2}]$. Such a function is normally called atan2 (e.g., in Fortran). Second, if $\sin\bar{\theta} = 0$, both numerator and denominator will be 0, and $M(\bar{\phi},\bar{\theta},\bar{\psi})$ will represent a simple rotation around the z axis

$$\begin{bmatrix} \pm\cos(\bar{\psi} \pm \bar{\phi}) & \mp\sin(\bar{\psi} \pm \bar{\phi}) & 0 \\ \sin(\bar{\psi} \pm \bar{\phi}) & \cos(\bar{\psi} \pm \bar{\phi}) & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \tag{7.10}$$

We may choose $\bar{\psi} = 0$, so that the matrix is

$$\begin{bmatrix} \pm\cos\bar{\psi} & -\sin\bar{\phi} & 0 \\ \pm\cos\bar{\phi} & \cos\bar{\phi} & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \tag{7.11}$$

In this case

$$\bar{\psi} = \arccos\left(\pm M_{11}(\bar{\phi},\bar{\theta},\bar{\psi})\right). \tag{7.12}$$

The answers are

i) $$\bar{\theta} = \arccos[\cos\theta_1 \cos\theta_2 - \cos(\phi_1 + \psi_2) \sin\theta_1 \sin\theta_2]. \tag{7.13}$$

ii) If $|\cos \bar{\theta}| \neq 1$, $\bar{\phi} = \arctan(a/b)$ where

$$\begin{aligned} a = {} & \sin\phi_2 \sin\theta_2 \cos\theta_1 + \cos\phi_2 \sin(\phi_1+\psi_2) \sin\theta_1 \\ & + \sin\phi_2 \cos(\phi_1+\psi_2) \cos\theta_2 \sin\theta_1 \end{aligned} \tag{7.14a}$$

$$\begin{aligned} b = {} & \cos\phi_2 \sin\theta_2 \cos\theta_1 - \sin\phi_2 \sin(\phi_1+\psi_2) \sin\theta_1 \\ & + \cos\phi_2 \cos(\phi_1+\psi_2) \cos\theta_2 \sin\theta_1. \end{aligned} \tag{7.14b}$$

If $\cos \bar{\theta} = 1$,

$$\begin{aligned} \bar{\phi} = \arccos[& -\sin\phi_2 \sin\psi_1 \cos(\phi_1 + \psi_2) - \cos\phi_2 \sin\theta_1 \sin\theta_2 \cos\psi_1 \\ & - \cos\phi_2 \cos\theta_2 \sin\psi_1 \sin(\phi_1 + \psi_2) \\ & - \sin\phi_2 \cos\theta_1 \cos\psi_1 \sin(\phi_1 + \psi_2) \\ & + \cos\phi_2 \cos\theta_2 \cos\theta_1 \cos\psi_1 \cos(\phi_1 + \psi_2)]. \end{aligned} \tag{7.14c}$$

iii) If $|\cos \bar{\theta}| \neq 1$, $\bar{\psi} = \arctan(a/b)$ where

$$\begin{aligned} a = {} & \sin\theta_1 \cos\theta_2 \sin\psi_1 + \sin(\phi_1+\psi_2) \sin\theta_2 \cos\psi_1 \\ & + \cos(\phi_1+\psi_2) \cos\theta_1 \sin\theta_2 \sin\psi_1, \end{aligned} \tag{7.15a}$$

$$\begin{aligned} b = {} & \sin\theta_1 \cos\theta_2 \cos\psi_2 - \sin(\phi_1+\psi_2) \sin\theta_2 \sin\psi_2 \\ & + \cos(\phi_1+\psi_2) \cos\theta_1 \sin\theta_2 \cos\psi_1. \end{aligned} \tag{7.15b}$$

If $|\cos \bar{\theta}| = 1$, $\bar{\psi} = 0$.

Inverses are easily obtained by noting that SO(3) parameterized by Euler angles is specified as a product of three easily-inverted rotations

$$R(\phi,\theta,\psi) = R_Z(\phi)\, R_Y(\theta)\, R_Z(\psi). \tag{7.16}$$

The inverses are

$$R_i^{-1}(\beta) = R_i(-\beta) \quad , \quad i = X,Y,Z, \tag{7.17}$$

so

$$R^{-1}(\phi,\theta,\psi) = R_Z(-\psi)\, R_Y(-\theta)\, R_Z(-\phi) \tag{7.18}$$

$$= R(-\psi,-\theta,-\phi).$$

PART II: Computation of Symplectic Maps

We now have the mathematical tools in hand to concatenate Lie transformations, track particles through them, and determine the fixed point, for maps that include first-order terms in the factorization. What has not been covered yet is the source of these maps; given an actual accelerator that we wish to model, how do we obtain the matrix and polynomials of the transfer map of each element?

This task is divided into four chapters. Chapter 8 shows a method for computing the factored Lie transformations from a Hamiltonian or from the Taylor series for the transfer map. It is based on the work of Dragt and Forest [1983]. Chapters 9 and 10 contain computations of the Lie transformations for the steering dipole and mispowered dipoles. The former is useful for error correction, which will be treated later, and the latter allows us to treat a particular kind of error that occurs in accelerators. Chapters 11 and 12 treat the problem of beamline element misalignment. Chapter 11 shows how to convert a misalignment at the fiducial point of an element into coordinate transformations (matching maps) for the entry and exit faces. These transformations depend only on the misalignment and the general geometry (straight or curved) of the element. Finally, Chapter 12 shows how to compute the realization of the Euclidean group, which we now have as coordinate transformations at the entry and exit faces, in terms of symplectic maps. When concatenated with the map for a perfect element, which is independent of the misalignment, the result is the map for the misaligned element. As a byproduct of these computations, some general comments about rotations are made.

The thrust of this part is computation of factored maps with first-order terms. This is not exactly the same thing as maps of elements with errors. For example, a mispowered quadrupole map will have no first-order term, because the quadrupole still sends a design particle out on the design trajectory. In situations of this sort, the factored maps are readily computed; the techniques of Chapter 8 will suffice for the case $H_1 = 0$. More likely, however, the ideal map has already been calculated and just needs to be computed with the actual parameters. In the quadrupole case, for example, Douglas [1982] has already computed the map, and we need only supply the actual powering to find the matrix and polynomials.

8. Computation of Factored Maps from a Hamiltonian

So far, we have seen that a symplectic map may be written in the factored form,

$$e^{:f_1:}\, e^{:f_2^{(c)}:}\, e^{:f_2^{(a)}:}\, e^{:f_3:}\, e^{:f_4:} \cdots, \tag{8.1}$$

and we have seen how to manipulate and use this factorization. In the course of showing that a symplectic map can be represented by factored Lie transformations, we have seen how to find the polynomials given the coefficients of the Taylor expansion. We do not yet, however, know how to get these polynomials directly from the Hamiltonian. This will be useful for some of the computations in succeeding chapters; we shall therefore treat it here.

We divide the task into three cases. First, break up the Hamiltonian order-by-order in the phase space variables,

$$H = H_1 + H_2 + H_3 + \cdots, \tag{8.2}$$

H_n = polynomial homogeneous of order n in the phase space variables. The first case is $H_1 = 0$, or that for which there are no constant terms in the transfer map. The method that applies in this case is described by Dragt and Forest [1983]. The remaining cases apply when $H_1 \neq 0$, for which there are constant terms in the transfer map. When H_1 is infinitesimal (i.e., we desire computation only to the required total order, in the sense of Chapter 3), the method of Dragt and Forest may be extended. For H_1 arbitrary, there are two alternatives. One is to

split the Hamiltonian into the product of two transformations, one first-order, the other second and higher order, and apply the Dragt and Forest techniques to the result. This does not always work. The second alternative is to compute the transfer map either from geometric considerations or by application of the Hamiltonian map to phase space, and then to integrate the result as described in Chapter 1. Of course geometric considerations may always be used. The overall procedure is summarized as a flow chart, Figure 8.1.

Let us begin by assuming that we are studying the transformation from a set of phase space variables ζ_o at time t_o to the set $\zeta(t)$ parameterized by the time t. The independent variable will be t, although it need not actually be time; it may be longitudinal position, for instance. Since the transformation is canonical, there is a Hamiltonian $H(\zeta,t)$ that gives the dynamics according to Hamilton's equations. Assuming that H is known, our goal is to find the functions f_1, f_2, f_3, etc., such that

$$\zeta(t) = e^{:f_1(\zeta,t):}\, e^{:f_2^c(\zeta,t):}\, e^{:f_2^a(\zeta,t):}\, e^{:f_3(\zeta,t):} \cdots \cdot|_{\zeta_o}. \qquad (8.3a)$$

or for short

$$\zeta(t) = M(t)\zeta_o. \qquad (8.3b)$$

It will be assumed that the phase space variables are small in the sense described in Chapter 3.

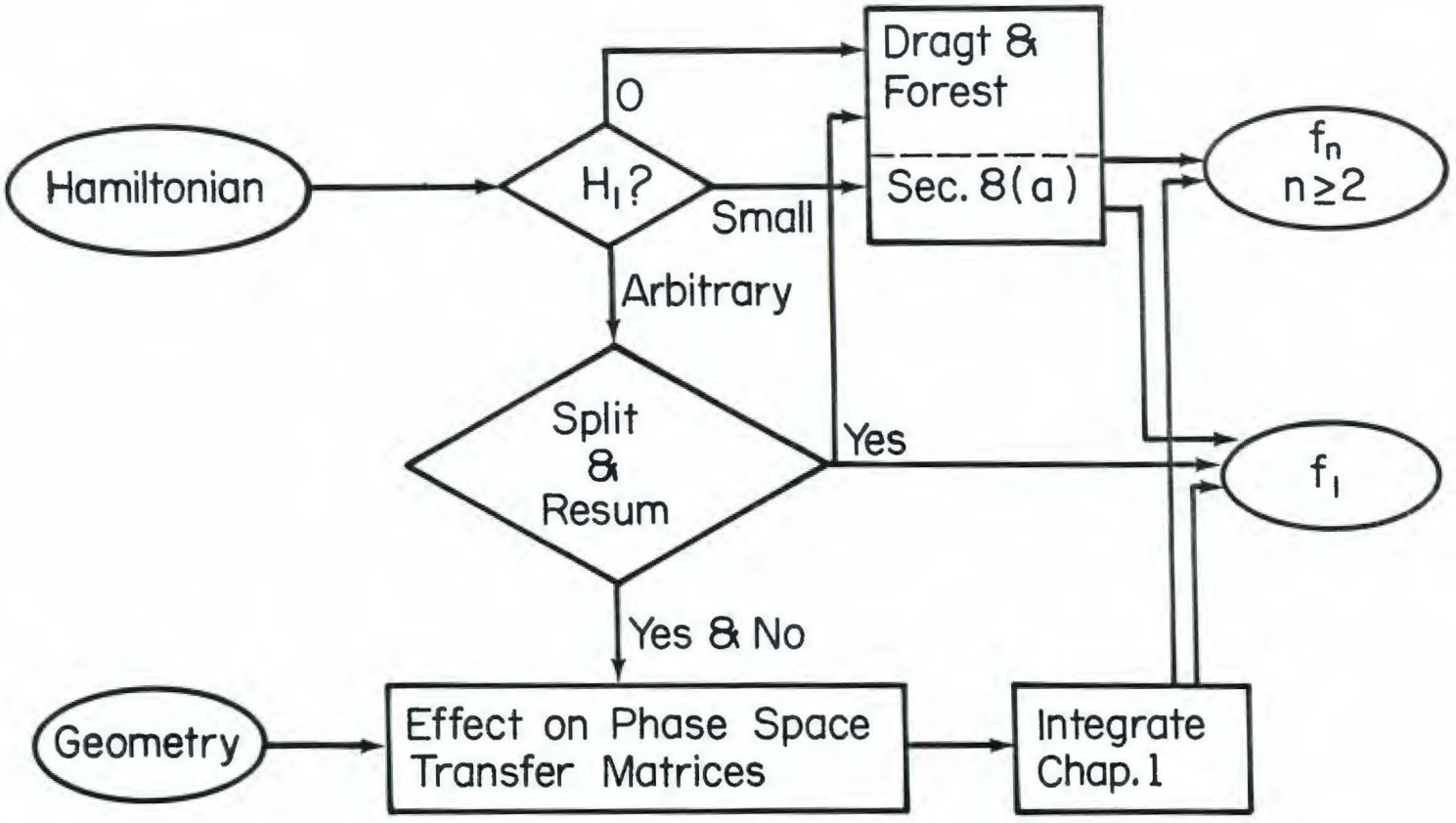


Figure 8.1 Flow Chart for Computation of a Factored Map

a. H_1 Small or Zero*

Let us rewrite the Hamiltonian factored by δ order,

$$H = H_1 + H_2 + H_3 + \ldots \tag{8.4}$$

The first-order part is small; H_1 contains a factor of ε, or the integration over the independent variable is a range of length ε. We may calculate the factored product expansion through the appropriate order in ε by a method similar to that of Dragt and Forest [1983] and Forest [1984]. In the case $\varepsilon = 0$, it reduces to those computations.

To begin, let M be the transfer map

$$\zeta(t) = M\zeta_0. \tag{8.5}$$

If g is any function on phase space we may say

$$g(\zeta) = g(M\zeta_0) = Mg(\zeta_0), \tag{8.6}$$

so

$$\dot{g}(\zeta) = \dot{M}\, g(\zeta_0). \tag{8.7}$$

From the equations of motion we note,

*I am grateful to Etienne Forest for suggesting this idea.

$$\dot{g}(\zeta) = [g(\zeta),H(\zeta,t)] \tag{8.8}$$

$$= [g(M\zeta_o),H(M\zeta_o,t)]$$

$$= [\,Mg(\zeta_o),\,MH(\zeta_o,t)] = M[g(\zeta_o),H(\zeta_o,t)]$$

$$= :-H(\zeta_o,t):\, g(\zeta_o)$$

Comparing this with (8.7) above, and noting g is arbitrary, we find obeys the equation of motion

$$\dot{M} = M:-H(\zeta_o,t): \tag{8.9}$$

The process of solving for M is iterative in powers of ε; for $\varepsilon = 0$, it has one step. Each step in the iteration produces a set of terms

$$N^{(i)} = e^{:g_2^{(i)}:}\, e^{:g_3^{(i)}:}\, e^{:g_4^{(i)}:} \cdots \tag{8.10}$$

on the right; when we are done we shall have a series of second-order-and-higher sets:

$$M = e^{:g_1:}\, N^{(n)} \cdots N^{(1)} \tag{8.11}$$

$$= e^{:g_1:}\, e^{:g_2^{(n)}:}\, e^{:g_3^{(n)}:}\, e^{:g_4^{(n)}:} \cdots e^{:g_2^{(1)}:}\, e^{:g_3^{(1)}:}\, e^{:g_4^{(1)}:} .$$

We may then concatenate the $N^{(i)}$ to form a single set

$$M = e^{:f_1:} e^{:f_2:} e^{:f_3:} e^{:f_4:} \tag{8.12}$$

with $f_1 = g_1$, using the techniques of Chapter 4.

The dynamical equation in M,

$$\dot{M}^{(i)} = M^{(i)} \; :-H^{(i)}:, \tag{8.13}$$

will be solved iteratively, starting with $H^{(1)} = H$. Each step will produce a partial answer for $M^{(i)}$

$$M^{(i)} = M^{(i+1)} \; N^{(i)}, \tag{8.14a}$$

with $N^{(i)}$ determined and $M^{(i+1)}$ undetermined explicitly but governed by the equation (8.13) with $H^{(i+1)}$ determined. We shall now see how to obtain $N^{(i)}$ and $H^{(i+1)}$. At the final step of the iteration, the solution obtained for M will be

$$M^{(n)} = N^{(n)} \; . \tag{8.14b}$$

Let $H_R^{(i)} = H_3^{(i)} + H_4^{(i)} + \ldots$ so that $H^{(i)} = H_1^{(i)} + H_2^{(i)} + H_R^{(i)}$. Then we may write (8.13) as

$$\dot{M}^{(i)} = M^{(i)} \; :-H_1^{(i)} - H_2^{(i)} - H_R^{(i)}: \tag{8.15}$$

Now, write $M^{(i)}$ in terms of the product

$$M^{(i)} = M^{(i+1)} \; N_R^{(i)'} \; N_2^{(i)} \tag{8.16}$$

where each term will be determined. Further, write N_R and N_2 in terms of Lie transformations,

$$N_R^{(i)'} = e^{:g_3':} e^{:g_4':} \cdots \tag{8.17}$$

and

$$N_2^{(i)} = e^{:g_2:}. \tag{8.18}$$

We shall first find dynamical expressions for $N_R^{(i)'}$ and $N_2^{(i)}$, then determine $g_n^{(i)'}$ and $g_2^{(i)}$ from them, and finally apply the concatenation rules of Chapter 4 to obtain the form (8.10) for $N^{(i)}$.

Substituting (8.16) in (8.15),

$$\dot{M}^{(i+1)} N_R^{(i)'} N_2^{(i)} + M^{(i+1)} \dot{N}_R^{(i)'} N_2^{(i)} + M^{(i+1)} N_R^{(i)'} \dot{N}_2^{(i)} \tag{8.19}$$

$$= M^{(i+1)} N_R^{(i)'} N_2^{(i)} :-H_1^{(i)}-H_2^{(i)}-H_R^{(i)}: .$$

Let N_2 satisfy the dynamical equation

$$\dot{N}_2^{(i)} = N_2^{(i)} :-H_2^{(i)}:. \tag{8.20}$$

If $H_2^{(i)}$ commutes with itself at different times, (8.20), is solved by the exponential of the integral of $H_2^{(i)}$,

$$N_2^{(i)}(t) = e^{:g_2^{(i)}(t):} = e^{-\int_{t_o}^{t} :H_2^{(i)}(\zeta_o,t'):dt'} . \tag{8.21}$$

Alternatively, we may write $H_2^{(i)}$ in terms of the symmetric matrix S,

$$H_2^{(i)}(\zeta,t) = \frac{1}{2} \sum_{ab} S_{ab}^{(i)} \zeta_a \zeta_b \tag{8.22}$$

We may then write for (8.20) a matrix differential equation

$$\dot{M} = JSM. \tag{8.23}$$

This matrix M will give the linear mapping of phase space. Section b has a more detailed discussion of the solution of (8.20).

With the H_2 term out of the way, the dynamical equation (8.19) reduces to the two higher-order terms

$$\dot{M}^{(i+1)} N_R^{(i)\prime} N_2^{(i)} + M^{(i+1)} \dot{N}_R^{(i)\prime} N_2^{(i)} = M^{(i+1)} N_2^{(i)\prime} N_2^{(i)} :-H_1^{(i)} - H_R^{(i)}:. \tag{8.24}$$

Next we ask that $N_R^{(i)\prime}$ satisfy

$$\dot{N}_R^{(i)\prime} = M_R^{(i)\prime} N_2^{(i)} :-H_R^{(i)}: N_2^{(i)-1} . \tag{8.25}$$

Let

$$H_R^{(i)\,\mathrm{int}}(\zeta_o,t) = N_2^{(i)}(t) : H_R^{(i)}(\zeta,t) \,|_{\zeta_o} : N_2^{(i)-1}(t) \tag{8.26}$$

$$= e^{:g_2^{(i)}(t):} :H_R^{(i)}(t): e^{-:g_2^{(i)}(t):} = H_R^{(i)}(e^{:g_2^{(i)}(t):} \cdot |_{\zeta_o}, t),$$

so (8.23) may be written as

$$\dot{N}_R^{(i)'} = N_R^{(i)'} :-H_R^{(i)^{int}}: \qquad (8.27)$$

We may solve this by noting

$$N_R^{(i)'}(t) = I + \int_{t_o}^{t} N_R^{(i)'}(t') :-H_R^{(i)^{int}}(t'):dt':, \qquad (8.28)$$

where I, the identity map, is the value of $N_R^{(i)'}$ at t=0. We may make a Born (or Neumann) expansion on this integral equation by repeatedly substituting the left-hand side into the right; thus

$$N_R^{(i)'}(t) = I + \int_{t_o}^{t} :-H_R^{(i)^{int}}(t'):dt' \qquad (8.29)$$

$$+ \int_{t_o}^{t}\int_{t_o}^{t'} :-H_R^{(i)^{int}}(t'')::- H_R^{(i)^{int}}(t'):dt'dt'' + \ldots$$

If we expand $H_R^{(i)^{int}}$,

$$H_R^{(i)^{int}} = H_3^{(i)^{int}} + H_4^{(i)^{int}} + \ldots , \qquad (8.30)$$

where $H_n^{(i)^{int}}$ is the homogeneous n^{th} order part of $H_R^{(i)^{int}}$, and separate the terms by the order they change a homogeneous polynomial, (8.29) becomes

$$N_R^{(i)'}(t) = I + \int_{t_o}^{t} :-H_3^{(i)^{int}}(t'):dt' \qquad (8.31)$$

$$+\left(\int_{t_o}^{t} :-H_4^{(i)^{int}}(t'):dt' + \int_{t_o}^{t}\int_{t_o}^{t'} :-H_3^{(i)^{int}}(t'')::-H_3^{(i)^{int}}(t'):dt''dt'\right)$$

$$+ \ldots$$

First, we write $N_2^{(i)'}$ in factored form, and then expand,

$$N_R^{(i)'}(t_f) = e^{:g_3^{(i)'}:} e^{:g_4^{(i)'}:} \cdots \tag{8.32}$$

$$= (1 + :g_3^{(i)'}: + \tfrac{1}{2} :g_3^{(i)'}:^2 + \ldots)(1 + :g_4^{(i)'}: + \ldots)$$

$$= 1 + :g_3^{(i)'}: + \tfrac{1}{2} :g_3^{(i)'}:^2 + :g_4^{(i)'}: + \ldots .$$

Comparing the order they change a homogeneous polynomial, we find that $:g_3^{(i)}:$ is

$$g_3^{(i)'} = \int_{t_o}^{t_f} - H_3^{(i)^{int}}(t')\, dt'. \tag{8.33}$$

The next order change is more complicated. Note that (8.33) implies that $:g_3^{(i)'}:^2$ is

$$\tfrac{1}{2} :g_3^{(i)'}:^2 = \tfrac{1}{2} \int_{t_o}^{t_f}\int_{t_o}^{t_f} :-H_3^{(i)^{int}}(t'')::-H_3^{(i)^{int}}(t'): dt''dt'. \tag{8.34}$$

Splitting up the second integral yields

$$\tfrac{1}{2} :g_3^{(i)'}:^2 = \tfrac{1}{2} \int_{t_o}^{t_f}\int_{t_o}^{t'} (:-H_3^{(i)^{int}}(t'')::-H_3^{(i)^{int}}(t'): \tag{8.35}$$

$$+ :-H_3^{(i)^{int}}(t')::-H_3^{(i)^{int}}(t''):)dt''dt',$$

so the next order is

$$\tfrac{1}{2} :g_3^{(i)'}:^2 + :g_4^{(i)'}: = \int_{t_o}^{t_f} :-H_4^{(i)^{int}}(t'): dt' \tag{8.36}$$

$$+ \int_{t_o}^{t_f}\int_{t_o}^{t'} :-H_3^{(i)^{int}}(t'')::-H_3^{(i)^{int}}(t'): dt''dt',$$

which gives : $g_4^{(i)'}$:,

$$:g_4^{(i)'}: = \int_{t_o}^{t_f} :-H_4^{(i)^{int}}(t'): dt' \tag{8.37}$$

$$+ \frac{1}{2}\int_{t_o}^{t_f}\int_{t_o}^{t'} (:-H_3^{(i)^{int}}(t'')::-H_3^{(i)^{int}}(t'):$$

$$- :-H_3^{(i)^{int}}(t')::-H_3^{(i)^{int}}(t''):) dt''dt'$$

$$= \int_{t_o}^{t_f} :-H_4^{(i)^{int}}(t'): dt'$$

$$+ \frac{1}{2}\int_{t_o}^{t_f}\int_{t_o}^{t'} [:-H_3^{(i)^{int}}(t'')::-H_3^{(i)^{int}}(t'):] dt''dt'.$$

Using the homomorphism between the Poisson bracket Lie algebra and its adjoint algebra, we have an expression for $g_4^{(i)'}$,

$$g_4^{(i)'} = -\int_{t_o}^{t_f} H_4^{(i)^{int}}(t') dt' \tag{8.38}$$

$$+ \frac{1}{2}\int_{t_o}^{t_f}\int_{t_o}^{t'} [-H_3^{(i)^{int}}(t''),-H_3^{(i)^{int}}(t')] dt''dt'.$$

To obtain $N^{(i)}$ in the standard factorization, we may put $N_2^{(i)}$ on the left by using the transformation rule:

$$N_2^{(i)}(t_f)\, N_R^{(i)}(t) = N_R^{(i)'}(t)\, N_2^{(i)}(t_f) \tag{8.39}$$

$$N_R^{(i)}(t) = N_2^{(i)^{-1}}(t_f)\, N_R^{(i)'}(t)\, N_2^{(i)}(t_f)$$

$$= e^{-:g_2^{(i)}(t):}\, e^{:g_3^{(i)\prime}:}\, e^{:g_4^{(i)\prime}:} \cdots e^{:g_2^{(i)}(t):}$$

$$= e^{:g_3^{(i)}:}\, e^{:g_4^{(i)}:} \cdots$$

where, for $n \geq 3$,

$$g_n^{(i)}(\zeta,t) = e^{-:g_2^{(i)}(t_f):} g_n^{(i)\prime}(\zeta,t) \tag{8.40}$$

$$= g_n^{(i)\prime}(e^{-:g_2^{(i)}(t_f):}\zeta,t).$$

To calculate these explicitly, note that the $g_n^{(i)}$ involve integrals of $H_n^{(i)^{int}}$. The $H_n^{(i)^{int}}$, in turn, are just $H_n^{(i)}$ with a linear transformation of the arguments:

$$H_n^{(i)^{tr}}(\zeta,t) \equiv H_n^{(i)^{int}}(e^{-:g_2^{(i)}(t_f):}\zeta,t) \tag{8.41}$$

$$= H_n^{(i)}(e^{-:g_2^{(i)}(t_f):}\, e^{:g_2^{(i)}(t):}\zeta,t).$$

If H_2 is independent of time, the linear transformations may be combined

$$H_n^{(i)^{tr}} = H_n^{(i)}(e^{:g_2^{(i)}(t - t_f):}\zeta,t). \tag{8.42}$$

Specifically, therefore, $g_3^{(i)}$ and $g_4^{(i)}$ are given by integrals over $H_3^{(i)^{tr}}$ and $H_4^{(i)^{tr}}$,

$$g_3^{(i)} = \int_{t_o}^{t_f} -H_3^{(i)^{tr}}(t')\, dt' \tag{8.43}$$

$$g_4^{(i)} = \int_{t_o}^{t_f} -H_4^{(i)^{tr}}(t')\,dt' + \frac{1}{2}\int_{t_o}^{t_f}\int_{t_o}^{t'} [-H_3^{(i)^{tr}}(t''),-H_3^{(i)^{tr}}(t')]\,dt''dt'. \tag{8.44}$$

The formulae for $g_5^{(i)}$ and higher may be computed by referring to Dragt and Forest [1983].

Returning at long last to (8.22), we may use (8.23) to eliminate the second term so that there is only one term left,

$$\dot{M}^{(i+1)}\, N_R^{(i)'}\, N_2^{(i)} = M^{(i+1)}\, N_R^{(i)'}\, N_2^{(i)}\, :-H_1^{(i)}:, \tag{8.45}$$

or

$$\dot{M}^{(i+1)}\, N_2^{(i)}\, N_R^{(i)} = M^{(i+1)}\, N_2^{(i)}\, N_R^{(i)}\, :-H_1^{(i)}:. \tag{8.46}$$

We may reduce this by moving the N maps to the right side,

$$\dot{M}^{(i+1)} = M^{(i+1)}\, N_2^{(i)}\, N_R^{(i)}\, :-H_1^{(i)}:\, N_R^{(i)^{-1}}\, N_2^{(i)^{-1}} \tag{8.47}$$

$$= M^{(i+1)}\; N^{(i)}\; :-H_1^{(i)}:\; N^{(i)^{-1}}$$

$$= M^{(i+1)}\, e^{:g_2^{(i)}:}\, e^{:g_3^{(i)}:}\, e^{:g_4^{(i)}:} \cdots :-H^{(i)}: \cdots e^{-:g_4^{(i)}:}\, e^{-:g_3^{(i)}:}\, e^{-:g_2^{(i)}:}$$

$$= M^{(i+1)}\, :-H^{(i)^T}:$$

where

$$H^{(i)^T}(\zeta,t) \equiv H_1^{(i)}(e^{:g_2^{(i)}(t):}\, e^{:g_3^{(i)}(t):}\, e^{:g_4^{(i)}(t):} \cdots \zeta,t). \tag{8.48}$$

If we let

$$H^{(i+1)} = H^{(i)^T}, \tag{8.49}$$

we will be done with this step of the iteration and ready to start again at (8.15) with the step i+1; this is

$$\dot{M}^{(i+1)} = M^{(i+1)} :-H^{(i+1)}:. \tag{8.50}$$

This iteration process must not last forever, of course, if we are ever to get an answer. Thus we need a termination criterion and solution. We achieve this by using the total order criterion of Chapter 3. Assume that the integral of H_1 over the specified range of the independent variable has a small factor ε multiplying it; that is, either H_1 involves ε or $t_f - t_0 = \varepsilon$.

Each iteration produces an addition factor of ε on all but the first-order term of $H_1^{(n)}$. Eventually, the higher order terms may be dropped. Assume that H_n $(n > 2)$ and $t_f - t_0$ is independent of ε. We may define G_1 so that $H_1 = \varepsilon G_1$. Then

$$H_1^{(1)} = \varepsilon G_1 \tag{8.51}$$

and $g_n^{(1)}$ will be independent of ε. Then

$$H_1^{(2)} = \varepsilon G_1^{(2)} + O(\varepsilon) \tag{8.52}$$

and $g_n^{(2)} \propto \varepsilon$. Continuing, we see that each iteration leaves a residual term that has one higher power of ε,

$$H_1^{(3)} = \varepsilon G_1^{(3)} + O(\varepsilon^2) \quad , \quad g_n^{(3)} \propto \varepsilon^2, \tag{8.53a}$$

$$H_1^{(4)} = \varepsilon G_1^{(4)} + O(\varepsilon^3) \quad , \quad g_n^{(4)} \propto \varepsilon^3, \tag{8.53b}$$

etc.

Eventually, our total order criterion will tell us to stop, and conclude

$$H_1^{(n+1)} = \varepsilon G_1^{(n+1)} \tag{8.54}$$

Consequently, the equation

$$\dot{M}^{(n+1)} = M^{(n+1)} :-H_1^{(n+1)}: \tag{8.55}$$

is easily solved; the solution is

$$N^{(n+1)} = e^{:g_1:} \tag{8.56}$$

where

$$g_1 = \int_{t_o}^{t} :- H_1^{(n+1)}(t'): dt'. \tag{8.57}$$

If the integral of H_n ($n \geq 2$) has dependencies on ε or its powers, this only hastens the iterative process.

Putting the results from each iteration together, we get (8.11)

$$M = e^{:g_1:} e^{:g_2^{(n)}:} e^{:g_3^{(n)}:} e^{:g_4^{(n)}:} \cdots e^{:g_2^{(1)}:} e^{:g_3^{(1)}:} e^{:g_4^{(1)}:} \cdots \quad (8.11)$$

and we may use the process described in Chapter 4 to write this as

$$M = e^{:f_1:} e^{:g_2^{(1)}:} \cdots e^{:g_2^{(n)}:} e^{:f_2:} e^{:f_n:} \cdots \quad (8.1)$$

We only have to use the transformation rule to move the $e^{:g_2^{(i)}:}$ to the left, then use the concatenation rules for higher-order terms to combine them into a single set of exponents. The first-order term need not be moved, because it is already on the left.

Subsequent chapters have examples of this computation; in particular, Chapters 9 and 10 have computations with a small factor in H_1, and Chapter 12 has a computation with a small integration region. Chapter 12's computation is also done a different way (see Section c of this chapter) so that all orders of the small quantity are computed; the results of the two methods are compared.

b. Computation of the Linear Part N_2

This section is devoted to addressing the question of solution of the linear part of the map N_2 in more detail. The dynamical differential equation for this map is (8.20)

$$\dot{N}_2 = N_2 :-H_2:, \tag{8.20}$$

where the iteration-count superscript have been dropped.

If $:H_2:$ commutes with itself at different times, then

$$N_2 = e^{:g_2:} \tag{8.55}$$

where

$$g_2 = -\int_{t_o}^{t} H_2(\zeta_o,t')\, dt'. \tag{8.56}$$

If it does not commute with itself, we must resort to other methods to solve this equation. Since we are dealing with a strictly linear map on phase space, we think in terms of matrices.

Instead of working with H_2 directly, we may write

$$H_2(\zeta,t) = \frac{1}{2} \sum_{a,b} S_{ab}\, \zeta_a \zeta_b \tag{8.57}$$

and work with the symmetric matrix S, as was noted in Section a. Equivalently, JS is obtained by applying the matrix correspondence rule (1.99) to $-H_2$. If we consider N_2 to be a matrix M, the dynamical equation (8.20) becomes

$$\dot{M} = JSM, \tag{8.58}$$

as is shown in Dragt and Forest [1983].

The most general solution to (8.58) is the Born integral series similar to (8.29),

$$M = I + \int_{t_o}^{t} JS(t')\,dt' + \int_{t_o}^{t}\int_{t_o}^{t'} JS(t')\,JS(t'')\,dt''dt' + \ldots \tag{8.59a}$$

or as a solution to (8.20),

$$N_2 = I + \int_{t_o}^{t} :-H_2:(t'):\,dt' + \int_{t_o}^{t}\int_{t_o}^{t'} :-H_2(t'')::-H_2(t'):dt''dt' + \ldots \tag{8.59b}$$

These may be seen to solve (8.58) or (8.20) simply by differentiating,

$$\dot{M} = JS(t) + JS(t)\int_{t_o}^{t} JS(t')\,dt' + \ldots = JSM. \tag{8.60}$$

If $:H_2:$ commutes with itself at different times, then the matrices JS(t) may be multiplied in any order for different times. The solution for M is in this case

$$M = e^{\int_{t_o}^{t} JS(t')\,dt'}, \tag{8.61}$$

the equivalent of (8.56).

In the general case, the series (8.59) would not be solvable in a simple fashion and one might resort to integrating (8.58) numerically. Even if $:H_2:$ (or JS) does not commute with itself, it may be feasible to use (8.59) to solve for M, based on considerations of ε-order.

Suppose that H_2, and thus S, has a small factor ε, which could

arise on a later iteration in solving a Hamiltonian with a small H_1. Then each term in the series (8.59) increases by one in ε-order. Eventually, the total order criterion (Chapter 3) will allow us to truncate the series. For example, if we keep through total order 4, we would stop after three terms, becasue the double integral would have ε order 2, and on a linear transformation, that makes total order 4. This is in fact the tactic used on the mispowered normal entry bending magnet (Chapter 9).

It is interesting to consider whether the whole effect of the linear transformation, N_2 or M can be written as a single second-order Lie transformation. That is, is there a matrix R such that

$$N_2 = e^{JR?} \tag{8.62a}$$

or equivalently, is there a second-order polynomial g_2 side that

$$N_2 = e^{:g_2:?} \tag{8.62b}$$

For a self-commuting Hamiltonian, of course, the answer is yes, with g_2 and R exhibited above. In the general case, the answer is no, one must allow at least two Lie transformations, (Equation (1.107) or Dragt [1982])

$$N_2 = e^{JS^a}\, e^{JS^c} \tag{8.63a}$$

or

$$N_2 = e^{:f_2^c:}\, e^{:f_2^a:} \,. \tag{8.63b}$$

Between the commuting and the general case, we may consider what happens in the ε-order regulated case. As pointed out above, if H_2 or JS has a factor of ε, each term in (8.59) will have an additional order of ε, and the series may eventually be truncated. In order to write as a single exponent we would say

$$e^{:g_2:} = I + \int_{t_o}^{t} :-H_2(t'): dt' + \int_{t_o}^{t}\int_{t_o}^{t'} :-H_2(t'')::-H_2(t'): dt''dt' \ \ldots \tag{8.64a}$$

or

$$e^{JR} = I + \int_{t_o}^{t} JS(t')\, dt' + \int_{t_o}^{t}\int_{t_o}^{t'} JS(t')\, JS(t'')\, dt''dt' \ \ldots \tag{8.64b}$$

where the series stops eventually. Let us concentate on the form (8.64a), knowing our result will carry over to the matrix form. Write g_2 as a power series in ε,

$$g_2 = \varepsilon g_{2,1} + \varepsilon^2 g_{2,2} + \ldots \tag{8.65}$$

Now expand the exponential

$$e^{:g_2:} = e^{:\varepsilon g_{2,1} + \varepsilon g_{2,2} + \ldots:} \tag{8.66}$$

$$= I + \varepsilon :g_{2,1}: + \varepsilon^2 :g_{2,2}: + \frac{1}{2}\varepsilon^2 :g_{2,1}:^2 + \ldots$$

$$= I + \varepsilon :g_{2,1}: + \varepsilon^2 \left(\tfrac{1}{2} :g_{2,1}:^2 + :g_{2,2}:\right) + \ldots$$

Since $e^{:g_2:}$ is now factored by powers of ε, we may match it term by term with (8.64), solving successively for g_2,

$$\varepsilon g_{2,1} = \int -H_2(t')\, dt' \tag{8.67}$$

and for $g_{2,2}$

$$\varepsilon^2 :g_{2,2}: = \int_{t_o}^{t} \int_{t_o}^{t'} :-H_2(t''):\!:-H_2(t')\, dt''dt' - \tfrac{1}{2} \varepsilon^2 :g_{2,1}:^2 \tag{8.68}$$

$$= \int_{t_o}^{t} \int_{t_o}^{t'} :-H_2(t''):\!:-H_2(t')\, dt''dt'$$

$$- \tfrac{1}{2} \int_{t_o}^{t} \int_{t_o}^{t'} :-H_2(t''):\!:-H_2(t')\, dt''dt'$$

$$= \int_{t_o}^{t} \int_{t_o}^{t'} :-H_2(t''):\!:-H_2(t'):\, dt''dt'$$

$$- \tfrac{1}{2} \int_{t_o}^{t} \int_{t_o}^{t'} \left(:-H_2(t''):\!:-H_2(t'):\, + :-H_2(t'):\!:-H_2(t''):\right) dt''dt'$$

$$= \tfrac{1}{2} \int_{t_o}^{t} \int_{t_o}^{t'} [:-H_2(t''):, :-H_2(t'):]\, dt''dt'$$

$$= \tfrac{1}{2} \int_{t_o}^{t} \int_{t_o}^{t'} :[-H_2(t''), -H_2(t')]:\, dt''dt'$$

or

$$g_{2,2} = \tfrac{1}{2} \int_{t_o}^{t} \int_{t_o}^{t'} [-H_2(t''), -H_2(t')]\, dt''dt' \tag{8.69}$$

and so one. All this should have a familiar ring to it: it is very much like the calculation of part a with a sequence of terms of δ-order, H_3, H_4, etc. In fact, there are some differences, but the process may be carried over.

The first major difference is that the δ-order factorization has separate exponents

$$N = e^{:g_3:} e^{:g_4:} \ldots \tag{8.70}$$

and the ε-order factorization, we wish to make a single exponent

$$e^{:g_2:} = e^{\varepsilon:g_{2,1}: + \varepsilon^2:g_{2,2}: + \ldots} \tag{8.71}$$

This may be remedied by first solving in factored form

$$e^{:g_2:} = e^{\varepsilon:g'_{2,1}:} e^{\varepsilon^2:g'_{2,2}:} \tag{8.72}$$

and then using the Baker-Campbell-Hausdorff formula to combine the $g'_{2,1}$ into a single exponent. The BCH series will terminate when the order of ε gets sufficiently high.

The second difference is that in this case, each successive term in powers of ε on the right side of (8.64) consists of one multiple integral, whereas each successive term in the powers of δ in (8.31) consists of one or more multiple integral summed; for example (8.36) consists of integral over both H_4^{int} and H_3^{int}. This may be remedied by disposing with all terms but the multiple integral required; each of the terms will be recognizable in the final answer. For example, if the

first term of (8.38) is removed, we will have the analogue of (8.69).

With these differences accounted for, the methods of part a will carry over. More to the point, the general method for finding arbitrary n^{th} ε-order polynomials (f_n, $n \geq 3$) given by Dragt and Forest may be carried over to find the ε-order polynomials $g_{2,i}$.

Whether or not the combination into a single exponent is feasible or possible, I shall metaphorically write M_2 as $e^{:g_2:}$.

c. H_1 Arbitrary and Geometric Considerations

If H_1 is arbitrary, i.e., not necessarily small, the situation is much different. In this case there are two possible strategies. The first is to try to split off the first-order part, i.e., try to find a G_1 and G_C such that

$$M = e^{:G_1:} e^{:G_C:}, \tag{8.73}$$

and then apply the techniques of Section a, or of Dragt and Forest [1983] to $e^{:G_C:}$. Unfortunately, there does not always exist such a G_C, and even if there does, it may not be easy to find.

The second strategy will always work. First, one finds the coefficients of the Taylor series of the map (1.3), and then integrates them as shown in Section 1g. The coefficients may be obtained in one of two ways. The first is by geometric considerations, i.e., knowing the trajectory of the particle in advance, one can compute its outgoing coordinates in terms of its incoming. For example, in a pure dipole

field, one knows the trajectory is circular. Therefore, from the angle and position of entry of a particle, and the poleface geometry of the magnet, one may calculate the angle and position of exit. With a little work, this may be converted into phase space variables and expanded to give the coefficients.

The other way of obtaining the coefficients is trying to solve the dynamical equation in M (8.9) directly. For example, if :H: commutes with itself at different times, (8.9) is solved by

$$M = e^{-\int_{t_o}^{t_f} :H(\zeta_o,t'): dt'} . \qquad (8.74)$$

It may be possible to perform this integral and compute its effects on phase space, then sum the series for the exponential in closed form to the order necessary.

A variation on this method is to use a known factorized map but alter the incoming map by adding a constant to one (or more) of the phase space variables. This we shall do, for instance, in Chapter 12 when discussing the midplane rotation.

9. Mispowered Normal-Entry Bending Magnet

This chapter and the next are concerned with a particular kind of accelerator error that gives rise to a first-order transformation, and that is the incorrect powering of dipole (bending) magnets. This chapter is a treatment of the normal-entry-and-exit bending magnet (NEBM), and the next chapter is a treatment of the various parallel face magnets, orbit correctors, and the general bending magnet. Although all computations are done for the mispowered magnets, the correctly powered magnet maps may be recovered by setting the mispowering parameter ε to 0.

In this chapter I present first two derivations of the mispowered normal-entry-and-exit bending magnet, the first based on the methods of Chapter 8 and the second based on geometric considerations and making use of the half-parallel-face magnet map given the next chapter.

a. Computation of the Map from the Hamiltonian

A normal-entry-and-exit bending magnet (normal-entry bend or NEBM for short) has the geometry shown in Figures 9.1 and 9.2. Its Hamiltonian and factored map were calculated by Douglas [1982]. I shall follow that calculation here, deviating, of course, from the ideal powering he assumed.

It is most convenient to describe this magnet in polar coordinates: ρ, p_ρ, y, p_y, ϕ, p_ϕ with t the independent variable. The radius is ρ, the radial momentum p_ρ. We shall eventually return to the familiar Cartesian coordinates. In this coordinate system, the Hamiltonian is

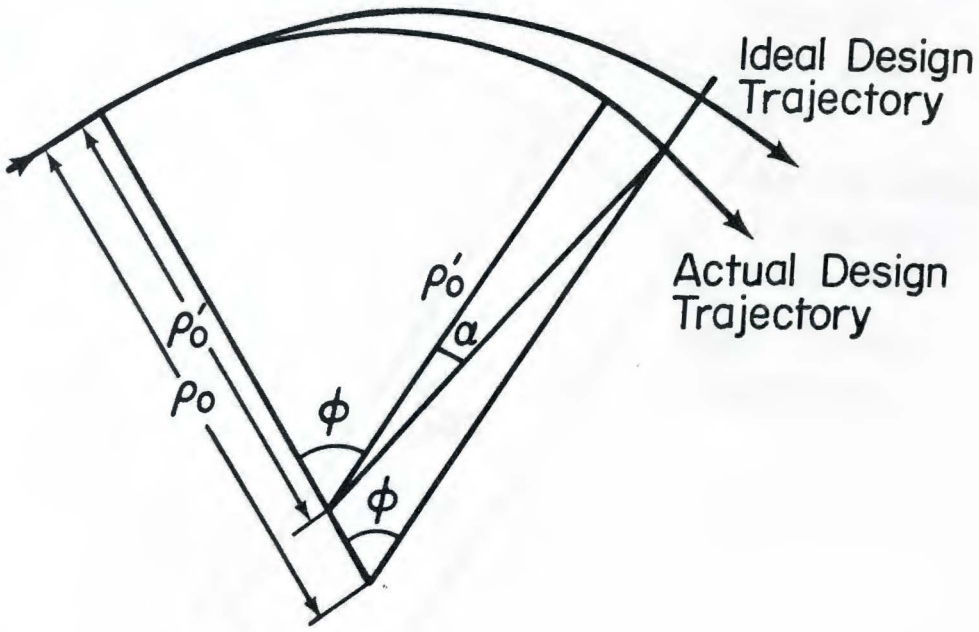


Figure 9.1 Geometry of the Mispowered Normal-Entry Bend
$B_{actual} > B_{ideal}$

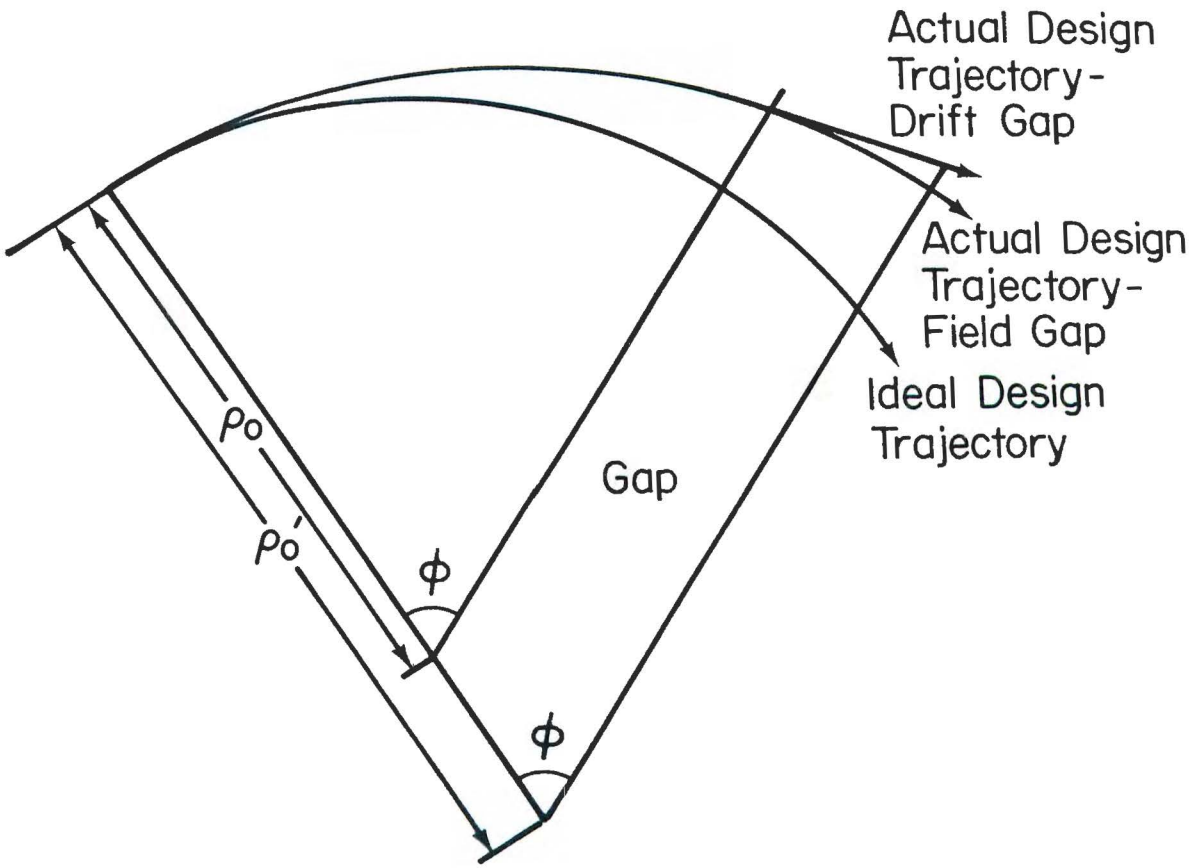


Figure 9.2 Geometry of a Mispowered Normal-Entry Bend
$B_{actual} < B_{ideal}$

$$H = c\sqrt{p_\rho^2 + \frac{(p_\phi - q\rho A_\phi)^2}{\rho^2} + p_y^2 + m^2c^2}\ . \qquad (9.1)$$

One obtains a new Hamiltonian by making the canonical transformation to ϕ as the independent variable and t, p_t as part of phase space,

$$\kappa = -p_\theta = -\frac{\rho}{c}\sqrt{p_t^2 - m^2c^4 - (p_\rho^2 + p_y^2)\,c^2} - q\rho A_\phi \qquad (9.2)$$

If we now replace the variables with ones that are small in deviation from the ideal design trajectory,

$$r \equiv \rho - \rho_o \ , \quad p_r \equiv p_\rho \ , \qquad (9.3)$$

$$T \equiv t - t_o \ , \quad P_T \equiv p_t - p_t^o \ ,$$

where ρ_o is the ideal design bending radius. The generating function of this transformation is

$$F \equiv [\,t - t^o(\phi)]\,[p_t^* + p_t^o], \qquad (9.4)$$

where the flight time along the ideal design trajectory is

$$t^o(\theta) = \frac{\phi\rho_o}{c\beta} = \frac{\phi\rho_o m\gamma}{p_o}\ , \qquad (9.5)$$

and $p_o = mc\beta\gamma$ is the design momentum. Then the new Hamiltonian, including the effects of the generating function, is

$$\kappa^{NEW} = -\frac{r + \rho_o}{c}\sqrt{(p_t^* + p_t^o)^2 - m^2c^4 - (p_r^2 + p_y^2)\,c^2} \qquad (9.6)$$

$$- \frac{\rho_o}{c\beta} (p_t^* + p_t^o) - q(r + \rho_o) A_\theta .$$

If we rescale to dimensionless variables as explained in Chapter 1,

$$R \equiv r/\ell \quad , \quad P_R = p_r/p_o , \tag{9.7}$$

$$Y \equiv r/\ell \quad , \quad P_Y = p_y/p_o ,$$

$$T \equiv t^*/\ell \quad , \quad P_T = p_t^*/p_o c ,$$

$$K = \frac{\kappa^{NEW}}{\ell p_o} ,$$

then in terms of the new variables, the Hamiltonian is

$$K = - \frac{R\ell + \rho_o}{cp_o\ell} \sqrt{(p_o c p_t + p_t^o)^2 - m^2c^4 - p_o^2c^2(P_R^2 + P_Y^2)} \tag{9.8}$$

$$- \frac{q}{\ell p_o} (R\ell + \rho_o) A_\theta - \frac{\rho_o}{c\beta} (p_o c P_T + p_t^o) \frac{1}{\ell p_o}$$

$$= - (R + \frac{\rho_o}{\ell}) \sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - P_R^2 - P_Y^2}$$

$$-q (R + \frac{\rho_o}{\ell}) \frac{A_\theta}{p_o} - \frac{\rho_o}{\beta\ell} P_T + \frac{\rho_o}{\ell\beta^2} .$$

For a pure bending magnet, the magnetic field is constant and points in the (±) Y direction. Thus the vector potential is

$$A_\phi = - \frac{1}{2} \rho B . \tag{9.9}$$

In the current set of coordinates, A_ϕ may be written

$$A_\phi = -\frac{1}{2}(R\ell + \rho_o)\, B. \tag{9.10}$$

Now, however, we must recall that B may not be as designed;

$$B_{actual} = (1 + \varepsilon)\, B_{ideal}, \tag{9.11}$$

where $|\varepsilon| \ll 1$, and ε may be positive or negative. Since the formula for A_ϕ involves B_{actual}, we may substitute (9.11) in (9.10),

$$A_\phi = -\frac{1}{2}(R\ell + \rho_o)(1 + \varepsilon)\, B_{ideal}. \tag{9.12}$$

Using the relation $p_o = \rho_o q B_{ideal}$, rewrite this as

$$A_\phi = -\frac{1}{2}(R\ell + \rho_o)(1 + \varepsilon)\frac{p_o}{\rho_o q}. \tag{9.13}$$

Thus, the Hamiltonian is now

$$K = -(R + \frac{\rho_o}{\ell})\sqrt{1 - \frac{2}{\beta}P_T + P_T^2 - P_R^2 - P_Y^2} \tag{9.14}$$

$$- \frac{\ell}{2\rho_o}(1 + \varepsilon)(R + \frac{\rho_o}{\ell})^2 - \frac{\rho_o}{\beta\ell}P_T + \frac{\rho_o}{\ell\beta^2}.$$

Expanding by order in phase space variables,

$$K = K_0 + K_1 + K_2 + K_3 + K_4 + \ldots, \tag{9.15}$$

where, ignoring K_0,

$$K_1 = H_1^{(1)} = -\varepsilon R \tag{9.16a}$$

$$K_2 = H_2^{(1)} = \frac{1}{\beta} RP_T + \frac{\lambda}{2\gamma^2\beta^2} P_T^2 + \frac{\lambda}{2} (P_R^2 + P_Y^2) + \frac{1+\varepsilon}{2\lambda} R^2 \tag{9.16b}$$

$$K_3 = H_3^{(1)} = \frac{1}{2\beta^2\gamma^2} RP_T^2 + \frac{1}{2} R(P_R^2 + P_Y^2) + \frac{\lambda}{2\beta^3\gamma^2} P_T^3 + \frac{\lambda}{2\beta} P_T(P_R^2 + P_Y^2) \tag{9.16c}$$

$$K_4 = H_4^{(1)} = \frac{1}{2\beta^3\gamma^2} RP_T^3 + \frac{1}{2\beta} RP_T(P_R^2 + P_Y^2) - \frac{\lambda}{8\beta^2\gamma^2} (1 - \frac{5}{\beta^2}) P_T^4$$

$$- \frac{\lambda}{4} (1 - \frac{3}{\beta^2}) P_T^2 (P_R^2 + P_Y^2) + \frac{\lambda}{8} (P_R^2 + P_Y^2)^2 , \tag{9.16d}$$

and $\lambda = \frac{\rho_o}{\ell}$.

Now we shall use the techniques of Chapter 8 to obtain the factored product expansion.

Let us first calculate the linear part of the map,

$$M_2^{(1)} = e^{-\int_o^\phi :K_2: \, d\phi'} , \tag{9.17}$$

as a matrix transformation on phase space. Since K_2 is independent of ϕ',

$$- \int_o^\phi :K_2: \, d\phi' = - \phi :K_2: . \tag{9.18}$$

The matrix that corresponds to $-\phi:K_2:$ is

$$JS = \phi \begin{bmatrix} 0 & \lambda & 0 & 0 & 0 & 0 \\ -\frac{1+\varepsilon}{\lambda} & 0 & 0 & 0 & 0 & -\frac{1}{\beta} \\ 0 & 0 & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{\beta} & 0 & 0 & 0 & 0 & \frac{1}{\gamma^2\beta^2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9.19}$$

Now the powers of JS are

$$(JS)^2 = \phi^2 \begin{bmatrix} -(1+\varepsilon) & 0 & 0 & 0 & 0 & -\frac{\lambda}{\beta} \\ 0 & -(1+\varepsilon) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\lambda}{\beta} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9.20}$$

and

$$(JS)^3 = \phi^3 \begin{bmatrix} 0 & -\lambda(1+\varepsilon) & 0 & 0 & 0 & 0 \\ \frac{(1+\varepsilon)^2}{\lambda} & 0 & 0 & 0 & 0 & -\frac{1+\varepsilon}{\beta} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1+\varepsilon}{\beta} & 0 & 0 & 0 & 0 & -\lambda/\beta^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{9.21}$$

In general, powers of JS have the recursion relation

$$JS^n = -(1+\varepsilon)\,\phi^2\,(JS)^{n-2} \quad , \quad n = 4, 5, 6, \ldots \tag{9.22}$$

Thus we may sum to all orders to calculate

$$M = e^{JS}, \tag{9.23}$$

$$M_{11} = M_{22} = 1 - \frac{\phi^2}{2!}(1+\varepsilon) + \frac{\phi^4}{4!}(1+\varepsilon)^2 \ldots = \cos\left(\phi\sqrt{1+\varepsilon}\right)$$

$$M_{16} = -M_{52} = -\frac{\lambda}{\beta}\left(\frac{\phi^2}{2!} - \frac{\phi^4}{4!}(1+\varepsilon) + \ldots\right)$$

$$= \frac{\lambda}{\beta(1+\varepsilon)}\left(\frac{\phi^2}{2!}(1+\varepsilon) - \frac{\phi^4}{4!}(1+\varepsilon)^2 + \ldots\right)$$

$$= \frac{\lambda}{\beta(1+\varepsilon)}\left[\cos\left(\phi\sqrt{1+\varepsilon}\right) - 1\right]$$

$$M_{12} = \lambda\left(1 - \frac{\phi^3}{3!}(1+\varepsilon) + \frac{\phi^5}{5!}(1+\varepsilon)^2 - \ldots\right)$$

$$= \frac{\lambda}{\sqrt{1+\varepsilon}}\left(\sqrt{1+\varepsilon} - \frac{\phi^3}{3!}\left(\sqrt{1+\varepsilon}\right)^3 + \frac{\phi^5}{5!}\left(\sqrt{1+\varepsilon}\right)^5 - \ldots\right)$$

$$= \frac{\lambda}{\sqrt{1+\varepsilon}} \sin(\phi\sqrt{1+\varepsilon})$$

$$M_{21} = -\frac{1+\varepsilon}{\lambda}\left(1 - \frac{\phi^3}{3!}(1+\varepsilon) + \frac{\phi^5}{5!}(1+\varepsilon^2) - \ldots\right)$$

$$= -\frac{\sqrt{1+\varepsilon}}{\lambda} \sin(\phi\sqrt{1+\varepsilon})$$

$$M_{26} = -M_{51} = -\frac{1}{\beta}\left(1 - \frac{\phi^3}{3!}(1+\varepsilon) + \frac{\phi^5}{5!}(1+\varepsilon)^2 - \ldots\right)$$

$$= -\frac{\lambda}{\beta\sqrt{1+\varepsilon}} \sin(\phi\sqrt{1+\varepsilon})$$

$$M_{34} = \lambda$$

$$M_{33} = M_{44} = 1$$

$$M_{55} = M_{66} = 1$$

$$M_{56} = \frac{\lambda}{\beta^2\gamma^2}\phi - \frac{\lambda}{\beta^2}\left(\frac{\phi^3}{3!} - \frac{\phi^5}{5!}(1+\varepsilon) + \frac{\phi^7}{7!}(1+\varepsilon)^2 - \ldots\right)$$

$$= \frac{\lambda}{\beta^2\gamma^2}\phi - \frac{\lambda}{\beta^2}\frac{1}{(\sqrt{1+\varepsilon})^3}\left(\frac{\phi^3}{3!}(1+\varepsilon)^{3/2} - \frac{\phi^5}{5!}(1+\varepsilon)^{5/2} + \ldots\right)$$

$$= \frac{\lambda}{\beta^2\gamma^2}\phi - \frac{\lambda}{\beta^2(1+\varepsilon)^{3/2}}\left[\phi\sqrt{1+\varepsilon} - \sin(\phi\sqrt{1+\varepsilon})\right]$$

$$= \frac{\lambda}{\beta^2}\left(\frac{1}{\gamma^2} - \frac{1}{1+\varepsilon}\right)\phi + \frac{\lambda}{\beta^2(1+\varepsilon)^{3/2}} \sin(\phi\sqrt{1+\varepsilon}).$$

Obviously, the measure of mispowering $\mu \equiv \sqrt{1+\varepsilon}$ is the most useful quantity here. Then

$$M(\phi) =$$

$$\begin{bmatrix} \cos(\mu\phi) & \frac{\lambda}{\mu}\sin(\mu\phi) & 0 & 0 & 0 & \frac{\lambda}{\beta\mu^2}[\cos(\mu\phi) - 1] \\ -\frac{\mu}{\lambda}\sin(\mu\phi) & \cos(\mu\phi) & 0 & 0 & 0 & -\frac{1}{\beta\mu}[\cos(\mu\phi) - 1] \\ 0 & 0 & 1 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{\beta\mu}\sin(\mu\phi) & -\frac{\lambda}{\beta\mu^2}[\cos(\mu\phi)-1] & 0 & 0 & 1 & \frac{\lambda}{\beta^2}(\frac{1}{\gamma^2} - \frac{1}{\mu^2})\phi + \frac{\lambda}{\beta^2\mu^2}\sin(\mu\phi) \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{9.24}$$

This gives Douglas's [1982] result as coded into MARYLIE in the case of no mispowering ($\mu = 1$).

The polynomials for the first iteration are given by the integral (8.43) of $-K_3^{tr}$, given by (8.42),

$$g_3^{(1)} = -\int_0^{\phi} K_3\,(M(\phi' - \phi)\,\zeta)\,d\phi' = -\int_{-\phi}^{0} K_3(M(\phi')\zeta)\,d\phi'. \tag{9.25}$$

The coefficients are listed in Table 9.1, and agree with the results of Douglas [1982] (Table 4.3) for the case $\varepsilon = 0$, $\mu = 1$. Furthermore, we may use for $g_4^{(1)}$ the fourth-order term of Douglas because we need keep no powers of ε, and thus may take $\varepsilon = 0$, $\mu = 1$. The coefficients of $g_4^{(1)}$ are given in Table 9.2.

The next step is to compute the polynomial $H^{(2)}$, using the rule (8.48) and (8.49). In this case, $H^{(2)}$ is

$$-H^{(2)}(\zeta) = -H_1^{(1)}\,(e^{:g_2^{(1)}:}\,e^{:g_3^{(1)}:}\,e^{:g_4^{(1)}:}\,\zeta) \tag{9.26}$$

$$= \varepsilon\, e^{:g_2^{(1)}:}\, e^{:g_3^{(1)}:}\, e^{:g_4^{(1)}:}\, R$$

$$= \varepsilon \left(e^{:g_2^{(1)}:} R + e^{:g_2^{(1)}:} [g_3^{(1)}, R] + \frac{1}{2} e^{:g_2^{(1)}:} [g_3^{(1)}, [g_3^{(1)}, R]] \right.$$

$$\left. + e^{:g_2^{(1)}:} [g_4^{(1)}, R] + \ldots \right).$$

Thus order by order, the terms $H_n^{(2)}$ are, with $C \equiv \cos(\mu\phi)$, $S \equiv \sin(\mu\phi)$,

$$-H_1^{(2)} = \varepsilon e^{:g_2^{(1)}:} R = \varepsilon \left(\cos(\mu\phi)\, R + \frac{\lambda}{\mu} \sin(\mu\phi)\, P_R + \frac{\lambda}{\beta\mu^2} [\cos(\mu\phi) - 1]\, P_T\right) \tag{9.27a}$$

$$-H_2^{(2)} = \varepsilon\, e^{:g_2^{(1)}:} [g_3^{(1)}, R] \tag{9.27b}$$

$$= \varepsilon \left(-\frac{1}{2\lambda} S^2 R^2 + \frac{1}{\mu} CS\, RP_R + \left(-\frac{1}{\beta\mu^2} S^2 - \frac{\varepsilon}{2\beta\mu} \phi S\right) RP_T \right.$$

$$+ \frac{\lambda}{2\mu} C(1 - C)\, P_R^2 + \left(\frac{\lambda}{2\beta\mu^3} S(\varepsilon + 2C) + \frac{\varepsilon\lambda}{2\beta\mu^2} \phi C\right) P_R P_T$$

$$\left. - \frac{\lambda}{2\mu^2} (1 - C)\, P_Y^2 - \left(\frac{\lambda}{2\beta^2\gamma^2\mu^2} (1 - C) + \frac{\lambda}{2\beta^2\mu^4} S^2 + \frac{\varepsilon\lambda}{2\beta^2\mu^3} \phi S\right) P_T^2 \right)$$

$$-H_3^{(2)} = \varepsilon\left(\frac{1}{2} e^{:g_2^{(1)}:} [g_3^{(1)}, [g_3^{(1)}, R]] + e^{:g_2^{(1)}:} [g_4^{(1)}, R]\right) \tag{9.27c}$$

$$= \varepsilon \left(-\frac{S^2}{2\beta\lambda} R^2 P_T - \frac{S^2}{2} RP_R^2 + \frac{SC}{\beta} RP_R P_T - \frac{S}{2} RP_Y^2 - \frac{(3 - \beta^2)\, S^2}{2} RP_T^2 \right.$$

$$+ P_T^2 + \frac{\lambda SC}{2} P_R^3 + \frac{\lambda(C - 1)}{2\beta} P_R^2 P_T + \frac{\lambda SC}{2} P_R^3 + \frac{\lambda(C - 1)}{2\beta} P_R^2 P_T$$

$$+ \frac{\lambda SC}{2} P_R P_Y^2 + \frac{\lambda(3 - \beta^2)\, SC}{2} P_R P_T^2 + \frac{\lambda(C - 1 - S^2)}{2\beta} P_Y^2 P_T$$

$$\left. + \left[\frac{\lambda(C - 1)}{2\beta^3\gamma^2} - \frac{\lambda(2 - \beta)^2 S^2}{2\beta^3}\right] P_T^3 \right)$$

$H_n^{(2)} = 0$ for $n \geq 4$, to the order we are concerned about.

Now we must solve (8.20) for $N_2^{(2)}$. Unfortunately, we may not use the exponential of the integral (8.23) because $H_2^{(2)}$ does not commute with itself at different "times" (values of ϕ). Therefore we shall attempt to solve for $N_2^{(1)}$ by solving the matrix differential equation (8.22) or (8.58),

$$\dot{M} = JSM \tag{8.22}$$

The matrix JS is formed from the coefficients of $H_2^{(2)}$ according to (8.57). A solution is given by the series (8.59a). Since we keep only two powers of ε in matrices, we may truncate the series (8.59a) after three terms. Let $N^{(2)}$ be the integral of $JS^{(2)}$,

$$N^{(2)}(\phi) = \int_0^\phi JS^{(2)}(\phi')\, d\phi'. \tag{9.28}$$

Then the value of $M^{(2)}$

$$M^{(2)} = I + N^{(2)}(\phi) + \int_0^\phi JS^{(2)}(\phi)\, N^{(2)}(\phi')\, d\phi' \tag{9.29}$$

will be sufficient for our purposes. The non-zero elements of $JS^{(2)}$ are given in Table 9.3, of $N^{(2)}$ in Table 9.4, and of $M^{(2)}$ in Table 9.5.

With the second linear transformation $M^{(2)}$ computed, we may now try to compute $g_3^{(2)'}$. We compute the prime version,

$$e^{:g_3^{(2)'}:}\, e^{:g_2^{(2)}:} = e^{:g_2^{(2)}:}\, e^{:g_3^{(2)}:} \tag{9.30}$$

to save the effort of transforming by the linear map. In order to compute $g_3^{(2)'}$, we will need

$$-H_3^{(2)^{int}}(\zeta,\phi) = -H_3^{(2)}(M^{(2)}\zeta,\phi). \tag{9.31}$$

Because $M^{(2)}$ is the identity plus a term of order ε, $H_3^{(2)}$ is of order ε, and we keep only terms of order ε in a third-order polynomial, we may say

$$-H_3^{(2)^{int}} = -H_3^{(2)} . \tag{9.32}$$

Thus

$$g_3^{(2)'} = \int_0^{\phi} -H_3^{(2)^{int}}(\zeta,\phi')\, d\phi' = \int_0^{\phi} -H_3^{(2)}(\zeta,\phi')\, d\phi'. \tag{9.33}$$

The results are given in Table 9.6.

The polynomial $g_4^{(2)}$ need not be calculated because it would be of order ε or greater. For fourth-order polynomials, no terms of order ε or greater are retained.

The next step is to find the new "Hamiltonian," $H^{(3)}$

$$H^{(3)} = H_1^{(2)}\left(e^{:g_3^{(2)'}:} e^{:g_2^{(2)}:} \zeta,\phi\right) \tag{9.34}$$

Factored, the second-order part is

$$-H_2^{(3)} = \varepsilon^2 \left[\frac{1}{2} C(\phi - CS)\, RP_R - \frac{1}{2\beta}\phi S\, RP_T - \frac{\lambda}{4}(2S^2C + \phi S)\, P_R^2 \right. \tag{9.35}$$

$$+ \frac{\lambda}{2\beta}(S^3 + 2\phi C - 2SC)\, P_R P_T - \frac{4}{\lambda}\phi S\, P_Y^2 + \frac{\lambda}{4}(\beta^2 - 3)\,\phi S\, P_T^2]$$

No linear transformation is necessary because it is proportional to ε^2, and $M^{(2)}$ is the identity plus order ε. Since we are keeping only two powers of ε for second-order polynomials, we may take $H_2^{(3)\,int} = H_2^{(3)}$. The equation (9.35) may be integrated to obtain the polynomial $g_2^{(3)}$. The coefficients are shown in Table 9.7. Exponentiating this polynomial is no trouble, because each term is proportional to ε^2. Thus one term will suffice:

$$N_2^{(3)} = e^{:g_2^{(3)}:} = I + :g_2^{(3)}: \qquad (9.36)$$

Since $H^{(3)}$ consisted of only first- and second-order terms, we are finally at the last step. The first order part

$$-H_1^{(3)}(\zeta,\phi) = -H_1^{(2)}(M^{(2)}\zeta,\phi) \qquad (9.37)$$

is long and disgusting, so it will not be given. The first-order transformation is determined by the equation (8.57)

$$g_1 = \int_0^\phi :H_1^{(3)}(\phi'):\, d\phi. \qquad (9.38)$$

The coefficients of this polynomial are given in Table 9.8.

To summarize, the map for the mispowered normal-entry-and-exit bending magnet is given by

$$= e^{:g_1:}\, e^{:g_2^{(3)}:}\, e^{:g_3^{(2)'}:}\, e^{:g_2^{(2)}:}\, e^{:g_2^{(1)}:}\, e^{:g_3^{(1)}:}\, e^{:g_4^{(1)}:}. \qquad (9.39)$$

The second-order transformations are kept as matrices. All these matrices and polynomials are given in (9.24) and tables 9.1 through 9.8. In these tables $\varepsilon \equiv \Delta B/B$ is the mispowering, $\mu \equiv \sqrt{1+\varepsilon}$ and $C \equiv \cos(\mu\phi)$, $S \equiv \sin(\mu\phi)$.

Putting (9.39) into the standard factorization would not be terribly difficult. The two second-order transformations $e^{:g_2^{(2)}:}$, $e^{:g_2^{(1)}:}$ would have to be moved to the left by using the transformation rule. Then the two third-order transformations $e^{:g_3^{(2)'T}:}$ $e^{:g_3^{(1)}:}$ would be adjacent (the first transformed). They could be combined in a single exponent by adding, since the Poisson bracket $[g_3^{(2)'T}, g_3^{(1)}]$ is too high in total order to worry about. The three linear transformations would be left as separate matrices.

Table 9.1 Nonzero Coefficients of $g_3^{(1)}$ for the Mispowered Normal-Entry Bending Magnet

Note: $S = \sin(\mu\phi)$
$C = \cos(\mu\phi)$

R^3	(28)	:	$-\frac{\mu}{6\lambda^2} S^3$
R^2P_R	(29)	:	$-\frac{1}{2\lambda} S^2C$
R^2P_T	(33)	:	$-\frac{1}{2\beta\lambda\mu} S^3 - \frac{\varepsilon}{4\beta\lambda}(\phi - \frac{SC}{\mu})$
RP_R^2	(34)	:	$-\frac{1}{2\mu} SC^2$
RP_RP_T	(38)	:	$-\frac{1}{\beta\mu^2} S^2 (C + \frac{\varepsilon}{2})$
RP_Y^2	(43)	:	$-\frac{1}{2\mu} S$
RP_T^2	(48)	:	$-\frac{1}{2\beta^2\mu^3} S^3 - \frac{\varepsilon}{2\beta^2\mu^2}(\phi - \frac{SC}{\mu}) - \frac{1}{2\beta^2\gamma^2\mu} S$
P_R^3	(49)	:	$\frac{\lambda}{6\mu^2}(1 - C^3)$
$P_R^2P_T$	(53)	:	$-\frac{\lambda}{2\beta\mu^3} SC^2 - \frac{\lambda}{4\beta\mu^2}\varepsilon(\phi + \frac{SC}{\mu})$
$P_RP_Y^2$	(58)	:	$\frac{1}{2\mu^2}(1 - C)$
$P_RP_T^2$	(63)	:	$\frac{\lambda}{2\beta^2\gamma^2\mu^2}(1 - C) - \frac{\lambda}{2\beta^2\mu^4} S^2(C + \varepsilon)$
$P_Y^2P_T$	(76)	:	$-\frac{\lambda}{2\beta\mu^3} S - \frac{\lambda}{2\beta\mu^2}\varepsilon\phi$
P_T^3	(83)	:	$-\frac{\lambda}{2\beta^3\mu^4}(\frac{S^3}{3\mu} + \frac{\varepsilon}{2}(\phi - \frac{SC}{\mu})) - \frac{\lambda}{2\beta^3\gamma^2\mu^2}(\frac{S}{\mu} + \varepsilon\phi)$

Table 9.2 Coefficients of $g_4^{(1)}$ for the Mispowered Normal-Entry Bending Magnet

R^3P_T (89) : $-\frac{1}{6\beta\lambda^2}S^3$

$R^2P_R^2$ (90) : $-\frac{1}{8\lambda}S^3$

$R^2P_RP_T$ (94) : $-\frac{1}{2\beta\lambda}S^2C$

$R^2P_Y^2$ (99) : $-\frac{1}{8\lambda}S^3$

$R^2P_T^2$ (104) : $-\frac{5-\beta^2}{8\beta^2\lambda}S^3$

RP_R^3 (105) : $-\frac{1}{4}S^2C$

$RP_R^2P_T$ (109) : $\frac{1}{4\beta}S^3-\frac{1}{2\beta}S$

$RP_RP_Y^2$ (114) : $-\frac{1}{4}S^2C$

$RP_RP_T^2$ (119) : $-(\frac{1}{4\beta^2\gamma^2}+\frac{1}{\beta^2})S^2C$

$RP_Y^2P_T$ (132) : $-\frac{1}{4\beta}S^3-\frac{1}{2\beta}S$

RP_T^3 (139) : $-(\frac{1}{4\beta^3\gamma^2}+\frac{1}{2\beta^3})S^3-\frac{1}{2\beta^2\gamma^2}S$

P_R^4 (140) : $-\frac{\lambda}{8}SC^2$

$P_R^3P_T$ (144) : $-\frac{\lambda}{12\beta}S^2C+\frac{\lambda}{6\beta}(1-C)$

$P_R^2P_Y^2$ (149) : $\frac{\lambda}{8}S^3-\frac{\lambda}{4}S$

$P_R^2 P_T^2$ (154) : $\frac{\lambda(4 - \beta^2)}{8\beta^2} S^3 - \lambda \left(\frac{1}{4\beta^2\gamma^2} + \frac{1}{2\beta^2}\right) S$

$P_R P_Y^2 P_T$ (167) : $- \frac{\lambda}{4\beta} S^2 C + \frac{\lambda}{2\beta} (1 - C)$

$P_R P_T^3$ (174) : $-\lambda \left(\frac{1}{4\beta^3\gamma^2} + \frac{1}{2\beta^3}\right) S^2 C + \frac{\lambda}{2\beta^3\gamma^2} (1 - C)$

P_Y^4 (195) : $\frac{\lambda}{8} S$

$P_Y^2 P_T^2$ (200) : $- \frac{\lambda}{8\beta^2} S^3 - \lambda \left(\frac{1}{4\beta^2\gamma^2} + \frac{1}{2\beta^2}\right) S$

P_T^4 (209) : $-\lambda \left(\frac{1}{8\beta^4\gamma^2} + \frac{1}{6\beta^4}\right) S^3 - \lambda \left(\frac{1}{2\beta^4\gamma^2} + \frac{1}{8\beta^4\gamma^4}\right) S$

Table 9.3 Nonzero Elements of $JS^{(2)}$ for the Mispowered Normal-Entry Bending Magnet

$$[JS^{(2)}]_{21} = -\frac{\varepsilon}{\lambda} S^2$$

$$-[JS^{(2)}]_{11} = [JS^{(2)}]_{22} = \frac{\varepsilon}{\mu} CS$$

$$[JS^{(2)}]_{26} = -[JS^{(2)}]_{51} = -\frac{\varepsilon}{\beta\mu^2} S^2 - \frac{\varepsilon^2}{2\beta\mu} \phi S$$

$$[JS^{(2)}]_{12} = -\frac{\lambda\varepsilon}{\mu} C(1 - C)$$

$$[JS^{(2)}]_{16} = [JS^{(2)}]_{52} = -\frac{\varepsilon\lambda}{2\beta\mu^3} (S + \mu\phi C) - \frac{\lambda}{\beta\mu^3} SC$$

$$[JS^{(2)}]_{34} = \frac{\lambda\varepsilon}{\mu^2} (1 - C)$$

$$[JS^{(2)}]_{56} = \frac{\lambda\varepsilon}{\beta^2\gamma^2\mu^2} (1 - C) + \frac{\lambda\varepsilon}{\beta^2\mu^4} S^2 + \frac{\lambda\varepsilon^2}{\beta^2\mu^2} \phi S$$

Table 9.4 Nonzero Elements of $N^{(2)}$ for the Mispowered Normal-Entry Bending Magnet

$$N_{21}^{(2)} = -\frac{\varepsilon}{2\lambda}\left(\phi - \frac{CS}{\mu}\right)$$

$$-N_{11}^{(2)} = N_{22}^{(2)} = \frac{\varepsilon}{2\mu^2} S^2$$

$$N_{26}^{(2)} = -N_{51}^{(2)} = -\frac{\varepsilon}{2\beta\mu^2}\phi(1 - \varepsilon C) - \frac{\varepsilon}{2\beta\mu^3} S(\varepsilon - C)$$

$$N_{12}^{(2)} = \frac{\lambda\varepsilon}{2\mu^2}\phi - \frac{\lambda\varepsilon}{2\mu^3} S(2 - C)$$

$$N_{16}^{(2)} = N_{52}^{(2)} = -\frac{\lambda\varepsilon}{2\beta\mu^4} S^2 - \frac{\lambda\varepsilon^2}{2\beta\mu^3}\phi S$$

$$N_{34}^{(2)} = \frac{\lambda\varepsilon}{\mu^2}\left(\phi - \frac{S}{\mu}\right)$$

$$N_{56}^{(2)} = \frac{\lambda\varepsilon}{\beta^2\gamma^2\mu^2}\left(\phi - \frac{S}{\mu}\right) + \frac{\lambda\varepsilon}{2\beta^2\mu^4}\phi(1 - 2\varepsilon C) + \frac{\lambda\varepsilon}{2\beta^2\mu^5} S(2\varepsilon - C)$$

Table 9.5 Non-Identity Elements of $M^{(2)}$ for the Mispowered Normal-Entry Bending Magnet

$$M^{(2)}_{1,1} = 1 + \varepsilon\left(-\frac{5}{12} + \frac{C}{2} + \frac{\phi S}{2} - \frac{\phi SC}{4} - \frac{C^2}{8} + \frac{C^3}{6} - \frac{C^4}{8} - \frac{\phi^2}{8} - \frac{S^2}{2}\right)$$

$$+ \varepsilon^2\left(\frac{3}{4} - \frac{3C}{4} - \frac{\phi S}{2} + \frac{\phi SC}{4} - \frac{C^2}{16} - \frac{C^3}{4} + \frac{5C^4}{16} + \frac{\phi^2}{16} + \frac{S^2}{2}\right)$$

$$M^{(2)}_{1,2} = \varepsilon\lambda\left(\frac{9\phi}{16} - S - \frac{S^3}{6} + \frac{7SC}{16} + \frac{S^3C}{8}\right)$$

$$+ \varepsilon^2\lambda\left(\frac{25\phi}{32} + \frac{3S}{2} + \frac{2S^3}{3} - \frac{11SC}{16} - \frac{3S^2C}{8} + \frac{\phi C^2}{4}\right)$$

$$M^{(2)}_{1,6} = \frac{\varepsilon\lambda}{\beta}\left(-\frac{5}{12} + \frac{C}{2} + \frac{\phi S}{2} - \frac{\phi SC}{4} - \frac{C^2}{8} + \frac{C^3}{6} - \frac{C^4}{8} - \frac{\phi^2}{8} - \frac{S^2}{2}\right)$$

$$+ \frac{\varepsilon^2\lambda}{\beta}\left(\frac{71}{72} - \frac{11C}{12} + \frac{\phi^2}{16} + S^2 - \frac{5C^2}{16} - \frac{7C^3}{36} + \frac{7C^4}{16} - \phi S - \frac{\phi S^3}{6} + \frac{\phi SC}{4}\right)$$

$$M^{(2)}_{2,1} = \frac{\varepsilon}{\lambda}\left(-\frac{\phi}{2} + \frac{SC}{2}\right) + \frac{\varepsilon^2}{\lambda}\left(\frac{1}{3} - \frac{C}{2} - \frac{\phi}{16} - \frac{7SC}{16} + \frac{S^3C}{8} + \frac{\phi C^2}{4} + \frac{C^3}{6}\right)$$

$$M^{(2)}_{2,2} = 1 + \frac{\varepsilon S^2}{2} + \varepsilon^2\left(\frac{1}{8} + \frac{\phi}{2} - \frac{S}{2} + \frac{\phi C}{2} - \frac{SC}{2} - \frac{C^2}{4} + \frac{C^4}{8} - \frac{S^2}{2} - \frac{S^3}{6}\right)$$

$$M^{(2)}_{2,6} = \frac{\varepsilon}{2\beta}(SC - \phi) + \frac{\varepsilon^2}{\beta}\left(\frac{1}{3} - \frac{C}{2} + \frac{7\phi}{16} - \frac{S}{2} + \frac{\phi C}{2} - \frac{15SC}{16} + \frac{S^3C}{8} + \frac{\phi C^2}{4} + C^3\right)$$

$$M^{(2)}_{3,4} = \varepsilon\lambda(\phi - S) + \varepsilon^2\lambda\left(\frac{3S}{2} - \phi\right)$$

$$M^{(2)}_{5,1} = \frac{\varepsilon}{\beta}\left(\frac{9\phi}{16} - \frac{5SC}{16} - \frac{S^3C}{8} - \frac{\phi C^2}{4}\right) + \frac{\varepsilon^2}{\beta}\left(-\frac{13\phi}{16} + \frac{S}{2} - \frac{\phi C}{2} + \frac{15SC}{32} + \frac{7S^3C}{16} + \frac{\phi C^2}{2}\right.$$

$$M^{(2)}_{5,2} = \frac{\varepsilon\lambda}{\beta}\left(-\frac{1}{8} - \frac{S^2}{2} + \frac{C^2}{4} - \frac{C^4}{8}\right) + \frac{\varepsilon^2\lambda}{\beta}\left(-\frac{1}{24} + C + \frac{\phi^2}{8}\right.$$

$$+ S^2 - \frac{9C^2}{8} - \frac{C^3}{3} + \frac{C^4}{2} - \frac{\phi S}{2} - \frac{\phi SC}{4})$$

$$M_{5,6}^{(2)} = \frac{\varepsilon\lambda}{\beta^2}(\frac{\phi}{\gamma^2} + \frac{9\phi}{16} - \frac{S}{\gamma^2} - \frac{5SC}{16} - \frac{S^3C}{8} - \frac{\phi C^2}{4})$$

$$+ \frac{\varepsilon^2\lambda}{\beta^2}(-\frac{\phi}{\gamma^2} - \frac{11\phi}{8} + \frac{3S}{2\gamma^2} + \frac{5S}{6} + \frac{2S^3}{9} - \phi C$$

$$+ \frac{25SC}{32} + \frac{9S^3C}{16} + \frac{3\phi C^2}{4} + \frac{\phi C^3}{6})$$

Table 9.6 Coefficients of $g_3^{(2)'}$ for the Mispowered Normal-Entry Bending Magnet

R^2P_T	(33)	:	$\frac{\varepsilon}{4\beta\lambda}(SC - \phi)$
RR_R^2	(34)	:	$\frac{\varepsilon}{4}(SC - \phi)$
RP_RP_T	(38)	:	$\frac{\varepsilon}{2\beta}S^2$
RP_Y^2	(43)	:	$\frac{\varepsilon}{4}(SC - \phi)$
RP_T^2	(48)	:	$\frac{\varepsilon}{4}(3 - \beta^2)(SC - \phi)$
P_R^3	(49)	:	$\frac{\varepsilon\lambda}{4}S^2$
$P_R^2P_T$	(53)	:	$\frac{\varepsilon\lambda}{2\beta}(S - \phi)$
$P_RP_Y^2$	(58)	:	$\frac{\varepsilon\lambda}{4}S^2$
$P_RP_T^2$	(63)	:	$\frac{\varepsilon\lambda}{4}(3 - \beta^2)\,S^2$
$P_Y^2P_T$	(76)	:	$\frac{\varepsilon\lambda}{4\beta}(-3\phi + 2S + SC)$
P_T^3	(83)	:	$\frac{\varepsilon\lambda}{2\beta^3\gamma^2}(S - \phi) + \frac{\varepsilon\lambda}{4\beta^3}(2 - \beta^2)(SC - \phi)$

Table 9.7 Coefficients of $g_2^{(3)}$ for the Mispowered Normal-Entry Bending Magnet

RP_R	(8)	:	$\frac{\varepsilon^2}{6}(-4 + 3C + 3\phi S + C^3)$
RP_T	(12)	:	$\frac{\varepsilon^2}{2\beta}(\phi C - S)$
P_R^2	(13)	:	$\frac{\varepsilon^2\lambda}{12}(3\phi C - 2S^3 - 3S)$
$P_R P_T$	(17)	:	$\frac{\varepsilon^2\lambda}{6\beta}(-7 + 3C + 6\phi S + 3C^2 + C^3)$
P_Y^2	(22)	:	$\frac{\varepsilon^2\lambda}{4}(\phi C - S)$
P_T^2	(27)	:	$\frac{\varepsilon^2\lambda}{4}(3 - \beta^2)(\phi C - S)$

Table 9.8 Coefficients of g_j for the Mispowered Normal-Entry Bending Magnet

$$X : \varepsilon S + \varepsilon^2 \left(\frac{7\phi}{16} - \frac{3S}{2} + \frac{\phi C}{4} + \frac{23SC}{48} - \frac{S^3C}{24} - \frac{\phi C^2}{4} + \frac{\phi C^3}{12} - \frac{\phi^2 S}{8} + \frac{11S^3}{72} - \frac{S^5}{40}\right)$$
$$+ \varepsilon^3 \left(\frac{1}{8} - \frac{C}{3} - \frac{21\phi}{32} + \frac{103S}{48} - \frac{\phi C}{16} - \frac{23SC}{24} + \frac{S^3C}{12}\right.$$
$$\left. + \frac{3\phi C^2}{8} - \frac{5\phi C^3}{24} + \frac{\phi^2 S}{8} + \frac{C^2}{4} - \frac{C^4}{24} - \frac{19S^3}{48} + \frac{S^5}{10}\right)$$

$$P_X : \varepsilon\lambda(1 - C) + \varepsilon^2\lambda \left(-\frac{193}{120} + \frac{17C}{16} + \frac{7C^2}{12} - \frac{C^3}{48} - \frac{C^4}{24} + \frac{C^5}{40} + \frac{9\phi S}{16}\right)$$
$$+ \varepsilon^3\lambda \left(\frac{1711}{720} - \frac{125C}{96} - \frac{3\phi}{16} + \frac{S}{2} - \frac{S^3}{6} - \frac{11C^2}{8} + \frac{65C^3}{288}\right.$$
$$\left. + \frac{3C^4}{16} - \frac{9C^5}{80} - \frac{\phi C}{2} + \frac{7SC}{16} + \frac{S^3C}{24} - \frac{13\phi S}{16} - \frac{\phi S^3}{12} - \frac{\phi C^2}{4}\right)$$

$$P_T : \frac{\varepsilon^2\lambda}{\beta} \left(\frac{7}{16}\phi - S + \frac{11S^3}{72} - \frac{S^5}{40} + \frac{\phi C}{4} + \frac{23SC}{48} - \frac{S^3C}{24} - \frac{\phi^2 S}{8} - \frac{\phi C^2}{4} + \frac{\phi C^3}{12}\right)$$
$$+ \frac{\varepsilon^3\lambda}{\beta} \left(\frac{1}{8} - \frac{C}{3} - \frac{191\phi}{192} + \frac{343S}{144} - \frac{191S^3}{48} + \frac{S^5}{8} + \frac{C^2}{4} - \frac{C^4}{24}\right.$$
$$\left. - \frac{9\phi C}{16} - \frac{455SC}{576} + \frac{17S^3C}{288} + \frac{\phi^2 S}{8} + \frac{11\phi C^2}{24} - \frac{5\phi C^3}{24} - \frac{\phi C^4}{24}\right)$$

b. Computation of the Map from Ideal Elements and Coordinate Transformations

It is possible to view the transfer map of the mispowered normal-entry-and-exit bending magnet, with bending angle α and actual bending radius ρ_o' as a succession of three maps (Figure 9.1). Consider first the case where the actual field is greater than the ideal. The first map is a normal-entry-and-exit bending magnet, correctly powered so that it has the same bending radius $\rho_o' < \rho_o$ for the design trajectory. The second is a correctly powered trailing half parallel face magnet (HPF) with bending radius ρ_o' and angle α. The half parallel face magnet is discussed in the next section. The third is a coordinate-transformation map at the trailing face,

$$M_{T_T}(\Delta T)\, M_{R_Y}(\alpha)\, M_{T_X}(\Delta X). \tag{9.40}$$

This is necessary because the actual design trajectory emerges at a different X and P_X coordinate and the two trajectories differ in flight time. Since we want deviation from the ideal and we have deviation from the actual, coordinate shifts in X, P_X and T are necessary. Once ρ_o', α, ΔX, and ΔT are known in terms of the ideal magnet parameters ρ_o, the ideal bending radius, ϕ, the bending angle, and the mispowering $\varepsilon \equiv \Delta B/B$, the maps may be computed in terms of these parameters and concatenated either analytically or numerically by the techniques of Chapter 4.

The case where the actual field is less than the ideal field (Figure 9.2) is essentially the same. Here the second map should be the inverse of a leading HPF magnet. It is readily verified that this is a

trailing HPF with negative angle. Thus, if the computation of α respects sign, the computation can be included in the one above. Likewise, the coordinate transformation map will be correct if α and ΔX respect sign.

With these considerations in mind, we may proceed to calculate ρ_o', α and ΔX. Recall that $\varepsilon = \Delta B/B_{ideal}$, $\Delta B = B_{actual} - B_{ideal}$ and $\rho_o = B\rho/B_{ideal}$ and $B\rho$ is the magnetic rigidity, a property of the machine. Then the actual bending radius is given by

$$\rho_o' = \frac{B\rho}{B_{actual}} = \frac{B\rho}{B_{ideal}} \frac{B_{ideal}}{B_{actual}} = \rho_o \frac{1}{1 + \varepsilon} . \qquad (9.41)$$

A close-up of the geometry at the apex (Figure 9.4) shows that the angle α is related to p_o, ρ_o' and ϕ by

$$(\rho_o - \rho_o') \cos (\phi - \tfrac{\pi}{2}) = \rho_o' \sin \alpha, \qquad (9.42)$$

or

$$\sin \alpha = \frac{\rho_o - \rho_o'}{\rho_o'} \sin \phi = \left(\frac{\rho_o}{\rho_o'} - 1\right) \sin \phi = \varepsilon \sin \phi. \qquad (9.43)$$

Thus $\alpha = \arcsin(\varepsilon \sin \phi)$, which will be negative if ε is, as we require.

To compute ΔX, consider the triangle at the apex (Figure 9.3). The law of sines relates the sides with their opposite angles,

$$\frac{\ell}{\sin(\pi - \alpha - \phi)} = \frac{\rho_o'}{\sin \phi} , \qquad (9.44)$$

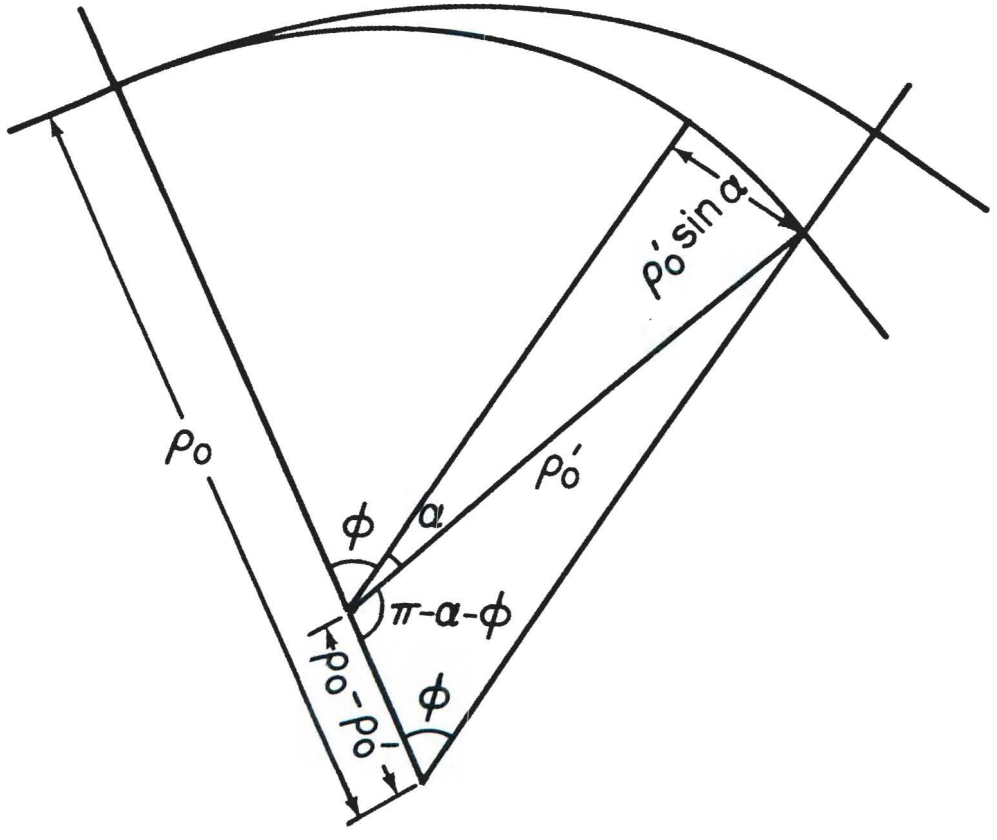


Figure 9.3 *Geometric Quantities in a Mispowered Normal-Entry Bending Magnet*

where $\ell = \rho_o - \Delta X$ is the distance from the apex to the exit point of the actual design trajectory. Solving for ℓ,

$$\ell = \frac{\sin(\alpha + \phi)}{\sin \phi} \rho_o' \tag{9.45}$$

$$= \rho_o' \frac{\sin \alpha \cos \phi + \cos \alpha \sin \phi}{\sin \phi}$$

$$= \rho_o \frac{\sqrt{1 - \varepsilon^2 \sin^2 \phi} + \varepsilon \cos \phi}{1 + \varepsilon}$$

so

$$\Delta X = \rho_o - \ell = \rho_o \left(1 - \frac{\sqrt{1 - \varepsilon^2 \sin^2 \phi} + \varepsilon \cos \phi}{1 + \varepsilon}\right). \tag{9.46}$$

Note that $\ell = \rho_o$ $(\Delta X = 0)$ when

$$\sqrt{1 - \varepsilon^2 \sin^2 \rho} + \varepsilon \cos \phi = 1 + \varepsilon \tag{9.47}$$

$$1 - \varepsilon^2 \sin^2 \phi = \left(1 + \varepsilon(1 - \cos \phi)\right)^2 = 1 + 2\varepsilon(1 - \cos \phi) + \varepsilon^2 (1 + \cos \phi)^2 \tag{9.48}$$

$$2\varepsilon (1 + \varepsilon)(1 - \cos \phi) = 0 \tag{9.49}$$

If the quantity on the left is less than 0, ΔX will less than zero, if it is greater, ΔX will be greater. The former will happen when $\varepsilon > 0$, the latter when $\varepsilon < 0$, because ε is small so that $1 + \varepsilon$ can never change sign, and ϕ is always positive and $\cos \phi$ never exceeds 1.

Finally, to compute ΔT, we must consider the two path lengths for

the design particle. For the ideally powered magnet, the path length is just the product of the bending radius and the angle,

$$s_{ideal} = \rho_o \phi. \tag{9.50}$$

For the mispowered magnet, the path length is the sum of the paths in the correctly powered NEBM of bending radius ρ_o' and the HPF which define the first two maps,

$$s_{actual} = \rho_o' \, (\alpha + \phi). \tag{9.51}$$

The difference in path length is $\Delta S = S_{actual} - S_{ideal}$,

$$\Delta S = \rho_o \left(\frac{\alpha + \phi}{1 + \varepsilon} - \phi \right) = \frac{\rho_o}{1 + \varepsilon} (\alpha - \varepsilon\phi). \tag{9.52}$$

Since both design particles move at the same constant speed $c\beta$, the difference in flight times is just the difference in path length divided by this speed,

$$\Delta T = \frac{\Delta S}{c\beta} = \frac{1}{c\beta} \frac{\rho_o}{1 + \varepsilon} (\alpha - \varepsilon\phi). \tag{9.53}$$

If this value ΔT is added to the time coordinate T measured relative to the actual design particle, the result will be the flight time measured relative to the ideal design particle. This change in time coordinate is accomplished by a first-order transformation proportional to ΔT,

$$M_{T_T}(\Delta T) = e^{-\Delta T : P_T :}. \tag{9.54}$$

In summary, the mispowered normal-entry-and-exit bending magnet of angle ϕ, bending radius ρ_o and fractional mispowering $\varepsilon = \frac{B_{actual} - B_{ideal}}{B_{ideal}}$ may be represented as three maps: First, a normal-entry-and-exit bending magnet of bending angle ϕ and radius ρ_o', a trailing HPF of angle α and radius ρ_o', and coordinate changes using the maps $M_{T_T}(\Delta T)$, $M_{R_Y}(\alpha)$, and $M_{T_X}(\Delta X)$, where

$$\rho_o' = \frac{\rho_o}{1 + \varepsilon} \tag{9.55}$$

$$\alpha = \arcsin(\varepsilon \sin \phi) \tag{9.56}$$

$$\Delta X = \rho_o \left(1 - \frac{\sqrt{1 - \varepsilon^2 \sin^2 \phi} + \varepsilon \cos \phi}{1 + \varepsilon}\right) \tag{9.57}$$

$$\Delta T = \frac{1}{c\beta} \frac{\rho_o}{1 + \varepsilon} (\alpha - \varepsilon\phi). \tag{9.58}$$

10. Mispowered Parallel-Face Magnets and General Bending Magnet

In this chapter, I calculate the maps for mispowered dipoles that have non-normal entry or exit, the parallel face magnets, and the general bending magnet. The first section treats the parallel-face bending magnets, which include the symmetric parallel-face magnet, and what I call the half-parallel-face magnets, which are explained below. The second section is a computation of an orbit corrector, which can be considered a mispowered zero-strength parallel-face bending magnet. As such, the computation is just a special case of those of the first section. The third part considers the effect of fringe fields in a mispowered dipole magnet, including normal-entry or -exit dipoles. Finally, the last section shows how to construct the map for a general bending magnet, one with arbitrary entry, exit and bending angles, out of the maps obtained in the earlier sections.

a. Parallel-Face Magnets

A parallel-face bending magnet (PFBM) is a dipole of uniform constant field, its entry and exit pole faces parallel, and the angle of entry and exit formed by the design trajectory and the normal to the pole faces equal. If a parallel-face magnet is cut half way down the axis, each part is a half-parallel-face magnet (HPF), either leading or trailing. An HPF has parallel faces, but is either normal-exit (leading HPF) or normal-entry (trailing HPF). The HPF will prove useful for construction of the general bending magnet (see Section 10e), and, when ideally powered, to compute the map of a mispowered normal entry bend (see Section 9b). Figure 10.1 is an illustration of these bending

magnets.

The derivations of the maps for the mispowered parallel magnets are all similar, differing only in the integration limits. Therefore, they are treated together in this section. This treatment includes the body only. To obtain the complete map, there must be concatenated at the pole faces the map for a fringe field and a map for a pole face (midplane) rotation, equal to the (half) bend angle α:

$$M = M_{\text{prot}} M_{\text{fringe}} M_{\text{body}} M_{\text{fringe}} M_{\text{prot}} \qquad (10.1)$$

Assuming a hard-edge fringe field, the pole face (midplane) rotations will be in a field-free region, and therefore unaffected by the powering of the magnet, so we may use the maps in Douglas [1982] or in Chapter 12, exact in the angle. The maps for the mispowered hard-edge fringe fields are covered in section d. Once all these maps have been computed, they may be concatenated numerically or analytically using the techniques of Chapter 4.

I shall suppose throughout this section that the magnet has a bending angle of 2α for the design trajectory in the ideally powered magnet (α for the HPF magnets), and an ideal bending radius ρ_o. If the magnetic field is B_{actual} instead of B_{ideal}, the fractional mispowering is

$$\varepsilon \equiv \frac{B_{\text{actual}} - B_{\text{ideal}}}{B_{\text{ideal}}} , \qquad (10.2)$$

which may be either positive or negative, and is presumed to be small $|\varepsilon| \ll 1$. The mispowered magnet has a design bending radius

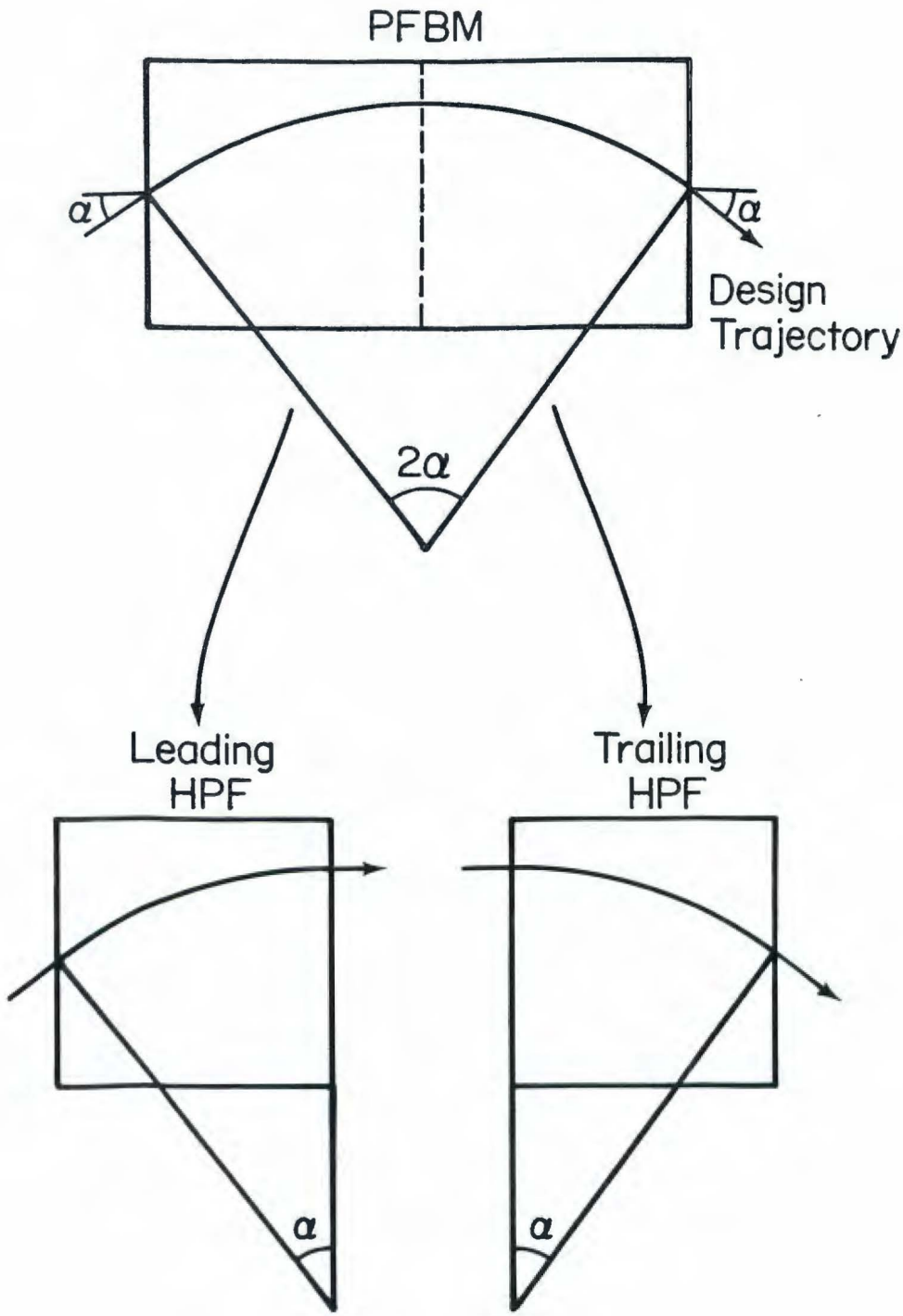


Figure 10.1 The Parallel-Face Bending Magnet, and the Half-Parallel-Face Magnets

$$\rho_o' = \frac{B\rho}{B_{actual}} = \frac{B\rho}{B_{ideal}(1 + \varepsilon)} = \frac{\rho_o}{1 + \varepsilon} \qquad (10.3)$$

where $B\rho$ is the magnetic rigidity of the beam.

In the parallel-face magnets, the mechanical momenta Π_x and Π_z are not small for the design trajectory. However, at mid-magnet for the parallel face, or at the leading or trailing face for the HPF, where the design trajectory is perpendicular to the faces, i.e. $\Pi_x = 0$, we may set the vector potential to be zero. If the magnet has (half) length L,

$$\vec{A} = \hat{x}B(z - \eta) \qquad (10.4)$$

where z is measured as shown in Figure 10.2.

The canonical momentum p_x differs from the mechanical by qA_x,

$$p_x = \Pi_x + qA_x = \Pi_x + qB(z - \eta), \qquad (10.5)$$

We choose the quantity η so that $\Pi_x = 0$ for the design trajectory at $z = \eta$. This implies that $p_x = 0$ at this point, but since p_x is conserved, it is zero everywhere. From Figure 10.3, we see that this happens at

$$\eta = \rho_o' \sin\alpha - \rho_o \sin\alpha \qquad (10.6)$$

$$= \rho_o \left(\frac{1}{1 + \varepsilon} - 1 \right) \sin\alpha = -\frac{\varepsilon \rho_o \sin\alpha}{1 + \varepsilon} \qquad (10.7)$$

for the leading HPF or the full parallel-face magnet. For the trailing HPF, the design trajectory is normal at entry by assumption, so $\eta = 0$.

The Hamiltonian is

$$H = -p_t = \sqrt{[p_x - qB(z - \eta)]^2 + p_y^2 + p_z^2 + m^2c^4} \ . \qquad (10.8)$$

Transforming this to the phase space x, p_x, y, p_y, t, p_t, gives the Hamiltonian

$$\kappa = -p_z = -\sqrt{p_t^2 - [p_x - qB(z - \eta)]^2 - p_y^2 - m^2c^4} \ . \qquad (10.9)$$

Now let us transform to coordinates and momenta near the design trajectory. In particular, κ and t must be measured from the design values, as these change during the flight of the design particle. First, consider the design flight time. The value of z is given, as a function of time, by

$$z = \rho_o \sin\left(\frac{c\beta}{\rho_o} t\right) \qquad (10.10)$$

where ρ_o = ideal design bending radius, $\beta = \frac{v}{c}$, and α the (half) bend angle. The zero of z is taken to be the midpoint of the full parallel bend. Inverting this relation gives the absolute time as a function of z,

$$t^o(z) = \frac{\rho_o}{c\beta} \arcsin\left(\frac{z}{\rho_o}\right). \qquad (10.11)$$

Note that in the limit $\rho_o = \infty$ (no field), this gives the expected result

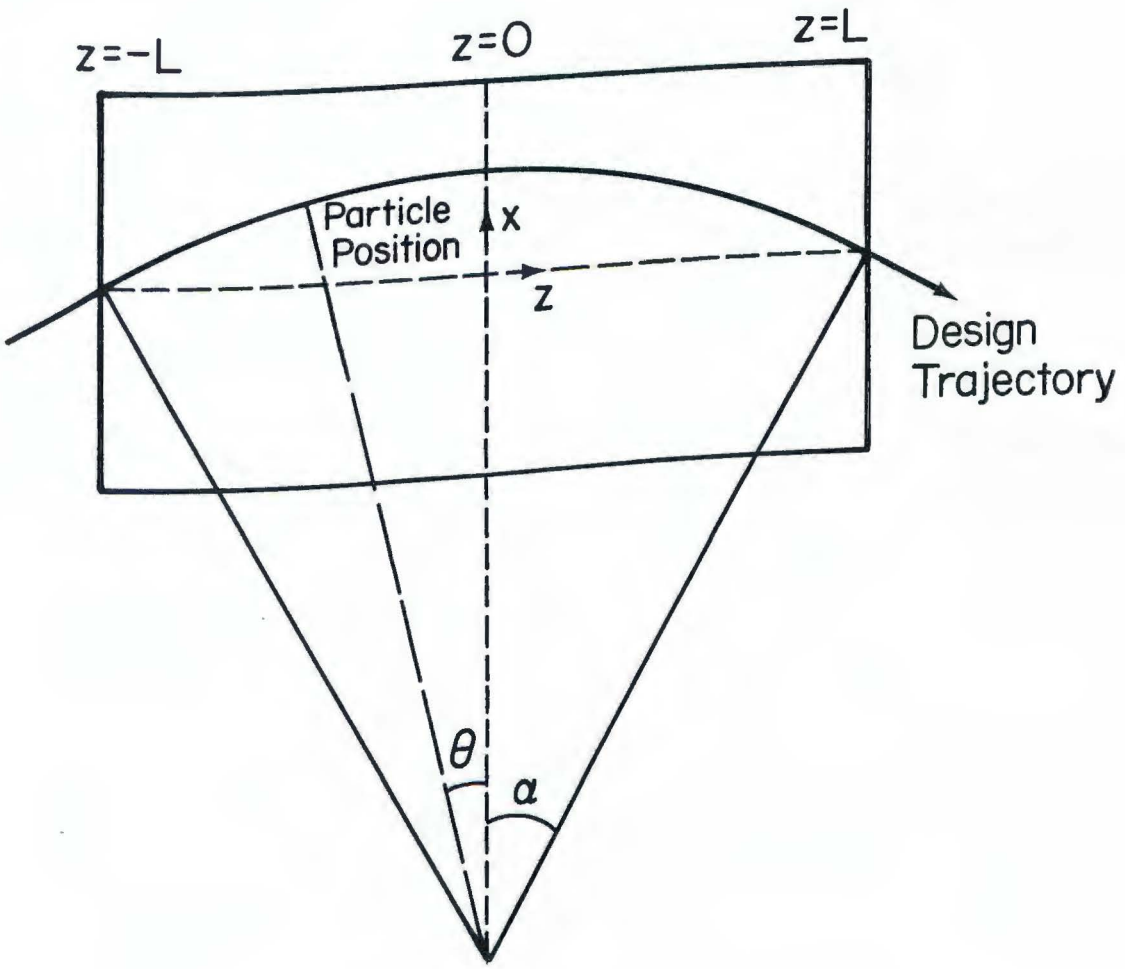


Figure 10.2 Geometry of the Parallel-Face Magnets, Ideally Powered

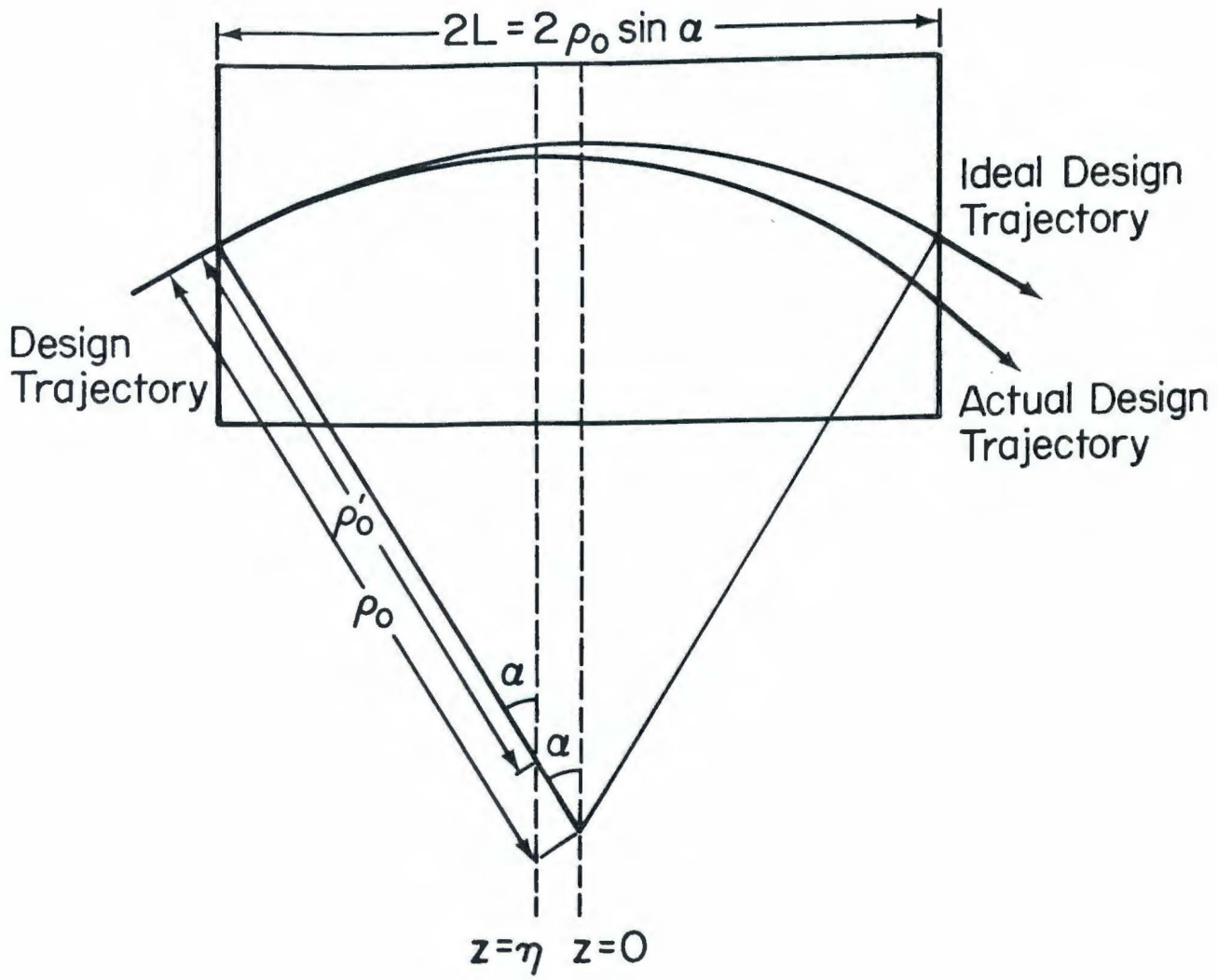


Figure 10.3 Determination of η for Mispowered Parallel-Face Magnets

$$t^o(z) = \frac{z}{c\beta} \,. \tag{10.12}$$

The value of the x coordinate is given by

$$x^o(z) = \rho_o \,(\cos\,\theta - \cos\,\alpha) \tag{10.13}$$

$$= \sqrt{\rho_o^2 - z^2} - \rho_o \cos\,\alpha.$$

Again, in the limit $\rho_o = \infty$ we get the expected value $x^o(z) = 0$.

The new variables will be

$$x^* = x - x^o = x - \left(\sqrt{\rho_o^2 - z^2} - \rho_o \cos\,\alpha\right) \tag{10.14}$$

and

$$t^* = t - t^o = t - \frac{\rho_o}{c\beta} \arcsin\left(\frac{z}{\rho_o}\right). \tag{10.15}$$

The corresponding design momentum for t is the negative of the energy

$$p_t^o \equiv -\gamma mc^2, \tag{10.16}$$

so the new momentum is

$$p_t^* = p_t - p_t^o \,. \tag{10.17}$$

Since the design value of the x momentum p_x is 0, we need not change that.

We now must use the generating function to get the new Hamiltonian κ^{NEW},

$$\kappa^{NEW} = \kappa + \frac{\partial F}{\partial z} \tag{10.18}$$

$$F(x,p_x^*,t,p_t^*;z) = [t - t^o][p_t^* + p_t^o] + [x - x^o]p_x \tag{10.19}$$

$$\frac{\partial F}{\partial z} = -\frac{1}{c\beta}\frac{1}{\sqrt{1 - \left(\frac{z}{p_o}\right)^2}}(p_t^* - \gamma mc^2) + \frac{p_x}{\sqrt{\rho_o^2 - z^2}}, \tag{10.20}$$

so

$$\kappa^{NEW} = -\sqrt{(P_t^* - \gamma mc^2)^2 - [p_x - qB(z - \eta)]^2 - p_y^2 - m^2c^4} \tag{10.21}$$

$$-\frac{1}{c\beta}\frac{1}{\sqrt{1 - \left(\frac{z}{\rho_o}\right)^2}}(p_t^* - \gamma mc^2) + \frac{p_x}{\sqrt{\rho_o^2 - z^2}}.$$

Now let's scale the momenta and Hamiltonian using (1.85). The new Hamiltonian is

$$K = -\sqrt{\frac{(p_t^* - \gamma mc^2)^2}{p_o^2} - [p_x - qB(z - \eta)]^2 - p_y^2 - m^2c^4} \tag{10.22}$$

$$-\frac{1}{c\beta}\frac{1}{\sqrt{1 - \left(\frac{z}{\rho_o}\right)^2}}\left(\frac{p_t^* - \gamma mc^2}{p_o}\right) + \frac{zp_x}{p_o\sqrt{\rho_o^2 - z^2}}.$$

Using the relation

$$\frac{(p_t^* - \gamma mc^2)^2}{p_o^2} - \frac{m^2c^4}{p_o^2} = 1 - \frac{2}{\beta}P_T + P_T^2, \tag{10.23}$$

K becomes

$$K = -\sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - \left[P_X - \frac{qB}{p_o}(z - \eta)\right]^2 - P_Y^2} \qquad (10.24)$$

$$- \frac{1}{\sqrt{1 - \left(\frac{z}{\rho_o}\right)^2}} \left(\frac{1}{\beta} P_T - \frac{\gamma mc^2}{p_o c\beta}\right) + \frac{zP_x}{\sqrt{\rho_o^2 - z^2}}$$

For convenience, let $Z = z/\rho_o$. Then, ignoring constants in K,

$$K = -\sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - \left[P_X - \frac{qB\rho_o}{p_o}\left(Z - \frac{\eta}{\rho_o}\right)\right]^2 - P_Y^2} \qquad (10.25)$$

$$- \frac{\frac{1}{\beta} P_T - ZP_X}{\sqrt{1 - Z^2}} .$$

The constant that multiples the Z in the large square root may be rewritten using the relation $qB_{ideal}\rho_o = p_o$,

$$\frac{qB\rho_o}{p_o} = \frac{qB_{ideal}\rho_o}{p_o} \frac{B_{actual}}{B_{ideal}} = \frac{B_{actual}}{B_{ideal}} = 1 + \varepsilon \qquad (10.26)$$

where

$$\varepsilon \equiv \frac{\Delta B}{B} = \frac{B_{actual} - B_{ideal}}{B_{ideal}} \qquad (10.27)$$

is the fractional mispowering of the magnet. Then

$$K = -\sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - \left[P_X - (1 + \varepsilon)\left(Z - \frac{\eta}{\rho_o}\right)\right]^2 - P_Y^2} \qquad (10.28)$$

$$- \frac{\frac{1}{\beta} P_T - Z P_X}{\sqrt{1 - Z^2}} .$$

The next step is to do an order-by-order expansion of K in the small phase space quantities. It is important to remember, however, that Z is

not a small phase space quantity, so it is useful to rewrite K as

$$K = -\sqrt{1 - (1+\varepsilon)^2 (Z - \frac{\eta}{\rho_o})}\sqrt{1 + \frac{-\frac{2}{\beta} P_T + 2(1+\varepsilon)(Z - \frac{\eta}{\rho_o}) P_x + P_x^2 - P_Y^2}{1 - (1+\varepsilon)^2 (Z - \frac{\eta}{\rho_o})^2}} - \frac{\frac{1}{\beta} P_T - ZP_X}{\sqrt{1 - Z^2}} . \qquad (10.29)$$

Expanding K in the small quantities P_X, P_Y, P_T, we obtain K_1, K_2, K_3, and K_4. They are given in Table 10.1.

In order to find g_2, g_3, and g_4, we shall have to integrate over z. Since w is related to z by

$$w = (1+\varepsilon)(Z - \frac{\eta}{\rho_o}) = \frac{1+\varepsilon}{\rho_o} z + \varepsilon \sin\alpha, \qquad (10.30)$$

the integrals are over

$$\int dw = \frac{\rho_o}{1+\varepsilon} \int dz. \qquad (10.31)$$

The limits of integration should correspond to the z values for leading HPF ($-\rho_o \sin\alpha$, 0), for trailing HPF (0, $\rho_o \sin\alpha$), and for the full PFBM ($-\rho_o \sin\alpha$, $\rho_o \sin\alpha$). Table 10.2 summarizes the values of w.

Table 10.1 Expansion of the Hamiltonian K for the Parallel-Face Magnets

Note: $w = (1 + \varepsilon)(Z - \frac{\eta}{\rho_o})$

$$K_1 = \frac{1}{\beta}\left(\frac{1}{\sqrt{1 - w^2}} - \frac{1}{\sqrt{1 - Z^2}}\right) P_T + \left(\frac{Z}{\sqrt{1 - Z^2}} - \frac{w}{\sqrt{1 - w^2}}\right) P_X$$

$$K_2 = \frac{w^2}{2(1 - w^2)^{3/2}} P_x^2 - \frac{w}{\beta(1 - w^2)^{3/2}} P_X P_T - \left(\frac{1}{2\sqrt{1-w^2}} - \frac{1}{2\beta^2(1-w^2)^{3/2}}\right) P_T^2$$

$$+ \frac{P_X^2}{2\sqrt{1 - w^2}} + \frac{P_Y^2}{2\sqrt{1 - w^2}}$$

$$K_3 = -\left(\frac{w}{2(1 - w^2)^{3/2}} + \frac{w^3}{2(1 - w^2)^{5/2}}\right) P_X^3$$

$$+ \frac{1}{2\beta(1 - w^2)^{3/2}} P_T (P_X^2 + P_Y^2) + \frac{3w^2}{2\beta(1 - w^2)^{5/2}} P_X^2 P_T$$

$$- \frac{w}{2(1 - w^2)^{3/2}} P_X P_Y^2 + \left[-\frac{3w}{2\beta^2(1 - w^2)^{5/2}} + \frac{w}{2(1 - w^2)^{5/2}}\right] P_X P_T^2$$

$$- \frac{1}{2\beta}\left(\frac{1}{(1 - w^2)^{3/2}} - \frac{1}{\beta^2(1 - w^2)^{5/2}}\right) P_T^3$$

$$K_4 = \left(\frac{1}{8(1-w^2)^{3/2}} + \frac{3w^2}{4(1-w^2)^{5/2}} + \frac{5w^4}{8(1-w^2)^{7/2}}\right) P_X^4$$

$$- \left(\frac{3w}{2\beta(1-w^2)^{5/2}} + \frac{5w^3}{2\beta(1-w^2)^{7/2}}\right) P_X^3 P_T$$

$$+ \left(\frac{1}{4(1-w^2)^{3/2}} + \frac{3w^2}{4(1-w^2)^{5/2}}\right) P_X^2 P_Y^2$$

$$- \left(\frac{1}{4(1-w^2)^{3/2}} - \frac{3}{4\beta^2(1-w^2)^{5/2}} + \frac{3w^2}{4(1-w^2)^{5/2}} - \frac{15w^2}{4\beta^2(1-w^2)^{7/2}}\right) P_X^2 P_T^2$$

$$- \frac{3w}{2\beta(1-w^2)^{5/2}} P_X P_Y^2 P_T - \left(- \frac{3w}{2\beta(1-w^2)^{5/2}} + \frac{5w}{2\beta^3(1-w^2)^{7/2}}\right) P_X P_T^3$$

$$+ \frac{1}{8(1-w^2)^{3/2}} P_Y^4 + \left(\frac{3}{4\beta^2(1-w^2)^{5/2}} - \frac{1}{4(1-w^2)^{3/2}}\right) P_Y^2 P_T^2$$

$$+ \left(\frac{1}{8(1-w^2)^{3/2}} - \frac{3}{4\beta^2(1-w^2)^{5/2}} + \frac{5}{8\beta^4(1-w^2)^{7/2}}\right) P_T^4$$

Table 10.2 Limits of Integration for the Parallel-Face Magnets

Magnet	z_{entry}	z_{exit}	η	w_{entry}	w_{exit}
Lead HPF	$-\rho_o \sin\alpha$	0	$-\frac{\varepsilon\rho_o \sin\alpha}{1+\varepsilon}$	$-\sin\alpha$	$\varepsilon \sin\alpha$
Trail HPF	0	$\rho_o \sin\alpha$	0	0	$(1+\varepsilon)\sin\alpha$
Full PFBM	$-\rho_o \sin\alpha$	$\rho_o \sin\alpha$	$-\frac{\varepsilon\rho_o \sin\alpha}{1+\varepsilon}$	$-\sin\alpha$	$(1+2\varepsilon)\sin\alpha$
Steering	0	L	0	0	$\frac{qB}{p_o}$ L

Table 10.3 Integrals for Evaluating Parallel-Face Magnet Maps

	Integrand	$\int$ dw evaluated at w = sin A
Even:	$\frac{1}{\sqrt{1 - w^2}}$	A
	$\frac{1}{(1 - w^2)^{3/2}}$	tan A
	$\frac{w^2}{(1 - w^2)^{3/2}}$	tan A - A
	$\frac{1}{(1 - w^2)^{5/2}}$	$\tan A + \frac{1}{3} \tan^3 A$
	$\frac{w^2}{(1 - w^2)^{5/2}}$	$\frac{1}{3} \tan^3 A$
	$\frac{1}{(1 - w^2)^{7/2}}$	$\tan A + \frac{2}{3} \tan^3 A + \frac{1}{5} \tan^5 A$
	$\frac{w^2}{(1 - w^2)^{7/2}}$	$\frac{1}{3} \tan^3 A + \frac{1}{5} \tan^5 A$
	$\frac{w^4}{(1 - w^2)^{7/2}}$	$\frac{1}{5} \tan^5 A$
Odd:	$\frac{w}{\sqrt{1 - w^2}}$	-cos A
	$\frac{w}{(1 - w^2)^{3/2}}$	sec A
	$\frac{w}{(1 - w^2)^{5/2}}$	$\frac{1}{3} \sec^3 A$
	$\frac{w}{(1 - w^2)^{7/2}}$	$\frac{1}{5} \sec^5 A$
	$\frac{w^3}{(1 - w^2)^{5/2}}$	$-\sec A + \frac{1}{3} \sec^3 A$
	$\frac{w^3}{(1 - w^2)^{7/2}}$	$-\frac{1}{3} \sec^3 A + \frac{1}{5} \sec^5 A$

The values of the integrals over w that will be needed are given in Table 10.3. They are evaluated at w = sin A, so that, to evaluate a given integral, one need only subtract the value at A = arcsin(w_{entry}) from A = arcsin(w_{exit}).

Note that all terms in K are momenta; thus, the Poisson bracket of one with another is zero. This makes life very simple: to calculate the polynomials g_n in the transfer map, we merely integrate $-K_n$ over the appropriate range.

The final step is to reconcile the discrepancy in the x momentum at the exit face. For the ideally powered magnet, the mechanical momentum of the design trajectory at the exit face is

$$\Pi_x^{ideal} = -qB_{ideal}\,(z_{exit} - \eta). \tag{10.32}$$

Immediately after z_{exit} on the outside of the magnet, the field is zero. Ignoring the hard-edge fringe field (section d), which does not affect the design trajectory, the momentum just outside is

$$p_x^{ideal}\,\Pi_x^{ideal} = -qB_{ideal}(z_{exit} - \eta). \tag{10.33}$$

Since we have agreed that the coordinates shall be deviations from the ideal design trajectory, this quantity must be subtracted. In the calculation of the finite-angle midplane rotation (Chapter 12 or Douglas [1982]) which is concatenated after the parallel-face body, one actually restores it before rotating, but we are not concerned with that here. What does concern us is that actual design trajectory will have a slightly different momentum

$$p_x^{actual} = \Pi_x^{actual} = -qB_{actual}(z_{exit} - \eta). \qquad (10.34)$$

Thus we must make a correction to the x momentum, so that we are measuring relative to the ideal design trajectory. In terms of the dimensionless momentum P_X the correction is

$$\Delta P_x = -(\Pi_x^{ideal} - \Pi_x^{actual})/p_o \qquad (10.35)$$

$$= -q(B_{ideal} - B_{actual})(z_{exit} - \eta)/p_o$$

$$= -q\,\Delta B\,(z_{exit} - \eta)/p_o.$$

This is effected by a first-order transformation proportional to X,

$$e^{-\frac{q\Delta B}{p_o}(z_{exit} - \eta)\ :X:}, \qquad (10.36)$$

which comes after the rest of the parallel-face body:

$$M_{PFBody} = e^{:g_1:}e^{:g_2:}e^{:g_3:}e^{:g_4:}e^{\frac{-q\Delta B}{p_o}(z_{exit} - \eta)\ :X:}. \qquad (10.37)$$

The transformations may be concatenated using the techniques of Chapter 4.

One may approximate the map of a mispowered parallel-face magnet by ignoring the effects that arise from a finite length of the magnet, i.e., to view the mispowering as a simple kick in momentum. In this case, the g_n would be taken as that of a correctly powered magnet and (10.37) would be

$$M_{PFBodyKick} = M_{PFBody}\, e^{-\frac{q\Delta B}{p_o}(z_{exit} - \eta)\,:X:} \quad . \qquad (10.38)$$

Table 10.4 Matrix and Polynomials for Mispowered Parallel-Face Magnets

Substitute the values $A_u = \arcsin(w_{exit})$, $A_\ell = \arcsin(w_{enter})$ from Table 10.2 for the particular magnet desired.

$$M = \frac{\rho_o}{1+\varepsilon}\begin{bmatrix} 1 & \tan A_u - \tan A_\ell & 0 & 0 & 0 & -\frac{1}{\beta}(\sec A_u - \sec A_\ell) \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & A_u - A_\ell & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{\beta}(\sec A_u - \sec A_\ell) & 0 & 0 & 1 & \frac{\tan A_u - \tan A_\ell}{\beta^2} - A_u + A_\ell \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Values given for polynomials g_3, g_4 should be evaluated at $A = A_\ell$, then subtracted from the value at $A = A_u$. All terms should be multipled by $\frac{\rho_o}{1+\varepsilon}$ to obtain coefficient

$$g_1 = \left(-\cos A_u + \cos A_\ell - \left[\sqrt{1 - \left(\frac{z_{exit}}{\rho_o}\right)^2} - \sqrt{1 - \left(\frac{z_{enter}}{\rho_o}\right)^2}\right]\right) P_X$$

$$- \frac{1}{\beta}\left(A_u - A_\ell - \left[\arcsin\left(\frac{z_{exit}}{\rho_o}\right) - \arcsin\left(\frac{z_{enter}}{\rho_o}\right)\right]\right) P_T$$

g_3

P_X^3 (49) : $\frac{1}{6} \sec^3 A$

$P_X^2 P_T$ (53) : $-\frac{1}{2\beta} \tan A \sec^2 A$

$P_X P_Y^2$ (58) : $\frac{1}{2} \sec A$

$P_X P_T^2$ (63) : $\frac{1}{2} \sec A \left(\frac{\sec^2 A}{\beta^2} - 1\right)$

$P_Y^2 P_T$ (76) : $-\frac{1}{2\beta} \tan A$

P_T^3 (83) : $-\frac{1}{2\beta^3} \tan A \left(\frac{1}{\gamma^2} + \frac{1}{3} \tan^2 A\right)$

g_4

P_X^4 (140) : $-\frac{1}{8} \tan A \sec^4 A$

$P_X^3 P_T$ (144) : $\frac{1}{2\beta} \sec^5 A - \frac{1}{3\beta} \sec^3 A$

$P_X^2 P_Y^2$ (149) : $-\frac{1}{4} \tan A \sec^2 A$

$P_X^2 P_T^2$ (154) : $\frac{1}{4} \tan A \sec^2 A - \frac{3}{4\beta^2} \tan A \sec^4 A$

$P_X P_Y^2 P_T$ (167) : $\frac{1}{2\beta} \sec^3 A$

$P_X P_T^3$ (174) : $-\frac{1}{2\beta} \sec^3 A + \frac{1}{2\beta^3} \sec^5 A$

P_Y^4 (195) : $-\frac{1}{8} \tan A$

$P_Y^2 P_T^2$ (200) : $-\frac{1}{4\beta^2} [\tan A \sec^2 A + \frac{1}{\gamma^2} \tan A + \tan A]$

P_T^4 (209) : $\frac{1}{4} \left[-\frac{1}{\beta^4} \left(\frac{5}{2} \tan A + \frac{5}{3} \tan^3 A + \frac{1}{2} \tan^5 A\right)\right.$

$\left. + \frac{1}{\beta^2} \tan A (3 + \tan^2 A) - \frac{1}{2} \tan A\right]$

c. The Steering Magnet

A steering magnet, or orbit corrector, is a weak adjustable parallel-faced dipole placed in the beamline to correct for the misalignment of other elements. For the purposes of computing its map, it may be thought of as a mispowered zero-strength parallel-face bending magnet (Figure 10.4). As such, the analysis in the previous section holds for the steering magnet. We need a slight modification in the notation, however, because the design magnetic field is zero, and the design bending radius infinite.

The design trajectory enters as shown in Figure 10.4, and gets bent one way or the other, or not at all, depending on the field applied. The mechanical momentum Π_x is zero at the entry face, $z = 0$. Thus, in the analysis of the last section, we may set $\eta = 0$. Furthermore, as mentioned in the last section, the design flight time will be

$$t^o(z) = \frac{z}{c\beta} , \tag{10.40}$$

and the x coordinate will be

$$x^o(z) = 0. \tag{10.41}$$

The Hamiltonian K given for the parallel-face magnet, (10.24), may then be used, ignoring constants and taking $\rho_o \to \infty$,

$$K = -\sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - (P_X - \frac{qB}{\rho_o} z)^2 - P_Y^2} - \frac{1}{\beta} P_T. \tag{10.42}$$

The part first order in P_X, P_Y, P_T is

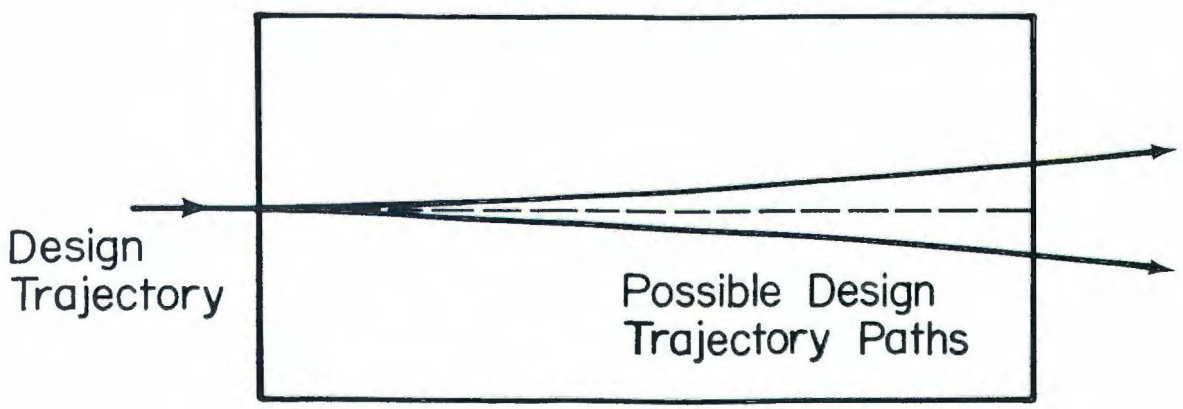


Figure 10.4 The Steering Magnet

$$K_1 = \frac{1}{\beta}\left(\frac{1}{\sqrt{1 - w^2}} - 1\right) P_T - \frac{w}{\sqrt{1 - w^2}} P_X \qquad (10.43)$$

and K_2, K_3, K_4 are as given in the last section, except w is interpreted as

$$w = \frac{qB}{p_o} z. \qquad (10.44)$$

When integrated, the limits are z = 0 to z = L, the length of the magnet. The matrix and polynomials are as given in Table 10.4, using the appropriate limits as stated in that table, and $w_{entry} = 0$, $w_{exit} = \frac{qB_o}{p_o} L$.

The final momentum change is given by the transformation

$$e^{-q\Delta B(z_{exit} - \eta) :X:} = e^{-\frac{qB}{p_o} L:X:} = e^{-\frac{L}{\rho_o'} :X:} \qquad (10.45)$$

where ρ_o' = actual design bending radius.

d. The Hard-Edge Fringe Field of a Mispowered Magnet

In any analysis of bending magnets, one must include the fringe fields at the entry and exit pole faces. Douglas [1982] and Dragt [1982b] have computed the Hamiltonian and map for the hard-edge fringe field of an ideally powered bending magnet. I shall analyze the case of a mispowered magnet.

The transformed Hamiltonian for an arbitrary vector potential in the x direction is

$$\kappa = -\sqrt{p_t^2 - [p_x - qA_x(y,z)]^2 - p_y^2 - m^2c^4} \ . \qquad (10.46)$$

With an appropriate model of the fringe field, we should factor this and integrate it over the region of the fringe field. In order to be consistent with Maxwell's equations, we may write the vector potential as

$$A_x(y,z) = b_{-1}(z) - \frac{1}{2} b_1(z)\, y^2 - \frac{1}{4} b_3(z)\, y^4 \ldots \qquad (10.47)$$

where

$$b_1(z) = \int a_o(z)\, dz \qquad (10.48a)$$

$$b_1(z) = a_o'(z) \qquad (10.48b)$$

$$b_3(z) = -\frac{1}{6} a_o'''(z) \text{ etc.} \qquad (10.48c)$$

In this case, a_o is the part of B_y independent of y. In the hard-edge model, we assume that the effect of the fringe field occurs over zero length in z, that is, it rises from 0 to the full field inside the magnet B in zero length, consistent with Maxwell's equations. We will sandwich these fringe fields maps around the magnet body map derived above.

Since the field rises in zero length, we use

$$a_o(z) = B\theta(L - z)\, \theta(L + z) \qquad (10.49)$$

where θ is the step function and the magnet extends from z = -L to z =

+L (although this illustration is with a parallel face-magnet, this fact is not essential to the result). Then

$$b_{-1} = c, \text{ determined by the design entry angle} \qquad (10.50a)$$

$$b_1 = a_o'(z) = \pm B\delta(L \pm z) \qquad (10.50b)$$

$$b_3 = \frac{1}{6} a_o'''(z) = \mp \frac{1}{6} B\delta'''(L \pm z) \qquad (10.50c)$$

where the upper sign is for the leading edge, the lower sign is for the trailing edge. The Hamiltonian becomes

$$\kappa = -\sqrt{p_t^2 - [p_x - qB(c \mp \frac{1}{2}\delta(L \pm z)\, y^2 \pm \frac{1}{6}\delta'''(L \pm z)\, y^4 \ \ldots]^2 - p_y^2 - m^2c^4} \qquad (10.51)$$

Note that when the term in brackets in this expression is expanded, terms like y^2, y^4, etc. and $p_x y^2$, $p_x y^4$, etc. will be produced. These terms will be proportional to B. If we write $B = B_{ideal} + \Delta B$, the terms caused by mispowering, proportional to ΔB, will be proportional to y^2, y^4, $p_x y^2$, $p_x y^4$, etc. Thus there are no first-order terms in the mispowered case. These are the only terms that are of concern in the expansion, since the integral will be over an infinitesimal length.

Since no first-order terms are produced, we need not pay particular attention to mispowering. Simply computing the transfer map with the actual magnetic field B is adequate. After scaling, these polynomials are as given by Douglas [1982] for the entry face

$$g_2 = -\frac{\ell}{2\rho_o'} \tan\alpha\, Y^2 \qquad (10.52)$$

$$g_3 = -\frac{\ell}{2\rho_o'} \sec^3\alpha\, Y^2 P_x - \frac{\ell}{2\rho_o'\beta} \tan\alpha \sec^2\alpha\, Y^2 P_T, \qquad (10.53)$$

where α is the entry. A similar set of polynomials holds for the exit face.

e. The General Bending Magnet

A general bending magnet is a dipole magnet that has arbitrary entrance (ψ) and exit (ϕ) angles, and an arbitrary bending angle (2α). See figure 10.5. It includes, as special cases, the normal-entry-and-exit bending magnet ($\psi = \phi = 0$) and the symmetric parallel-face magnet ($\psi = \phi = \alpha$).

With the map for the normal entry and exit magnet and for the symmetric parallel face magnet already available (Douglas [1982]), the general bending magnet is most conveniently calculated by imagining it to be three separate magnets in succession (Fig. 10.5). The center magnet is a normal-entry-and-exit magnet, and the outer magnets are HPF magnets. Finally, the leading and trailing midplane (or pole face) rotations, as for the parallel-face magnet, go at the beginning and end, with any fringe field maps between them and the body. Once all the maps are obtained, they may be concatenated numerically or analytically to obtain the map of the general bending magnet.

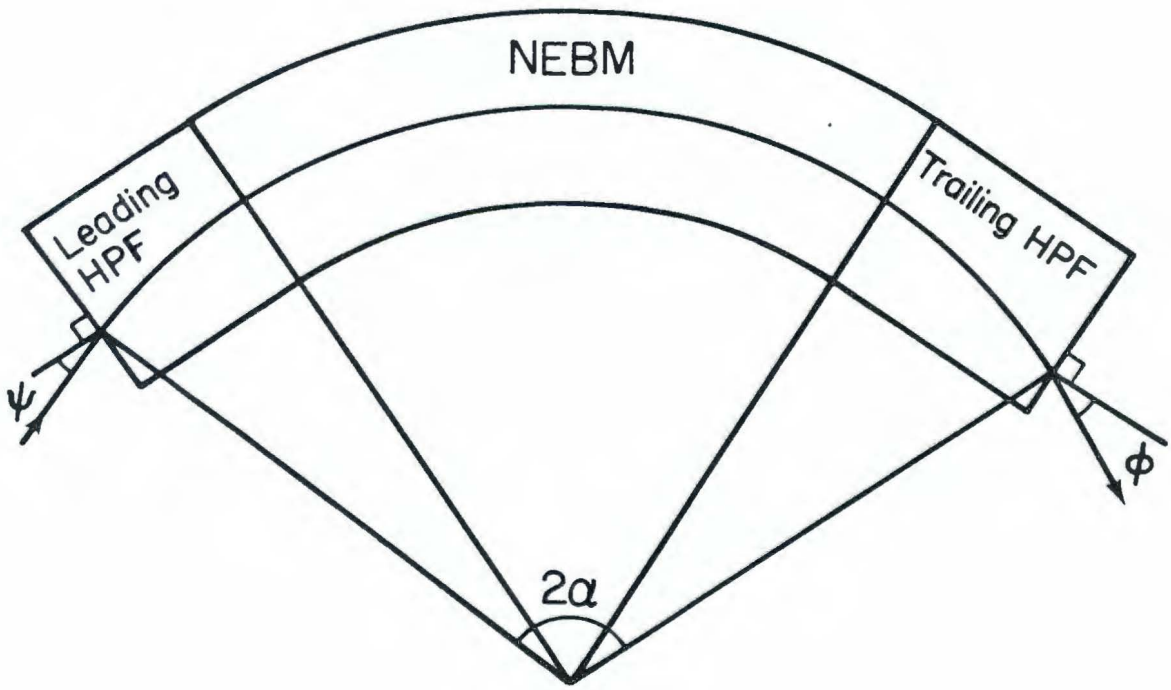


Figure 10.5 The General Bending Magnet

11. Description of Alignment Errors

Having treated in the last two chapters one major source of accelerator errors, the mispowering of dipoles, we now turn our attention to another major source, alignment and position errors of beamline elements. One expects that such errors produce first-order Lie transformations; for example, a quadrupole translated horizontally will exert a dipole force on the design particle, thus bending it off the design trajectory. This chapter and the next are devoted to handling these kinds of problems; this chapter gives a procedure for the complete description of an arbitrary misalignment, and the next shows how to turn that description into a map for a misaligned element.

Before we can hope to compute the effects of alignment errors in accelerators, we need to be able to describe them. Since we wish to allow for all possible misalignments, we will want to describe the misalignment as rigid body motion of some fiducial point on the element from the ideal position and orientation to the actual. We already know that such motions form a group, the Euclidean group, which may be parameterized by six numbers, three translations and three rotations.

The first temptation might be to proceed to calculate the polynomials (and matrix) of the transfer map with the six misalignment parameters, in addition to the parameters of a perfect element, as shown in Chapter 8. While this is certainly possible, it does not make use of the transfer map for a perfect element that is already calculated, and requires the recomputation of all maps.

We would like to be able to decouple the misalignment from the per-

fect element map. It is clear that this is possible when one notes that a misaligned element has the same transfer map from its entry face to its exit face as a perfect one, the element itself not having changed. We merely have to sandwich this map between two maps which transform the coordinates at either face from the ideal to the actual or vice versa (see Figure 11.1).

$$M_{actual} = M_{\ell}\ M_{ideal}\ M_{t}. \tag{11.1}$$

If we are able to calculate these coordinate-transformation, or matching, maps M_{ℓ}, M_{t}, we may use the existing ideal maps, and with the concatenation techniques shown in Chapter 4, obtain the map for the misaligned element.

The first step is to note that a coordinate shift map is a realization of the Euclidean group. Each misalignment coordinate change is described by an element of the Euclidean group. Therefore, our task is two-fold: first, finding the Euclidean group element giving the coordinate change at each face, given the misalignment at the fiducial point, and second, generating the map M_{ℓ} or M_{t} given the Euclidean group element. The first task will be dealt with in this chapter and the second in Chapter 12.

We seek the function that gives the misalignment at each of the pole faces given the misalignment at the fiducial point of the magnet:

$$F : E \to E, \tag{11.2}$$

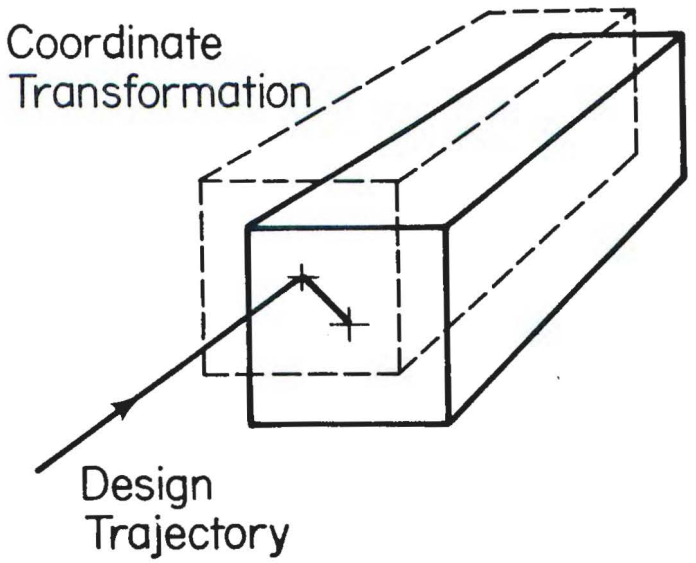


Figure 11.1 Geometry of a Misaligned Element

where E is the Euclidean group parameterized by (ΔX, ΔT, ΔZ, ϕ, θ, ψ) (see Chapter 7). We may compute this knowing the rules for group multiplication and inversion and using a little physical insight. Let A ε E be the rigid body motion from the coordinate plane intersecting the fiducial point to the entry or exit face (Fig. 11.2). Let B ε E be the motion from the ideal location of this plane to the actual location, and let C ε E be the motion from the ideal location of the pole face in question to its actual location. We wish to find C, given A, a property of the magnet, and B, the quantity specified in giving the magnet's misalignment. These quantities have the relation $A = C^{-1}AB$. Thus

$$C = ABA^{-1} \qquad (11.3)$$

The procedure for multiplying and inverting Euclidean group elements was given in Chapter 7, so using these rules and the above relation, we will be able to find the group element at each face.

Since B is specified in giving the misalignment, we need only compute A, given properties of the element, in order to obtain C. The computation of A is straightforward given a few simple geometric properties of the element. If the element is straight-line (no design dipole field), the Euclidean group element that gives the transformation from the fiducial point to the entry or exit face is a simple translation along the z axis of plus or minus half one length of the element, with no rotation (see Figure 11.3a).

If the element is curved, i.e. has a dipole field, the situation is only slightly more complicated (see Figure 11.3b). First, the coordinate phase must be rotated to be parallel to the face. This is a

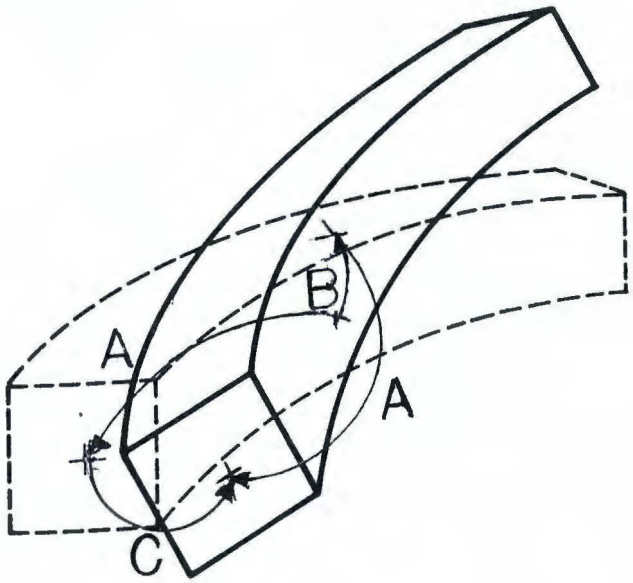
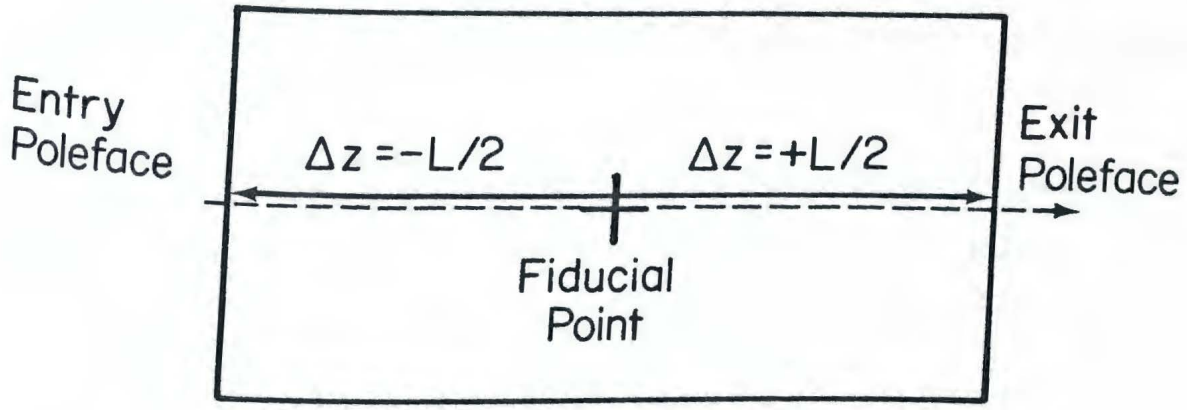
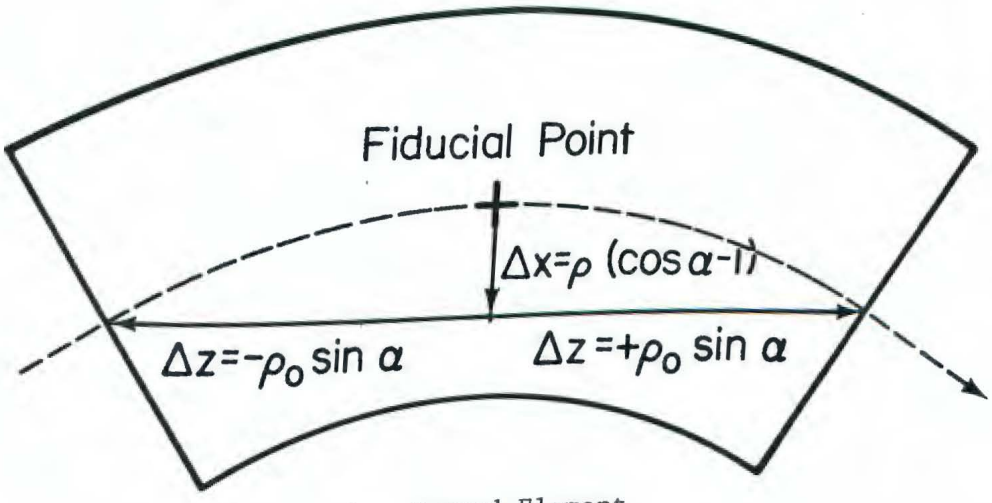


Figure 11.2 Euclidean Group Elements in a Misalignment



a. Straight-Line Element



b. Curved Element

Figure 11.3 Computation of the Euclidean Group Element Transforming Coordinates from the Fiducial Point to a Pole Face

midplane rotation, so the Euler angle θ will be equal to $\pm\alpha$, one-half the bending angle. Then there must be a translation to the face of $\mp\rho_o \sin\alpha$ in the Z direction, where ρ_o is the design bending radius, and by $\rho_o (\cos\alpha - 1)$ in the X direction.

Now that we have the Euclidean group element giving the coordinate transformation at the entry and exit face, we wish to be able to calculate these symplectic maps M_ℓ and M_t. This is the subject of the next chapter.

12. Realization of the Euclidean Group by Symplectic Maps

We now have the description of the coordinate change at the entry and exit face in terms of the Euclidean group. What we need, in order to concatenate and find the map for a misaligned element, is the symplectic transformation that this produces, as given by the polynomials of the factored Lie transformation. These transformations are a realization of the Euclidean group; that is, the group composition of the Euclidean group is reflected in the concatenation of the symplectic maps. As we shall see, these symplectic maps are non-linear in general.

The realization of the Euclidean group can be decomposed into six separate maps, corresponding to each of the parameters. The rotations come first, in the order given in Chapter 7, and then the translations, in any order, because they commute. We may write

$$M = M_{R_Z(\phi)}\ M_{R_Y(\theta)}\ M_{R_Z(\phi)}\ M_{T_X(\Delta_X)}\ M_{T_Y(\Delta_Y)}\ M_{T_Z(\Delta_Z)}. \tag{12.1}$$

In the parameterization of the Euclidean group used here, the rotations are performed first and the translations second. The rotations of the axes are performed actively, relative to the original axes which are fixed (Chapter 7). The translations of the axes are then performed, described in terms of the new (rotated) axes. Figure 12.1 illustrates the element $R_Z(\frac{\pi}{2})\ T_X(\Delta X)$. From the particle's point of view, the transformation is passive.

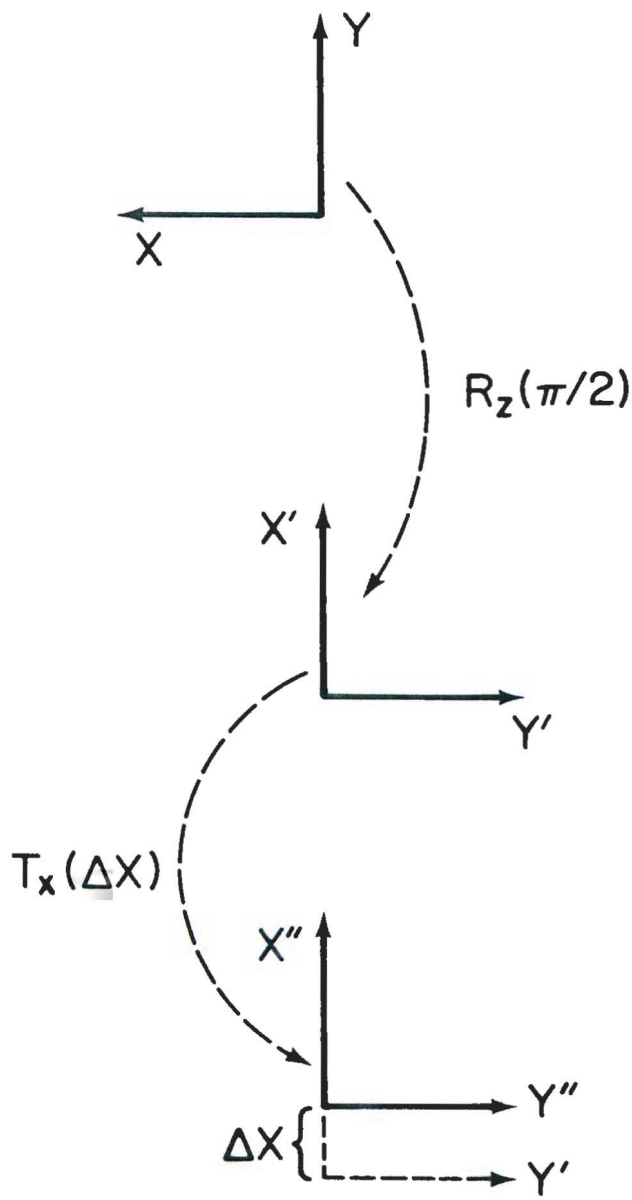


Figure 12.1 The coordinate transformation $R_Z(\frac{\pi}{2})\ T_X(\Delta X)$

a. Translations

Although they occur last, let us treat the translations first. T_X and T_Y, translations in the X and Y directions, may be taken together, because their computation is identical, save for exchanging Y with X. T_Z requires a different treatment because Z is the independent variable.

To translate in the X direction by Δ_X we need to send all values of the X coordinate from X to X - Δ_X; active motion of the element means passive motion of the coordinates since the set of coordinates used in the symplectic map on the particles is considered to be attached to the element. This coordinate shift is effected by $f_1 = \Delta_X P_X$,

$$e^{:f_1:} = e^{\Delta_X:P_X:}, \tag{12.2}$$

so that after applying this transformation,

$$\bar{X} = e^{\Delta_X:P_X:} X = X - \Delta_X, \tag{12.3a}$$

$$\bar{P}_X = P_X, \tag{12.3b}$$

etc.

This is exactly the effect desired. Clearly, replacing X with Y and P_X with P_Y has the same effect on Y.

Translation in the Z direction is slightly more complicated, owing to the fact that Z is the independent variable and not a part of phase space. We have made a canonical transformation so that T, P_T are a

canonical conjugate pair in phase space, Z is the independent variable and $-P_Z$ is the Hamiltonian function. Assuming that in the process of misaligning in the Z direction only free space is covered or exposed (see Fig. 12.2), the map giving the Z coordinate change is a drift of positive or negative length. This is an ordinary drift except that the flight time for a design particle is zero, so we must introduce a translation in the T coordinate, similar to the translations in X and Y above, of the actual flight time in the drift. This amount Δ_T is $\ell/c\beta$, where $c\beta$ is the design velocity.

One may find the Hamiltonian for the drift as Douglas [1982] did. Start with the Hamiltonian in ordinary phase space,

$$H = \sqrt{p^2c^2 + m^2c^4} \tag{12.4}$$

or

$$p^2 = p_x^2 + p_y^2 + p_z^2 = \frac{H^2}{c^2} - m^2c^2. \tag{12.5}$$

In making the canonical transformation to new phase space coordintes x, p_x, y, p_y, t, p_t from the old x, p_x, y, p_y, z, p_z, the new Hamiltonian κ is what was $-p_z$, and the old Hamiltonian H becomes $-p_t$,

$$\kappa(x,p_x,y,p_y,t,p_t) = -p_z = -\sqrt{\frac{p_t^2}{c^2} - m^2c^2 - p_x^2 - p_y^2}\ . \tag{12.6}$$

Scaling $K = \frac{1}{p_o}\kappa^{NEW}$ and the momenta as shown in Chapter 1, with $p_o = \gamma\beta mc$, we have

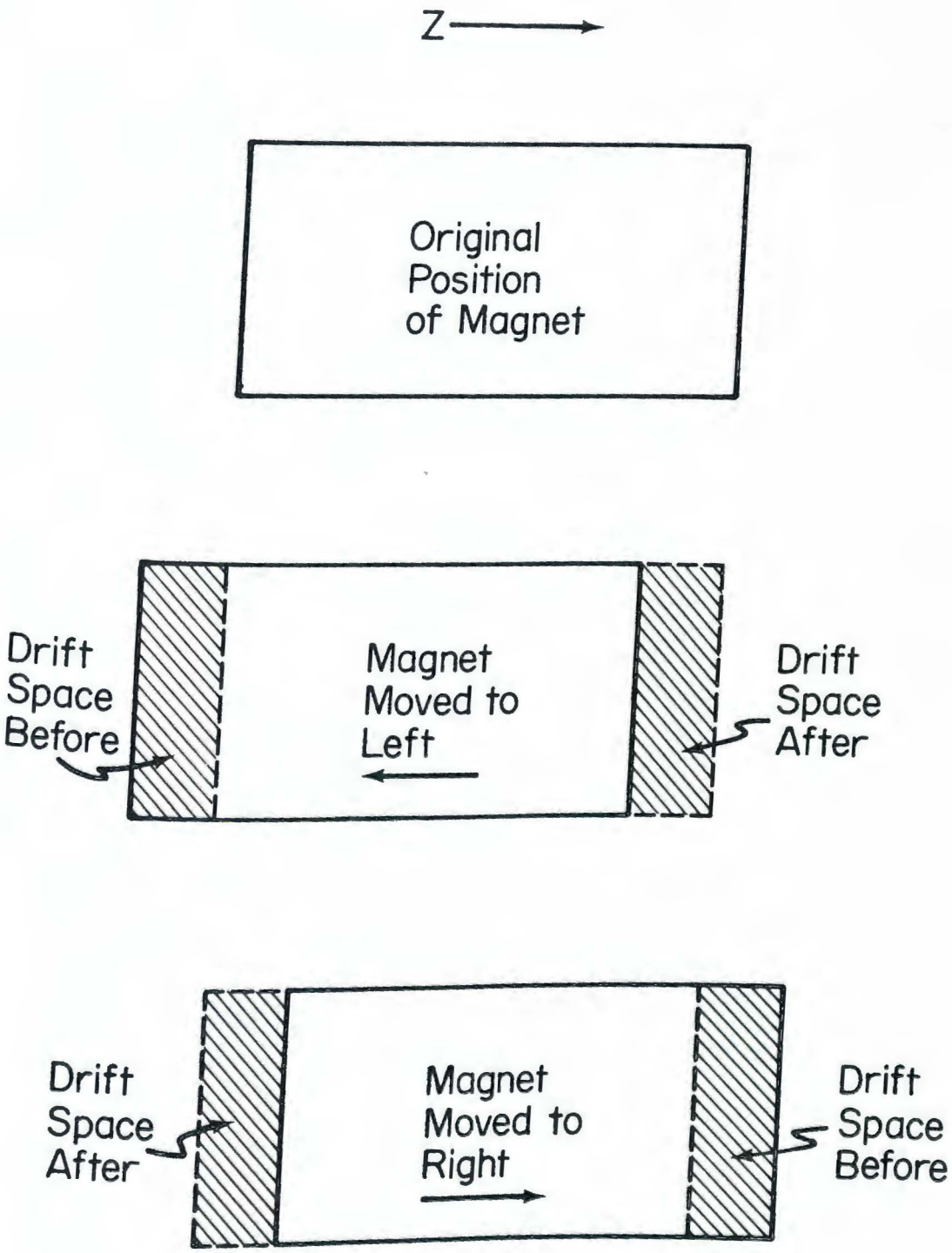


Figure 12.2 Translation in the Z Direction

$$K = -\sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - P_X^2 - P_Y^2}\ , \tag{12.7}$$

dropping the irrelevant constant term. In preparation for calculating the polynomials of the map, we expand K order by order in the phase space variables. We find

$$K = K_0 + K_1 + K_2 + K_3 + K_4 \ \ldots \tag{12.8}$$

where

$$K_1 = \frac{1}{\beta} P_T \tag{12.9a}$$

$$K_2 = \frac{1}{2} (P_X^2 + P_Y^2) + \frac{1}{2\beta^2\gamma^2} P_T^2 \tag{12.9b}$$

$$K_3 + \frac{1}{2\beta} P_T(P_X^2 + P_Y^2) + \frac{1}{2\beta^3\gamma^2} P_T^3 \tag{12.9c}$$

$$K_4 = \frac{1}{8} (P_X^2 + P_Y^2)^2 - \frac{1}{4} (1- \frac{3}{\beta^2}) P_T^2 (P_X^2 + P_Y^2) - \frac{1}{8\beta^2\gamma^2} (1 - \frac{5}{\beta^2}) P_T^4. \tag{12.9d}$$

Computation of the polynomials is easy because all terms of :K: commute with themselves at different times, so

$$f_1 = \frac{\ell}{\beta} P_T, \tag{12.10a}$$

$$f_2 = \ell (\frac{1}{2} (P_X^2 + P_Y^2) + \frac{1}{2\gamma^2\beta^2} P_T^2)\ , \tag{12.10b}$$

$$f_3 = -\ell (\frac{1}{2\beta} P_T (P_X^2 + P_Y^2) + \frac{1}{2\gamma^2\beta^3} P_T^3)\ , \tag{12.10c}$$

$$f_4 = -\ell \left(\frac{1}{8\beta^2\gamma^2}\left(1 - \frac{5}{\beta^2}\right) P_T^4 + \frac{1}{8}\left(P_X^2 + P_Y^2\right)^2 - \frac{1}{4}\left(1 - \frac{3}{\beta^2}\right) P_T^2 \left(P_X^2 + P_Y^2\right)\right) . \tag{12.10d}$$

$e^{:f_2:}$ has the corresponding matrix

$$\begin{bmatrix} 1 & \ell & & & & \\ 0 & 1 & & & & \\ & & 1 & \ell & & \\ & & 0 & 1 & & \\ & & & & 1 & \frac{1}{2\gamma^2\beta^2} \\ & & & & 0 & 1 \end{bmatrix} . \tag{12.11}$$

Although it is not strictly speaking a misalignment, it is appropriate at this point to consider how to treat a phasing error in an RF cavity. An RF cavity is an active accelerating/decelerating section with a longitudinal electric field

$$\vec{E} = \hat{z}\, E_o \sin(\omega t). \tag{12.12}$$

One possible cavity to consider is that of a short buncher. The effective Hamiltonian for a short buncher is (see Douglas [1982], Table 4.9)

$$\kappa = -\sqrt{1 - \frac{2}{\beta} P_T + P_T^2 - \left(P_X^2 + P_Y^2\right)} - \frac{1}{\beta} P_T + \frac{qE_o}{p_o \omega} \cos\left(\frac{\omega T \ell}{c}\right), \tag{12.13}$$

which may be expanded and factored as Douglas does to obtain the map.

Now suppose that there is a phasing error in the cavity, so that the electric field is

$$\vec{E} = \hat{z}\, E_o \sin(\omega t + \alpha). \tag{12.14}$$

One may modify the Hamiltonian (12.13) appropriately, and do an expansion and integration. This is analogous to what was first proposed for misalignments: include the misalignments in the Hamiltonian and solve for the map. As with the misalignment case, however, we may separate out the phasing error so that we may take advantage of the existing perfect-element map already computed. Specifically, we precede and follow the map of the perfectly phased cavity with time translation map: it shifts the time $T \to T + \frac{c}{\ell\omega}\alpha$. This is accomplished with a first-order transformation proportional to P_T,

$$M_{\text{real cavity}} = e^{-\frac{c\alpha}{\ell\omega}:P_T:} M_{\text{perfect cavity}} e^{\frac{c\alpha}{\ell\omega}:P_T:} \qquad (12.15)$$

This is just a translation in time in the same way that a horizontal misalignment perpendicular to the beam is a translation in X.

b. Rotations

Although there are three coordinate-rotation maps, there are only two that are different, R_Y and R_Z. Because of the definition of the Euler angles, R_X does not occur and need not be computed. For the moment, however, we shall concern ourselves with all three rotations and how they relate to each other.

In order to have a realization of SO(3) in the group of symplectic maps, there must be a realization of the Lie algebra of SO(3) in the Poisson bracket Lie algebra; that is, there must be a Lie algebra homomorphism from the Lie algebra of SO(3) to the Poisson bracket Lie algebra. The Lie algebra, or the generators, of SO(3) are what can be called the angular momenta L_X, L_Y, L_Z. They are related by the

structure equations,

$$[L_i, L_j] = L_k \quad , \quad i,j,k = \text{cyclic combinations of } X,Y,Z. \tag{12.16}$$

If the phase space were X, P_X, Y, P_Y, Z, P_Z, all the rotations of SO(3) would be linear and, thus, the realization a representation. This may be represented in the Poisson bracket Lie algebra with

$$L_X = YP_Z - ZP_Y \tag{12.17a}$$

$$L_Y = ZP_X - XP_Z \tag{12.17b}$$

$$L_Z = XP_Y - YP_X. \tag{12.17c}$$

In the matrix Lie algebra (see Chapter 1, Section f), we have

$$L_Z = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{12.18}$$

Thus the element of the symplectic group which represents a passive rotation (rotation of the element clockwise by the right-hand side for $\phi > 0$) about the Z axis by an angle ϕ is

$$M_{R_Z(\phi)} = e^{\phi:L_Z:} \tag{12.19}$$

or as a matrix

$$\begin{bmatrix} \cos\phi & 0 & \sin\phi & 0 & 0 & 0 \\ 0 & \cos\phi & 0 & \sin\phi & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 & 0 & 0 \\ 0 & -\sin\phi & 0 & \cos\phi & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{12.20}$$

Because T and P_T are part of phase space and not Z and P_Z, however, we may not use all of the above representations. In particular, rotation about the Z axis, R_Z, is still linear and may be treated as above, but the midplane rotation, R_Y, is not as simple. Since this transformation changes the value of the independent variable (Z) for some particles, we must specify what kind of Hamiltonian is acting. The rotation is assumed to occur at the entry and exit faces of the magnet and any fringe fields are hard-edge (zero length), we may assume no field, and therefore a drift Hamiltonian. In addition, since our goal is a realization of the rotation group SO(3) in the group of symplectic maps (actually $Q^{(2)}$, see Chapter 3), the generators L_X, L_Y, L_Z must obey the structure equations (12.16) as they are realized in the Poisson bracket Lie algebra.

We shall approach this by creating L_X and L_Y so that the dynamics are the same as a drift, then verifying that they, along with $L_Z = XP_Y - YP_X$, satisfy the structure equations (12.16). Let $H(\zeta)$ be the Hamiltonian of a drift, calculated above, and suppose, to start, that we are considering L_Y, an infinitesimal rotation about the Y axis.

Consider the propagation of a particle under the Hamiltonian that is near in phase space to $X = \ell$. Then, because the independent variable Z changes by $\Delta Z = -\ell\theta$ (see Fig. 12.3), θ small, we have the final phase space coordinates as a function of the initial (to first order in ΔZ)

$$\bar{\zeta} = \zeta + \Delta Z \frac{d\zeta}{dz} \tag{12.21}$$

$$= \zeta + \Delta Z \ [\zeta, K]$$

$$= \zeta - \ell\theta \ [\zeta, K].$$

When the transformation is applied to ζ, we obtain the alternate result

$$\bar{\zeta} = e^{:\theta L_y:} \ \zeta \approx \zeta + \theta :L_y:\zeta \tag{12.22}$$

These two calculations should give the same result for $\bar{X}$, $\bar{Y}$, $\bar{P}_Y$, etc. (but not $\bar{P}_X$) to first order in θ. Thus, for example, we compute $\bar{X}$ for some small initial conditions v_o around $X = \ell$ ($v_{oX} = 0$)

$$(X + :\theta L_y:X) \Big|_{\zeta = \ell\hat{X} + v_o} = (X - \ell\theta \ [X,K] \Big|_{\zeta = \ell\hat{X} + v_o} \tag{12.23}$$

$$:L_Y: \ X \Big|_{\zeta = \ell\hat{X} + v_o} = -\ell \ [X,K] \Big|_{\zeta = \ell\hat{X} + v_o} \tag{12.24}$$

$$\frac{\partial L_Y}{\partial P_X} = X \frac{\partial K}{\partial P_X} \tag{12.25}$$

$$L_Y = XK + c_{p_x} \tag{12.26}$$

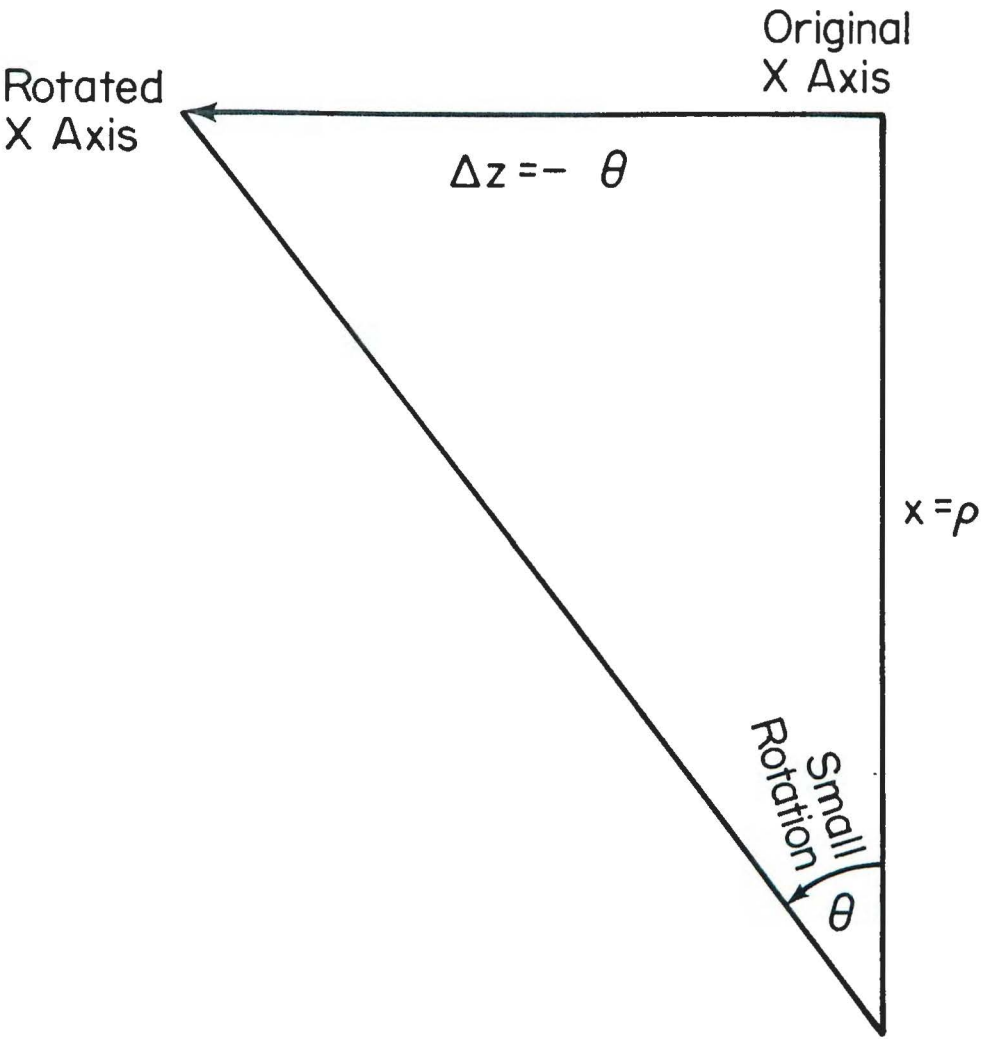


Figure 12.3 Drift of a Particle to Rotated Coordinates

where c_{P_X} is a constant independent of P_x. By a similar computation, we may conclude

$$L_Y = XK + w_X \tag{12.27}$$

where w_X is dependent only on X.

Similarly, we may compute L_X

$$L_X = -YK + w_Y \tag{12.28}$$

and, as before, $L_Z = XP_Y - YP_X$. The next step will show that $w_X = w_Y = 0$.

To verify that these realizations of L_X, L_Y, and L_Z satisfy the structure equations, we need to substitute the Hamiltonian for a drift κ (12.7). Then the generators of drift rotations for this phase space are

$$L_X = Y\sqrt{1 - \frac{2}{\beta}P_T + P_T^2 - P_X^2 - P_Y^2} \tag{12.29a}$$

$$L_Y = -X\sqrt{1 - \frac{2}{\beta}P_T + P_T^2 - P_X^2 - P_Y^2} \tag{12.29b}$$

$$L_Z = XP_Y - YP_X. \tag{12.29c}$$

In computing the Poisson brackets, note that

$$\frac{\partial K}{\partial P_X} = -\frac{P_X}{K} \quad , \quad \frac{\partial K}{\partial P_Y} = -\frac{P_Y}{K} \, . \tag{12.30}$$

The Poisson Bracket $[L_X, L_Y]$ is then seen to be L_Z,

$$[L_X, L_Y] = [YK, -XK] \tag{12.31}$$

$$= (K + X\frac{\partial K}{\partial X})(-Y\frac{\partial K}{\partial P_X}) - (X\frac{\partial K}{\partial P_X})(-Y\frac{\partial K}{\partial X})$$

$$+ (X\frac{\partial K}{\partial Y})(-Y\frac{\partial K}{\partial P_Y}) - (X\frac{\partial K}{\partial P_Y})(-K-Y\frac{\partial K}{\partial Y})$$

$$+ (X\frac{\partial K}{\partial T})(-Y\frac{\partial K}{\partial P_T}) - (X\frac{\partial K}{\partial P_T})(-Y\frac{\partial K}{\partial T})$$

$$= -KY\frac{\partial K}{\partial P_X} - XY\frac{\partial K}{\partial X}\frac{\partial K}{\partial P_X} + XY\frac{\partial K}{\partial P_X}\frac{\partial K}{\partial X}$$

$$- XY\frac{\partial K}{\partial Y}\frac{\partial K}{\partial P_Y} + XK\frac{\partial K}{\partial P_Y} + XY\frac{\partial K}{\partial P_Y}\frac{\partial K}{\partial Y}$$

$$- XY\frac{\partial K}{\partial T}\frac{\partial K}{\partial P_T} + XY\frac{\partial K}{\partial P_T}\frac{\partial K}{\partial T}$$

$$= -K(X\frac{\partial K}{\partial P_Y} - Y\frac{\partial K}{\partial P_X}) = XP_Y - YP_X = L_Z$$

as desired. The Poisson Bracket $[L_Z, L_X]$, may be computed to verify that it indeed is L_Y,

$$[L_Z, L_X] = [XP_Y - YP_X, YK] = P_Y Y\frac{\partial K}{\partial P_X} - (-Y) Y\frac{\partial K}{\partial X} \tag{12.32}$$

$$- P_X Y\frac{\partial K}{\partial P_Y} - X(K + Y\frac{\partial K}{\partial Y})$$

$$= -X K + Y(P_Y\frac{\partial K}{\partial P_X} - P_X\frac{\partial K}{\partial P_Y}) + Y(Y\frac{\partial K}{\partial X} - X\frac{\partial K}{\partial Y})$$

$$= -XK = L_Y$$

The reader may easily verify that $[L_Y, L_Z] = L_X$, too.

Although we shall not need to here, it should be noted that this process can be carried over to an arbitrary Hamiltonian. This possibility is discussed in part c.

We may now rotate to our heart's content. In particular, we are interested in finding the map for rotations about the Y axis, M_{R_Y},

$$M_{R_Y}(\phi) = e^{\phi:L_Y:} \tag{12.33}$$

$$= e^{-\phi:X\sqrt{1 - \frac{2}{\beta}P_T + P_T^2 - P_X^2 - P_Y^2}:}$$

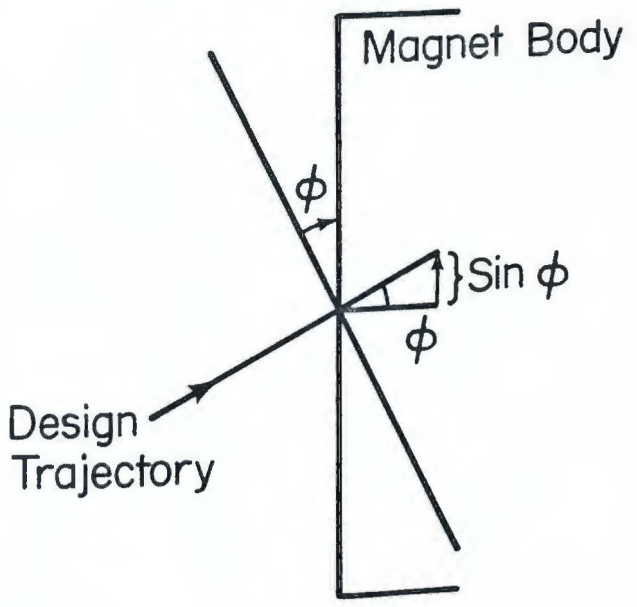
and factorizing it so that it may be applied to the phase space coordinates to see how they transform. We shall do this for arbitrary ϕ to compare with Douglas's [1982] result, but later we shall use the process described in Chapter 8 to factorize the map for infinitesimal ϕ, as is suitable for misalignments. The results will then be compared.

To calculate the factored map for arbitrary ϕ, we may consider the effect of the unfactored map (12.33) on phase space, and then extract the factored expression of the map after expanding in the phase space variables. For example, one may compute $\bar{P}_X$:

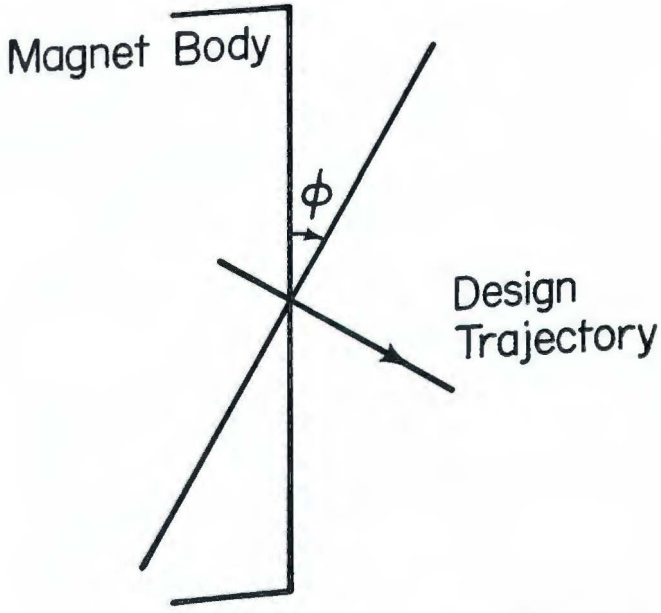
$$\bar{P}_X = e^{-\phi:XK:}\, P_x = P_X - \phi K - \frac{1}{2}\phi^2 P_X + \frac{1}{6}\phi^3 K \ldots \tag{12.34}$$

$$= \cos\phi\, P_X - \sin\phi\, K,$$

because :XK: forms a cycle of two acting on P_X,



a. Leading Midplane Rotation



b. Trailing Midplane Rotations

Figure 12.4 Leading and Trailing Midplane Rotations for a Parallel-Face Magnet

where the combined effect of $e^{:f_2^c:}\ e^{:f_2^a:}$ is the matrix

$$M = \begin{bmatrix} \cos\phi & 0 & 0 & 0 & 0 & -\frac{1}{\beta}\sin\phi \\ 0 & \sec\phi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{\beta}\sin\phi & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (12.38)$$

and the polynomials are

$$f_1 = \sin\phi\, X \qquad (12.39a)$$

$$f_3 = -\frac{1}{2}\tan\phi\, X(P_X^2 + P_Y^2 + \frac{1}{\gamma^2\beta^2}P_T^2) \qquad (12.39b)$$

$$f_4 = \frac{1}{2}\tan\phi\, X\,(\frac{1}{2}\tan\phi\, P_X - \frac{1}{\beta}P_T)\,(P_X^2 + P_Y^2 + \frac{1}{\gamma^2\beta^2}P_T^2). \qquad (12.39c)$$

Note that this is accurate to all orders in ϕ, even though we do not need them all.

Alternatively, we may calculate the factored map for small ϕ using the techniques of Chapter 8. This will provide another instructive example in those techniques, as well as allowing us to compare these results with those exact in ϕ as derived above.

We start with the full Hamiltonian and then expand it

$$-H = -XK = X\sqrt{1 - \frac{2}{\beta}P_T + P_T^2 - P_X^2 - P_Y^2} \qquad (12.40)$$

$$= X[1 - \frac{1}{\beta}P_T - \frac{1}{2\beta^2\gamma^2}P_T^2 - \frac{1}{2}(P_X^2 + P_Y^2)$$

$$- \frac{1}{2\beta^3\gamma^2}P_T^3 - \frac{1}{2\beta}P_T(P_X^2 + P_Y^2) + \ldots]$$

so that, with ϕ the small quantity ε of Chapter 8,

$$- H_1^{(1)} = X \tag{12.41a}$$

$$- H_2^{(1)} = - \frac{1}{\beta} XP_T. \tag{12.41b}$$

Thus

$$:g_2^{(1)}(\phi): = \int_o^\phi : - \frac{1}{\beta} XP_T : d\phi' = - \frac{\phi}{\beta} :XP_T:, \tag{12.42}$$

and the matrix corresponding to $e^{:g_2^{(1)}(\phi)}$ is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -\frac{\phi}{\beta} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{\phi}{\beta} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \tag{12.43}$$

Then we may substitute this linear transformation to get H_n^{int},

$$- H_3^{(1)^{int}} = - \frac{1}{2\beta^2\gamma^2} XP_T^2 - \frac{1}{2} X((P_X - \frac{\phi}{\beta} P_T)^2 + P_Y^2) \tag{12.44a}$$

$$- H_4^{(1)^{int}} = - \frac{1}{2\beta^3\gamma^2} XP_T^3 - \frac{1}{2\beta} XP_T((P_X - \frac{\phi}{\beta} P_T)^2 + P_Y^2) \tag{12.44b}$$

Using (8.33), we find

$$g_3^{(1)'}(\phi) = -\int_o^\phi H_3^{(1)\,\mathrm{int}}\, d\phi' \qquad (12.45)$$

$$= -\frac{1}{2\beta^2}\left(\frac{\phi}{\gamma^2} + \frac{\phi^3}{3}\right) XP_T^2 - \frac{\phi}{2} XP_X^2 + \frac{\phi^2}{2\beta} XP_X P_T - \frac{\phi}{2} XP_Y^2 \,.$$

Since third-order polynomials retain only one order of the small quantity ϕ, we may shorten this slightly to find

$$g_3^{(1)'}(\phi) = -\frac{\phi}{2\beta^2\gamma^2} XP_T^2 - \frac{\phi}{2} XP_X^2 - \frac{\phi}{2} XP_Y^2 \,. \qquad (12.46)$$

Further, transforming by $e^{:g_2^{(1)}(\emptyset):}$ has no effect, because the order in ϕ will be too high, so $g_3^{(1)} = g_3^{(1)'}$. We may calculate $g_4^{(1)}$ by the formula (8.44), but since the integral over the independent variable ϕ' is only over the range 0 to the small quantity ϕ, and $H_4^{(1)\,\mathrm{int}}$ is independent of ϕ, we know in advance all these terms will be proportional to ϕ' with $n \geq 1$. This is beyond what we need to retain, so we may take $g_4 = 0$.

This ends the first pass. We now may compute the second "Hamiltonian" $H^{(2)}$

$$H^{(2)} = e^{:g_2^{(1)}(\phi'):}\, e^{:g_3(\phi'):} X. \qquad (12.47)$$

Because $e^{:g_2^{(1)}(\phi'):}$ has no effect, $H^{(2)}$ has the expansion

$$H^{(2)} = X + \phi' XP_X + \cdots \qquad (12.48)$$

$H^{(2)}$ splits up by order into

$$H_1^{(2)} = X, \tag{12.49a}$$

$$H_2^{(2)} = \phi' X P_X. \tag{12.49b}$$

For $n \geq 3$, the $H_n^{(2)}$ are zero because they will be too high order in ϕ; note that since a small quantity ϕ is the independent variable, we will get (at least) one extra power of ϕ on integration; therefore, terms of total order 4 and higher will be excluded.

Now we solve for $g_2^{(2)}$ with the "Hamiltonian" $H^{(2)}$, using (8.21),

$$g_2^{(2)} = \int_o^\phi \phi' X P_X \, d\phi' = \frac{\phi}{2} X P_X.$$

The map $e^{:g_2^{(2)}:}$ may be represented as the matrix

$$\begin{bmatrix} 1 - \frac{\phi^2}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 + \frac{\phi^2}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{12.50}$$

to the appropriate order in ϕ.

The higher-order dynamics equations yield $g_n^{(2)} = 0$, $n > 2$, because $H_n^{(2)} = 0$. Thus we may go right to the last step of the iteration, and integrate the first-order term. Step by step in the iteration, it is

$$H_1^{(2)} = H_1^{(1)} = X, \tag{12.51}$$

$$H_1^{(3)}(\phi) = (1 - \frac{\phi^2}{2})\, X. \tag{12.52}$$

The integral is

$$g_1 = \int_o^\phi (1 - \frac{\phi'^2}{2})\, X\, d\phi' = (\phi - \frac{\phi^3}{3})\, X. \tag{12.53}$$

Summarizing the results, the map in factored form is

$$M = e^{:g_1:}\, e^{:g_2^{(2)}:}\, e^{:g_2^{(1)}:}\, e^{:g_3:} \tag{12.54}$$

with the polynomials defined by

$$g_1 = (\phi - \frac{\phi^3}{3})\, X \tag{12.54a}$$

$$g_2^{(2)} = \frac{\phi^2}{2}\, XP_X \tag{12.54b}$$

$$g_2^{(1)} = \frac{\phi}{\beta}\, XP_T \tag{12.54c}$$

$$g_3 = -\frac{\phi}{2}\,(\frac{1}{\beta^2\gamma^2}\, XP_T^2 + X(P_X^2 + P_Y^2)) \tag{12.54d}$$

The matrix corresponding to

$$e^{:g_2^{(2)}:}\, e^{:g_2^{(1)}:} \tag{12.55}$$

is, to the proper order in ϕ,

$$\begin{bmatrix} 1-\frac{\phi^2}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1+\frac{\phi^2}{2} & 0 & 0 & 0 & -\frac{\phi}{\beta} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{\phi}{\beta} & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (12.56)$$

A comparison with the results exact in ϕ, (12.38) and (12.39), reveals this derivation to be identical to the appropriate order in ϕ.

c. Rotations with Propagation Under an Arbitrary Hamiltonian

Although it is off the main line of work, it is interesting to speculate about rotations under an arbitrary Hamiltonian, not necessarily a drift. This would be must useful for the computation of the fringe field rotation for a parallel-face bending magnet, for some finite-length fringe field, i.e., not a hard edge fringe field. Thus it would be necessary to obtain the map for an arbitrary, not necessarily small, rotation.

The general process would be similar to that above for the drift rotation. One would compare the effects of a Hamiltonian infinitesimally with that of the rotation. It is important to note the z evolution would in general be different than the result for a drift (Fig. 12.5). The rotation obtained, L_X,L_Y, together with L_Z, should be checked with the structure equations of the Lie algebra of SO(3) to make sure they are, in fact, a realization. Then the whole result would have to be factored.

It would make an interesting line of investigation for some bright young graduate student (or for some withered old post-doc!).

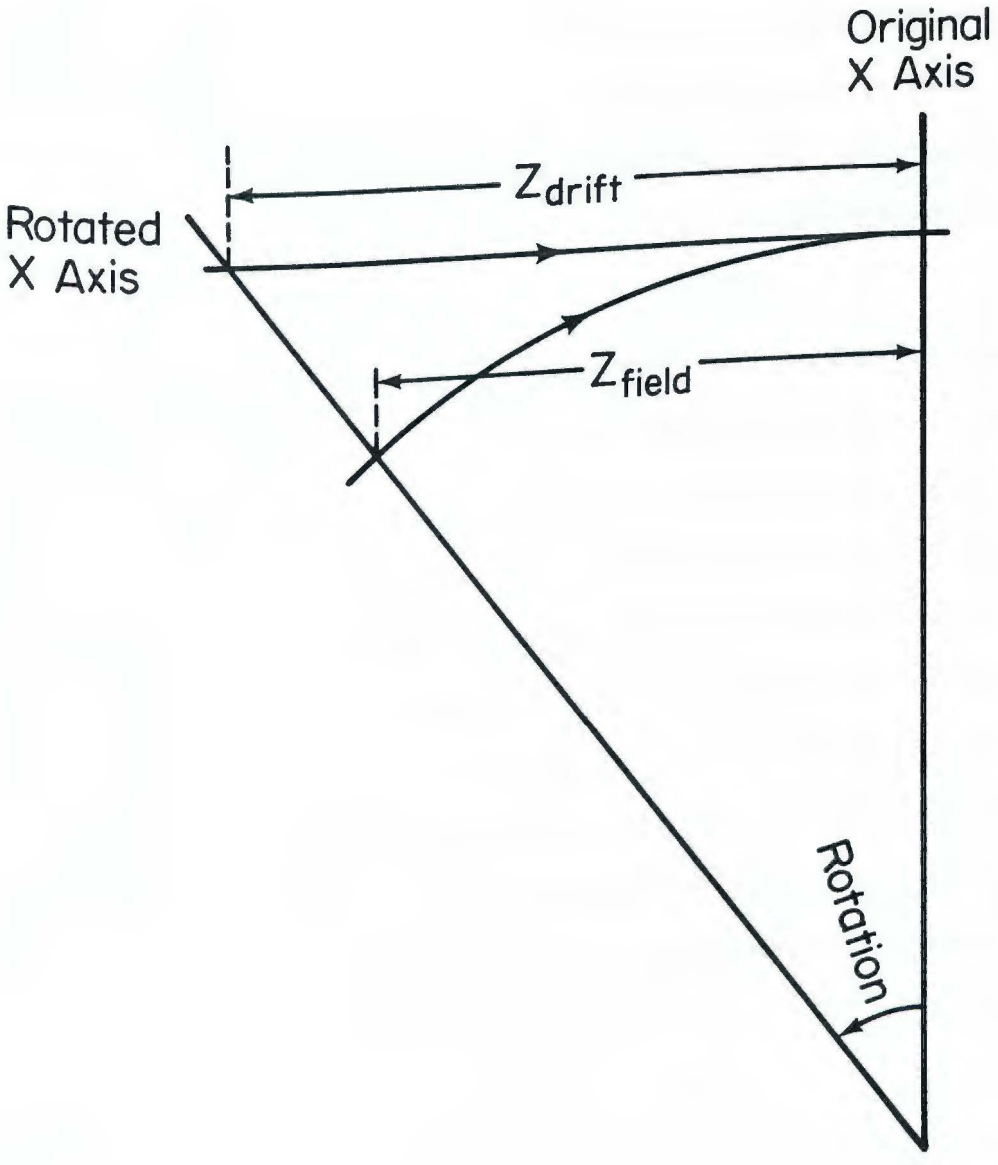


Figure 12.5 Change in z for Drift Rotation and for Rotation with a Non-Zero Field

Appendix A. Treatment of Random Distributions of Errors

The mechanism described in parts I and II shows us how to compute a map for a beamline element that is possibly misaligned and mispowered, and how to concatenate it with other such elements to produce a map for a section of beamline or a whole accelerator. With this map, one may track particles or extract information about the lattice such as tune and chromaticity.

Throughout this procedure, it has been assumed that all parameters, even those representing errors, may be precisely specified. This is certainly reasonable in many circumstances; suppose, for example, that a survey of an accelerator shows that a certain quadrupole is misaligned in the horizontal direction by .5 mm. Then we have the tools to see how this affects the tune, dynamic aperture, and so on. On the other hand, there are cases where the parameters are not known precisely. For example, in a prospective accelerator, one does not know where the elements will actually end up, but we may be able to say from past experience that the quadrupoles will be placed within ± .6 mm of where they should be.

The way it stands, the only way to treat such random distributions of errors is to select samples from such distributions, i.e., as a Monte Carlo technique, and analyze the results for the distribution of desired quantities. Such a calculation could require a large number of computations, particularly if there are many parameters subject to error. Also, one must take care that the source of random numbers is really random, in order that the results are reliable. Even so, the result may be a coincidence, due to the numbers picked.

In this chapter, I describe a way of obtaining the behavior not only at given values of the parameters, but nearby as well. It is based on the method of propagation of errors. One assumes that the parameters are statistical with a certain distribution, and follows the effects of their variation through the creation and concatenation of the maps, and then to the lattice functions or tracking.

There are two assumptions made, both reasonable for practical accelerator design. The first is that the statistical quantities, the uncertain parameters, have a Gaussian distribution with the given mean and variance. This is reasonable in view of the central limit theorem: the sum of statistical quantities tends to a Gaussian, no matter what their distribution. Specifically, errors in accelerator construction and operation are in reality the sums of other errors, e.g., the error of field strength of the magnet is a result of errors in the placement of the wire, the quality of the power supply, etc.

The second assumption is that parameter errors are independent. It is reasonable to assume, for instance, that errors in powering a magnet have nothing to do with errors in its physical length. In the method developed here, it is also assumed that the parameters of two maps are statistically independent. This will not be the case, for instance, in a misalignment, as the leading and trailing shift maps will be functions of the same statistical parameter. Nevertheless, this is adequate for many maps and suffices as a start.

What is given here is a general description of the propagation of errors technique, including nonlinear propagation of errors, and then an application to the mathematical operations of Lie algebraic code such as

MARYLIE: element map generation, concatenation, and computation of lattice functions. The concatenation computed here only includes matrix multiplication and not Poisson bracketing or transforming, and so effects from nonlinearities and errors where first-order transformations are produced are not included. The principles of propagation of error can be extended to these also. Finally, there is a discussion of the results applied to the calculation of the tune of a simple bending cell, including a comparison with results from samples of the distribution.

a. Statistical Distributions and Propagation of Errors

Suppose we have a set of n quantities $\{x_1, x_2, \ldots, x_n\}$, that have a statistical distribution $\phi(x)$, where x is a vector representing the whole set. A convenient way of characterizing the distribution function ϕ is by the mean of x, $\bar{x}$, and the covariance matrix σ^2_{xx}. The means and covariances are defined by integrating over the distribution ϕ,

$$\bar{x}_i = \int x_i \, \phi(x) \, dx \qquad (A.1)$$

and

$$\sigma^2_{x_i x_j} = \int (x_i - \bar{x}_i)(x_j - \bar{x}_j) \, \phi(x) \, dx \qquad (A.2)$$

Note that for the covariance matrix is symmetric. Although it is n by n and therefore has n^2 elements, there are only $\frac{n(n-1)}{2}$ independent off-diagonal terms. Together with the n diagonal terms, there are $\frac{n(n+1)}{2}$ independent terms. Although the diagonal elements are properly called variances, when referring to them as a class I will call them co-

variances.

The technique of propagation of errors allows us to consider the statistical properties of functions of the statistical variables x_1, ..., x_n. Specifically, suppose there are m functions f_1, ..., f_n of these n variables

$$\begin{aligned} & f_1(x_1, \ldots, x_n) \\ & \vdots \\ & f_m(x_1, \ldots, x_n). \end{aligned} \tag{A.3}$$

As with the original statistical variables x, we would like to know about the mean $\bar{f}_i$ and covariance $\sigma^2_{f_i f_j}$ of these functions in terms of the mean and covariance of the x.

To simplify the expression, let us pick any two functions (which may be the same) from the set f_1, ..., f_m, and call then f and g. In f (and g) we may write the Taylor expansion

$$f(x) = \sum_{m=0}^{\infty} \frac{1}{m!} \Big(\sum_{i=1}^{n} \alpha_i \partial_i \Big)^2 f(x) \Big|_{x=\bar{x}} \tag{A.4}$$

where $\alpha_i = x_i - \bar{x}_i$ for all i, and ∂_i acts as $\frac{\partial}{\partial x_i}$ only on f. Explicitly,

$$\begin{aligned} f(x) = f(\bar{x}) + \sum_i \alpha_i \partial_i f + \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \partial_i \partial_j f \\ + \frac{1}{6} \sum_{i,j,k} \alpha_i \alpha_j \alpha_k \partial_i \partial_j \partial_k f \\ + \frac{1}{24} \sum_{i,j,k,\ell} \alpha_i \alpha_j \alpha_k \alpha_\ell \partial_i \partial_j \partial_k \partial_\ell f + \ldots \end{aligned} \tag{A.5}$$

While for most functions the linear term is sufficient in the expansion, we shall consider the general case (non-linear propagation of errors) because we will encounter cases where the extra terms are significant. Integrating with the distribution $\phi(x)$ yields an expansion of the mean in terms of moments,

$$\bar{f} = f(\bar{x}) + \frac{1}{2} \sum_{i,j} \mu_{ij} f_{ij} + \frac{1}{6} \sum_{i,j,k} \mu_{ijk} f_{ijk} + \frac{1}{24} \sum_{i,j,k,\ell} \mu_{ijk\ell} f_{ijk\ell} + \cdots \tag{A.6}$$

where $\mu_{ij\ldots}$ is a moment of the distribution,

$$\mu_{ij\ldots} = \int (x_i - \bar{x}_i)(x_j - \bar{x}_j)\, \phi(x)\, d^n x, \tag{A.7}$$

and $f_{ij\ldots}$ is a multiple derivative of f evaluated at the mean value of x, $\bar{x}_i$

$$f_{ij\ldots} = \partial_i \partial_j \cdots f(x) \Big|_{x=\bar{x}} . \tag{A.8}$$

For completeness of notation, let $f_o \equiv f(\bar{x})$.

Since we assume the random variables have a Gaussian distribution, we shall need these moments; however, let us postpone that task and continue the calculation for an arbitrary distribution.

We may calculate the variance by

$$\sigma^2_{fg} = \overline{(fg)} - \bar{f}\,\bar{g} \tag{A.9}$$

As $\bar{f}$ and $\bar{g}$ are already available with the formula (A.6), we need only calculate $\overline{(fg)}$.

Let $h(x) = f(x)g(x)$. Then $\overline{(fg)} = \bar{h}$, and the formula (A.6) may be used. We need only determine the derivatives of h, $h_{i_1 \ldots i_n}$, in terms of those of f and g. If $C = \{i_1, \ldots, i_n\}$ is the set of indices with which derivatives are to be taken, h_C is the sum of the product of all possible ways of distributing those derivatives between f and g

$$h_C = h_{i_1 \ldots i_n} = \sum_{\substack{\text{divisions of C} \\ \text{into 2 sets A,B}}} f_A g_B . \qquad (A.10)$$

For example,

$$h_{ij} = f_{ij}g_o + f_i g_j + f_j g_i + f_o g_{ij} . \qquad (A.11)$$

We may show the relation (A.10) by induction. More precisely, we may show the relation before evaluation of the derivatives,

$$\partial_C h(x) = \sum_{\substack{\text{division of C} \\ \text{into 2 sets A,B}}} [\partial_A f(x)][\partial_B g(x)], \qquad (A.12)$$

by induction on the size of the set C. Start by assuming that C is the null set. The equation (A.10) then is $h(x) = f(x)\, g(x)$, which is true by the defintion of h. Now assume the formula is true for C of size n-1, and prove it for size n. Specifically, let C' be a set of the n-1 differentiation variables,

$$C' = \{i_1, \ldots, i_{n-1}\}, \qquad (A.13)$$

and let C be C' augmented with a new variable i,

$$C = C' \bigcup \{i\} \tag{A.14}$$

Then the multiple derivative C may be written in terms of the derivatives on C'

$$\partial_C h(x) = \partial_i \partial_{C'} h(x) \tag{A.15}$$

$$= \partial_i \sum_{\substack{\text{divisions} \\ \text{of } C'}} [\partial_A f(x)][\partial_B g(x)]$$

$$= \sum_{\substack{\text{divisions} \\ \text{of } C'}} [\partial_i \partial_A f(x)][\partial_B g(x)] + [\partial_A f(x)][\partial_i \partial_B g(x)]$$

$$= \sum_{\substack{\text{divisions} \\ \text{of } C}} [\partial_A f(x)][\partial_B g(x)]$$

because the possible divisions of C are just all those of C' with i in one or the other. When evaluated at $\bar{x}$, the relation (A.10) is obtained.

Now it is possible to write out the first few terms of the covariance using the relation

$$\sigma^2_{fg} = \overline{(fg)} - \bar{f}\,\bar{g} \tag{A.16}$$

$$= f_o g_o + \frac{1}{2} \sum \mu_{ij} (f_o g_{ij} + f_i g_j + f_j g_i + f_{ij} g_o) + \ldots$$

$$- (f_o + \frac{1}{2} \sum \mu_{ij} f_{ij} + \ldots)(g_o + \frac{1}{2} \sum \mu_{ij} g_{ij} + \ldots)$$

$$= \sum \mu_{ij} f_i g_j + \frac{1}{2} \sum \mu_{ijk} (f_i g_{jk} + f_{jk} g_i)$$

$$+ \sum \left[\frac{1}{6}\,\mu_{ijk\ell}(f_i g_{jk\ell} + f_{jk\ell} g_i) + \frac{1}{4}\,(\mu_{ijk\ell} - \mu_{ij}\mu_{k\ell})\, f_{ij} g_{k\ell}\right]$$

$$+ \sum \left[\frac{1}{24}\,\mu_{ijk\ell m}(f_i g_{jk\ell m} + f_{jk\ell m} g_i)\right.$$

$$\left. + \frac{1}{12}\,(\mu_{ijk\ell m} - \mu_{ij}\mu_{k\ell m})(f_{ij} g_{k\ell m} + f_{k\ell m} g_{ij})\right.$$

$$+ \sum \left[\frac{1}{120}\,\mu_{ijk\ell mn}(f_i g_{jk\ell mn} + f_{jk\ell mn} g_i)\right.$$

$$\left. + \frac{1}{48}\,(\mu_{ijk\ell mn} - \mu_{ij}\mu_{k\ell mn})\,(f_{ij} g_{k\ell m} + f_{k\ell m} g_{ij})\right.$$

$$\left. + \frac{1}{36}\,(\mu_{ijk\ell mn} - \mu_{ijk}\mu_{\ell mn})\, f_{ijk} g_{\ell mn}\right] + \ldots$$

where use has been made of the fact that there is summation over all indices.

The remaining task is to find the moments $\mu_{i_1 \ldots i_n}$ based on the distribution function ϕ for a Gaussian. It is the exponential of a quadratic form,

$$\phi(\vec{x}) = \frac{1}{(2\pi)^{n/2}\ |V|^{1/2}}\, e^{-\frac{1}{2}\,\widetilde{(\vec{x} - \vec{x})}\, V^{-1} (\vec{x} - \vec{x})} \,. \qquad \text{(A.17)}$$

Since a Gaussian is completely characterized by its mean and variance, we can express the moments, of any order, in terms of variances σ^2_{ij}. In fact, the even moments are symmetrized products of the variance, and the odd moments are zero,

$$\mu_{i_1 \ldots i_{2n}} = \frac{(2n)!}{n!\,2^n}\, \sigma^2_{[i_1 i_2} \cdots \sigma^2_{i_{2n-1} i_{2n}]} \qquad \text{(A.18a)}$$

$$\mu_{i_1 \cdots i_{2n+1}} = 0, \tag{A.18b}$$

where the bracketed subscripts indicate they are symmetrized.

Explicitly, the first few orders of the mean and variance formulas for a Gaussian are computed by substitution of (A.18) into (A.6)

$$\bar{f} = f_o + \frac{1}{2} \sum_{i,j} \sigma^2_{ij} f_{ij} + \frac{1}{8} \sum_{i,j,k,\ell} \sigma^2_{[ij}\sigma^2_{k\ell]} f_{ijk\ell} \tag{A.19}$$

$$+ \frac{1}{48} \sum_{i,j,k,\ell,m,n} \sigma^2_{[ij}\sigma^2_{k\ell}\sigma^2_{mn]} f_{ijk\ell mn} + \cdots ,$$

and for the variance by substituting (A.18) in (A.16),

$$\sigma^2_{fg} = \sum \sigma^2_{ij} f_i g_i \tag{A.20}$$

$$+ \sum [(\sigma^2_{ij}\sigma^2_{k\ell} + \sigma^2_{ik}\sigma^2_{j\ell} + \sigma^2_{i\ell}\sigma^2_{jk})(\tfrac{1}{6}(f_i g_{jk\ell} + f_{jk\ell} g_i)$$

$$+ \tfrac{1}{4} f_{ij} g_{k\ell}) - \tfrac{1}{4} \sigma^2_{ij}\sigma^2_{k\ell} f_{ij} g_{k\ell}]$$

$$+ \sum [(\sigma^2_{ij}\sigma^2_{k\ell}\sigma^2_{mn} + \ldots) - [\tfrac{1}{120} (f_i g_{ik\ell mn} + f_{jk\ell mn} g_i)$$

$$+ \tfrac{1}{48} (f_{ij} g_{k\ell mn} + f_{k\ell mn} g_{ij}) + \tfrac{1}{36} f_{ijk} g_{\ell mn}]$$

$$- \tfrac{1}{48} \sigma^2_{ij}(\sigma^2_{k\ell}\sigma^2_{mn} + \sigma^2_{kn}\sigma^2_{\ell m} + \sigma^2_{km}\sigma^2_{\ell n}) (f_{ij} g_{k\ell mn} + f_{k\ell mn} g_{ij})]$$

$$+ \ldots$$

For most purposes, the functions are close enough to linear that we may

take just the first term

$$\bar{f} = f_o \tag{A.21}$$

$$\sigma_{fg}^2 = \sum \sigma_{ij}^2 f_i g_j . \tag{A.22}$$

b. Application of the Propagation of Errors Techniques to Accelerator Design

Having seen how to propagate errors in the general case, we are now ready to apply the method to the problem of accelerator design. It may be applied to almost any accelerator design code, but, naturally, I will apply it to the Lie algebraic techniques described in this thesis and by others (Dragt [1981], Douglas [1982], Dragt et. al. [1985]). Although there are several stages to the computation of specifying a lattice, computing the maps, and calculating the lattice functions, we may reduce this to a few mathematical operations. In particular, we divide the process into three tasks: map generation, concatenation, and lattice function computation.

i. Generation of Maps

For each type of beamline element, such as a quadrupole or drift segment, a map, consisting of a matrix and polynomial coefficient, is calculated based on parameters supplied. By computing their derivatives with respect to the parameters, we will be able to propagate the statistical effect of these parameters to the maps.

Assume for instance, that the matrix is a function of the para-

meters $\alpha_1, \ldots, \alpha_n$

$$M(\alpha_1, \ldots, \alpha_n) = \begin{bmatrix} m_{11}(\alpha_1, \ldots, \alpha_n) & m_{12}(\alpha_1, \ldots, \alpha_n) & m_{16}(\alpha_1, \ldots, \alpha_n) \\ m_{21}(\alpha_1, \ldots, \alpha_n) & m_{22}(\alpha_1, \ldots, \alpha_n) & \vdots \\ \vdots & & \\ m_{61}(\alpha_1, \ldots, \alpha_n) & \cdots & m_{66}(\alpha_1, \ldots, \alpha_n) \end{bmatrix} \tag{A.23}$$

From the derivation above, the mean value is

$$\bar{M} = M(\bar{\alpha}_1, \ldots, \bar{\alpha}_n) \ . \tag{A.24}$$

If we have the derivative matrices

$$\frac{\partial M(\alpha_1, \ldots, \alpha_n)}{\partial \alpha_i} = \begin{bmatrix} \frac{\partial m_{11}}{\partial \alpha_i} & \frac{\partial m_{12}}{\partial \alpha_i} & \cdots & \frac{\partial m_{16}}{\partial \alpha_i} \\ \frac{\partial m_{21}}{\partial \alpha_i} & \frac{\partial m_{22}}{\partial \alpha_i} & & \cdot \\ & & & \cdot \\ \cdot & & & \cdot \\ \cdot & & & \\ \cdot & & & \\ \frac{\partial m_{61}}{\partial \alpha_i} & \cdot \ \cdot \ \cdot & & \frac{\partial m_{66}}{\partial \alpha_i} \end{bmatrix} \tag{A.25}$$

we may calculate the covariance matrix between the elements.

$$\sigma^2_{m_{ij} m_{k\ell}} = \sum_A \sigma^2_{\alpha_n} \frac{\partial m_{ij}}{\partial \alpha_n} \frac{\partial m_{k\ell}}{\partial \alpha_n} \tag{A.26}$$

Note that the 6x6 = 36 elements of the matrix are merely 36

separate functions from a statistical analysis point of view. Thus the covariance matrix was 36x36 = 1296 quantities, of which 18x37 = 666 are independent.

As an example on a smaller matrix, consider the simple 2x2 phase advance matrix

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} , \qquad (A.27)$$

with θ a statistical parameter having mean $\bar{\theta}$ and variance σ_θ^2. Then the mean is the matrix evaluated at $\bar{\theta}$,

$$\bar{M} \approx \begin{bmatrix} \cos\bar{\theta} & -\sin\bar{\theta} \\ \sin\theta & \cos\theta \end{bmatrix} . \qquad (A.28)$$

The covariances can be obtained from the derivative matrix

$$\frac{\partial M}{\partial \theta} = \begin{bmatrix} -\sin\theta & -\cos\theta \\ \cos\theta & -\sin\theta \end{bmatrix} , \qquad (A.29)$$

and they are

$$\sigma^2_{m_{11}} = \sigma^2_{m_{22}} = \sigma_\theta^2 \sin^2\theta \qquad (A.30a)$$

$$\sigma^2_{m_{12}} = \sigma^2_{m_{21}} = \sigma_\theta^2 \cos^2\theta \qquad (A.30b)$$

$$\sigma^2_{m_{11}m_{12}} = \sigma_\theta^2 \sin\theta \cos\theta \qquad (A.30c)$$

$$\sigma^2_{m_{11}m_{21}} = -\sigma_\theta^2 \sin\theta \cos\theta \qquad (A.30d)$$

$$\sigma^2_{m_{11}m_{22}} = \sigma^2_\theta \sin^2\theta \qquad \text{(A.30e)}$$

$$\sigma^2_{m_{12}m_{21}} = -\sigma^2_\theta \cos^2\theta \qquad \text{(A.30f)}$$

$$\sigma^2_{m_{12}m_{22}} = \sigma^2_\theta \sin\theta \cos\theta \qquad \text{(A.30g)}$$

$$\sigma^2_{m_{21}m_{22}} = -\sigma^2_\theta \sin\theta \cos\theta \qquad \text{(A.30h)}$$

The mean and variance of the polynomial coefficients may be calculated in the same way. Of course, there will be covariances among themselves and with the matrix, so altogether there will be for computations through fourth order polynomials

$$36 \text{ (matrix elements)} + 6 \text{ (first-order polynomials)} \qquad \text{(A.31)}$$

$$+ 56 \text{ (third-order polynomials)}$$

$$+ 126 \text{ (fourth-order polynomials)} = 224 \text{ quantities}$$

$$\frac{224}{2} \times (224 + 1) = 25{,}200 \text{ independent covariances.}$$

ii. Concatenation

As we saw in Chapter 4, concatenation involves much computation: matrix multiplication, moving the first-order term, and so on. However, if we know the propagation of errors formulae for addition and multiplication, this task is reduced to finding the propagation of errors for matrix multiplication, Poisson bracketing, and transforming. Each of

these, in turn, is again essentially just multiplication and addition.

Consider first the multiplication of two matrices M and N to form P

$$P = MN, \tag{A.32}$$

where the elements are P_{ij}, M_{ij}, r_{ij},

$$p_{ij} = \sum_r m_{ir} n_{rj}. \tag{A.33}$$

Further, let us assume that both M and N are subject to independent statistical error. By the covariance propagation formula (A.22), we may find the covariance in P to first order, as a function of M and N involving only products and sums,

$$\sigma^2_{p_{ij}p_{k\ell}} = \sum_{r,s} (\sigma^2_{m_{ir}m_{ks}} n_{rj}n_{s\ell} + \sigma^2_{n_{rj}n_{s\ell}} m_{ir}m_{ks}). \tag{A.34}$$

The mean of an element is, to first order, just the sum of the products of the means (A.21),

$$\bar{p}_{ij} = \sum_r \bar{m}_{ir}\bar{m}_{rj} . \tag{A.35}$$

For this relationship, a first order formula is reasonable, as multiplication and addition are not too nonlinear!

Taking the Poisson bracket of two polynomials involves just differentiation with respect to phase space, multiplication, and addition. If we let the $f_{i_1 \ldots i_n}$ be the coefficient of the monomial $\zeta_{i_1} \ldots \zeta_{i_n}$ in the polynomial f, and similarly for $g_{j_1 \ldots j_n}$ for the polynomial g,

then if h = [f,g] the Poisson bracket may be expanded

$$h = \int f_{i_1 \cdots i_n} g_{j_1 \cdots j_n} [\zeta_{i_1} \cdots \zeta_{i_n}, \zeta_{j_1} \cdots \zeta_{j_m}] \, dx. \quad (A.36)$$

Since the statistical quantities are the coefficients $f_{i_1 \cdots i_n}$ and $g_{j_1 \cdots j_n}$, the mean value is given in lowest order by the value at the mean,

$$\bar{h} = \int \bar{f}_{i_1 \cdots i_n} \bar{g}_{j_i \cdots j_m} [\zeta_{i_1} \cdots \zeta_n, \zeta_{j_1} \cdots \zeta_{j_m}] \, dx \quad (A.37)$$

or

$$\bar{h} = [\bar{f}, \bar{g}] \, . \quad (A.38)$$

A particular coefficient of h is the sum over particular products of the coefficients of f and g.

$$h_{k_1 \cdots k_{m+n-2}} = \sum n_{i_1 \cdots i_n j_1 \cdots j_m} f_{i_1 \cdots i_n} g_{j_1 \cdots j_m} \quad (A.39)$$

where n is an integration factor from the Poisson bracket. The covariance between two coefficients of h depends on f and g

$$\sigma^2_{h_{k_1 \cdots k_{m+n-2}} h_{\ell_1 \cdots \ell_{m+n-2}}} = \quad (A.40)$$

$$\sum n_{i_1 \cdots i_n j_1 \cdots j_n} n_{i'_1 \cdots i'_n j'_1 \cdots j'_m}$$

$$(\sigma^2_{f_{i_1 \cdots i_n} f_{i'_1 \cdots i'_n}} g_{j_1 \cdots j_m} g_{j'_1 \cdots j'_m}$$

$$+ \sigma^2_{f_{i_1} \cdots i_n g_{j_1} \cdots j_m} f_{i_1' \cdots i_n'} g_{j_1' \cdots j_m'} + \cdots$$

$$+ \sigma^2_{g_{j_1} \cdots j_m g_{j_1'} \cdots j_m'} f_{i_1} \cdots i_n f_{i_1'} \cdots i_n)$$

The final operation on which we need to know how to propagate errors is the process of transforming a polynomial by a matrix,

$$f^T(\zeta) = f(M\zeta). \tag{A.41}$$

The coefficients of f^T are related to the coefficients of f by the matrix elements

$$f^T_{i_1 \cdots i_n} = \sum_{j_1 \cdots j_n} f_{j_1 \cdots j_n} M_{i_1 j_1} M_{i_2 j_2} \cdots M_{i_n j_n}. \tag{A.42}$$

Thus in lowest order the mean is

$$\bar{f}^T(\zeta) = \bar{f}(\bar{M}\zeta), \tag{A.43}$$

where $\bar{f}$ is the polynomial with coefficients equal to the mean $\bar{f}_{j_1 \cdots j_n}$. The covariance is

$$\sigma^2_{f^T_{i_1 \cdots i_n} f^T_{i_1' \cdots i_n'}} \tag{A.44}$$

$$= \sum_{\substack{j_1 \cdots j_n \\ j_1' \cdots j_n'}} [\sigma^2_{f_{j_1 \cdots j_n' f_{j_1' \cdots j_n'}}} \times (M_{i_1 j_1} M_{i_2 j_2} \cdots M_{i_1 j_n} M_{i_1' j_1'} \cdots M_{i_n' j_n'})$$

$$+ \sigma^2_{f_{j_1} \cdots j_n} M_{i_1 j_1} (f_{j_1' \cdots j_n'} M_{i_2 j_2} \cdots M_{i_n j_n} M_{i_1' j_1'} \cdots M_{i_n' j_n'}) + \ldots].$$

iii. Lattice Functions

The ultimate goal of constructing the map for a lattice is usually not the map itself, but some function of it, such as the tune or its effect on particles. If we know the mean and covariance of the map, it will be possible to determine the mean and covariance of these lattice functions. As an example, I will consider the case of the tune in an accelerator with midplane symmetry.

For an accelerator with midplane symmetry, the horizontal and vertical degrees of freedom are uncoupled, so the tune is

$$\nu_i(M) = \frac{1}{2\pi} \arccos\left(\frac{tr_i(M)}{2}\right) \qquad (A.45)$$

where $tr_i(M)$ is the trace of the i^{th} degree of freedom of matrix M, e.g. $tr_1(M) = M_{11} + M_{22}$. The tune function is dependent only on the linear part of the map, the matrix.

The tune function, unfortunately, is not very linear, especially at values of the trace near ± 2 (see Figure A.1). Therefore, we shall have to do a nonlinear propagation of errors. The derivatives, as a function of the trace of the small matrix, are given in Table A.1. We may then say, according to (A.19), that the mean is

$$\bar{\nu}_i = \frac{1}{2\pi} \arccos\left(\frac{\bar{x}_i}{2}\right) - \frac{1}{2} \frac{1}{16\pi} s_i^2 \frac{1}{(1 - \frac{x^2}{4})^{3/2}} \ldots \qquad (A.46)$$

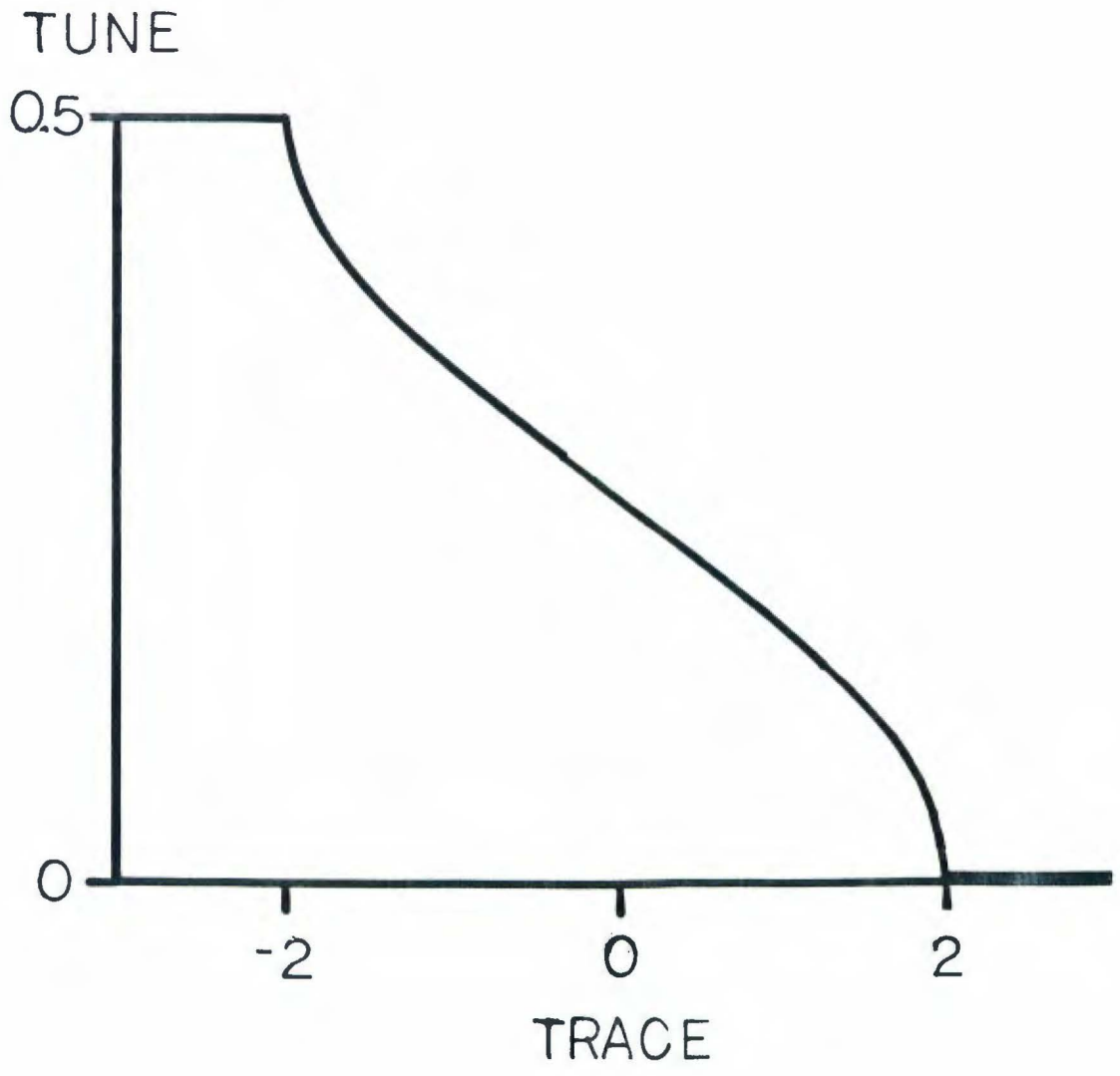


Figure A.1 The Tune Function

and the variance from (A.20) is

$$\sigma^2_{\nu_i} = \frac{s_i^2}{16\pi^2} \frac{1}{1 - \frac{\bar{x}_i^2}{4}} + s_i^4 \left(\frac{1 + \frac{\bar{x}_i^2}{2}}{64\pi^2 \left(1 - \frac{\bar{x}_i^2}{4}\right)^3} - \frac{1}{2} \frac{\bar{x}_i^2}{64\pi^2 \left(1 - \frac{\bar{x}_i^2}{4}\right)^3} \right) + \ldots, \tag{A.47}$$

where

$$x_i = tr_i(M), \quad \bar{x}_i = tr_i(\bar{M}) \tag{A.48}$$

and

$$s_i^2 = \sigma^2_{M_{ii}} + \sigma^2_{M_{i+1,i+1}} + 2\sigma^2_{M_{ii}M_{i+1,i+1}} \tag{A.49}$$

is the variance of the trace.

Table A.1 Derivatives of the Tune as a Function of the Trace of the Small Matrix

	Function	Value for x = -1.81958
0)	$f(x) = \frac{1}{2\pi} \arccos\left(\frac{x}{2}\right)$	0.43187874
1)	$f'(x) = -\frac{1}{4\pi\sqrt{1 - \frac{x^2}{4}}}$	- .191723
2)	$f''(x) = -\frac{x}{16\pi\,(1 - \frac{x^2}{4})^{3/2}}$	.506235
3)	$f'''(x) = -\frac{1 + \frac{x^2}{2}}{16\pi\,(1 - \frac{x^2}{4})^{5/2}}$	- 4.28829
4)	$f^{(4)}(x) = -\frac{18x + 3x^3}{128\pi\,(1 - \frac{x^2}{4})^{7/2}}$	59.5537
5)	$f^{(5)}(x) = -\frac{18 + 36x^2 + 3x^4}{128\pi\,(1 - \frac{x^2}{4})^{9/2}}$	- 1156.66

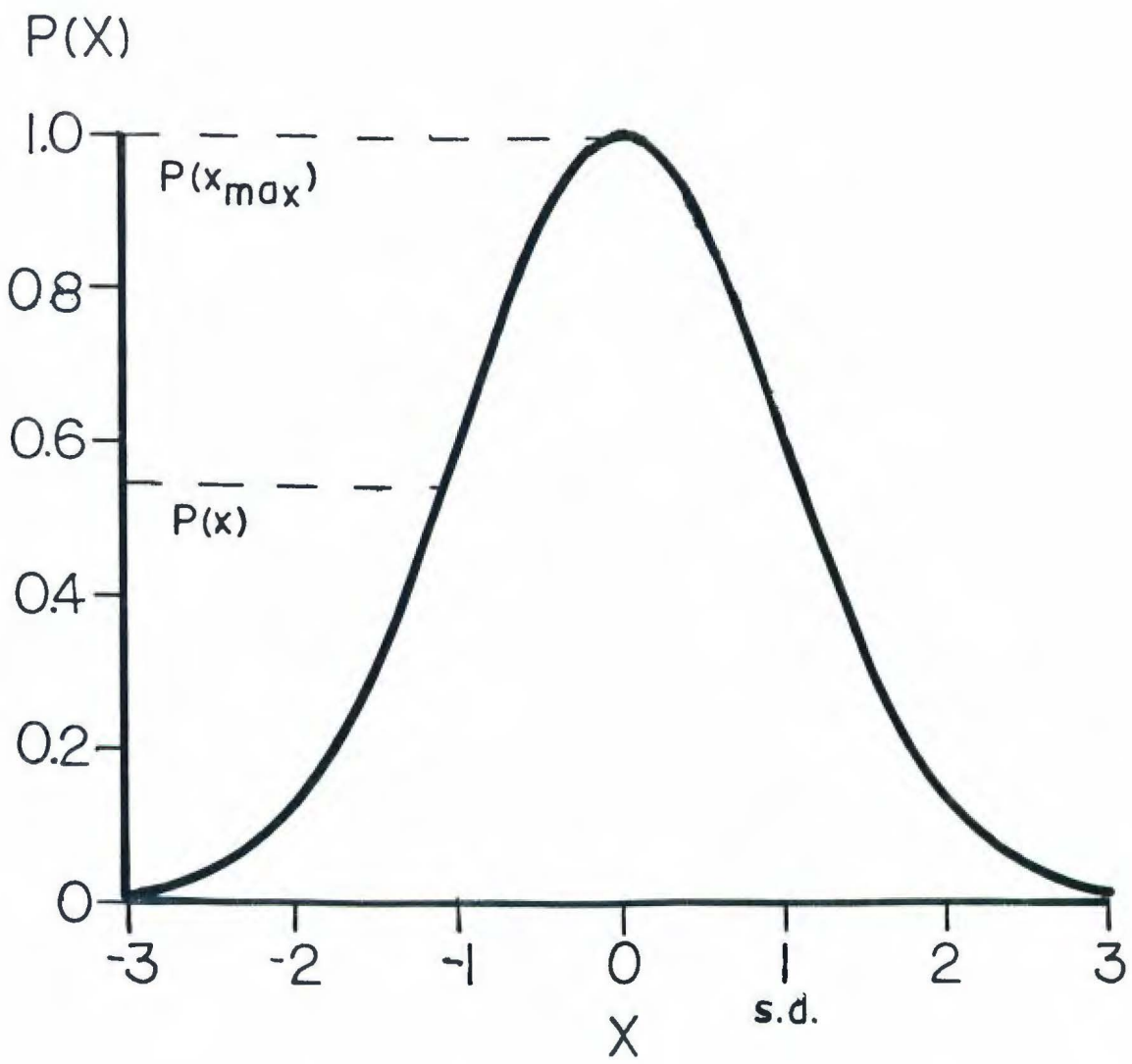
c. Results

The statistical computations described above, including maps and their derivatives, matrix multipliation, and tune computation were coded in order to compare the statistical method with the results from varying samples of data.

In order to make such comparisons, one must have a measure of how close the mean and covariances obtained from the propagation of errors is to that of the sample data. Such a measure comes under the general heading of **test statistics.** A test statistic is a computation that takes a theoretical distribution and an actual result and gives a number between 0 and 1 indicating how close they are, 1 being very close. The test statistic I use is the maximum likelihood ratio (Eadie et. al. [1971]). Given the theoretical probability distribution, one takes the ratio of the probability density at the actual value to that of the most likely value (see Figure A.2). A low number indicates poor agreement between theory and the actual results, a high number good agreement.

For a Gaussian distribution of a random variable which we are assuming, the probability distribution of a sampled mean of these variables is what is called a t-distribution (Eadie [1971], Bevington [1969]) and the probability distribution of a sampled covariance is a χ^2 distribution. Both these distributions are well-known and tabulated (see, for example, the references given), making computation of the maximum likelihood ratio straightforward.

To apply the maximum likelihood ratio to a covariance matrix, we will need to look at the covariances in normal coordinates, that is, coordinates where there is no covariance between different elements,



$$MLR = \frac{P(X)}{P(X_{max})}$$

Figure A.2 The Maximum Likelihood Ratio

only variances (covariances between identical elements). This means that we are interested in the diagonalized covariance matrix. We then may apply the maximum likelihood ratio to these eigenvalues to judge the relative validity of the sampling and theory.

As an actual test, we may propose looking at a simple bending cell (see Figure A.3) consisting of a horizontally focusing quadrupole, a drift, a parallel-face bend, a drift and a horizontally defocusing quadrupole. We suppose that the quadrupoles have strengths that vary according to a Gaussian distribution about some mean. This means that the design trajectory will remain the same as these parameters will vary, so there will be no first-order term in the Lie transformation. We may then look at how well the means of the matrix elements and the tunes agree with the theoretical predictions, and how well the eigenvalues of the covariance matrix agree, according to the maximum likelihood ratio.

One can imagine several diffrent maps of generating sample distributions of parameters in accordance with the parent Gaussian distribution. I have selected two, a pseudo-random number generator such as is typically part of numerical computation packages, and a regular distribution, obtained by dividing the integral into n equal intervals and using variable value at the center of each interval (see Figure A.4). The latter method produces a regular set of data, with values clustered in the area of highest probability density.

The results of a particular run are shown in Table A.2, for both kinds of samples. From this result and many others, we may draw a few conclusions. First of all, it is necessary when doing the propagation

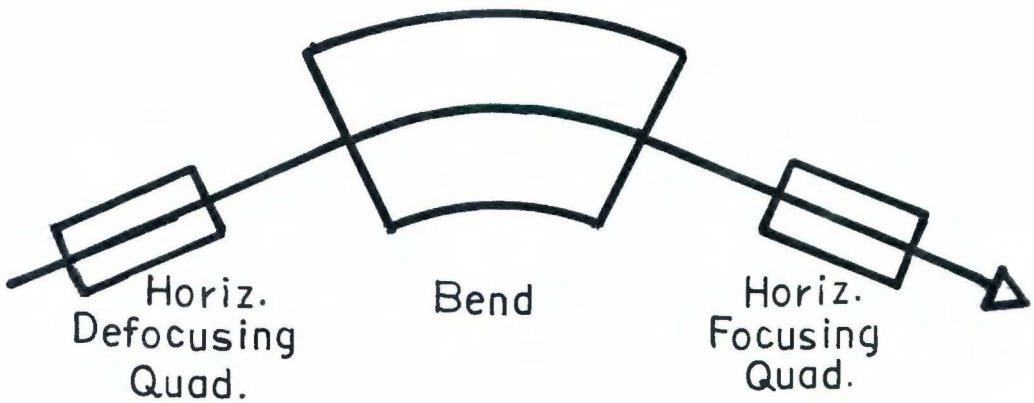


Figure A.3 A Bending Cell

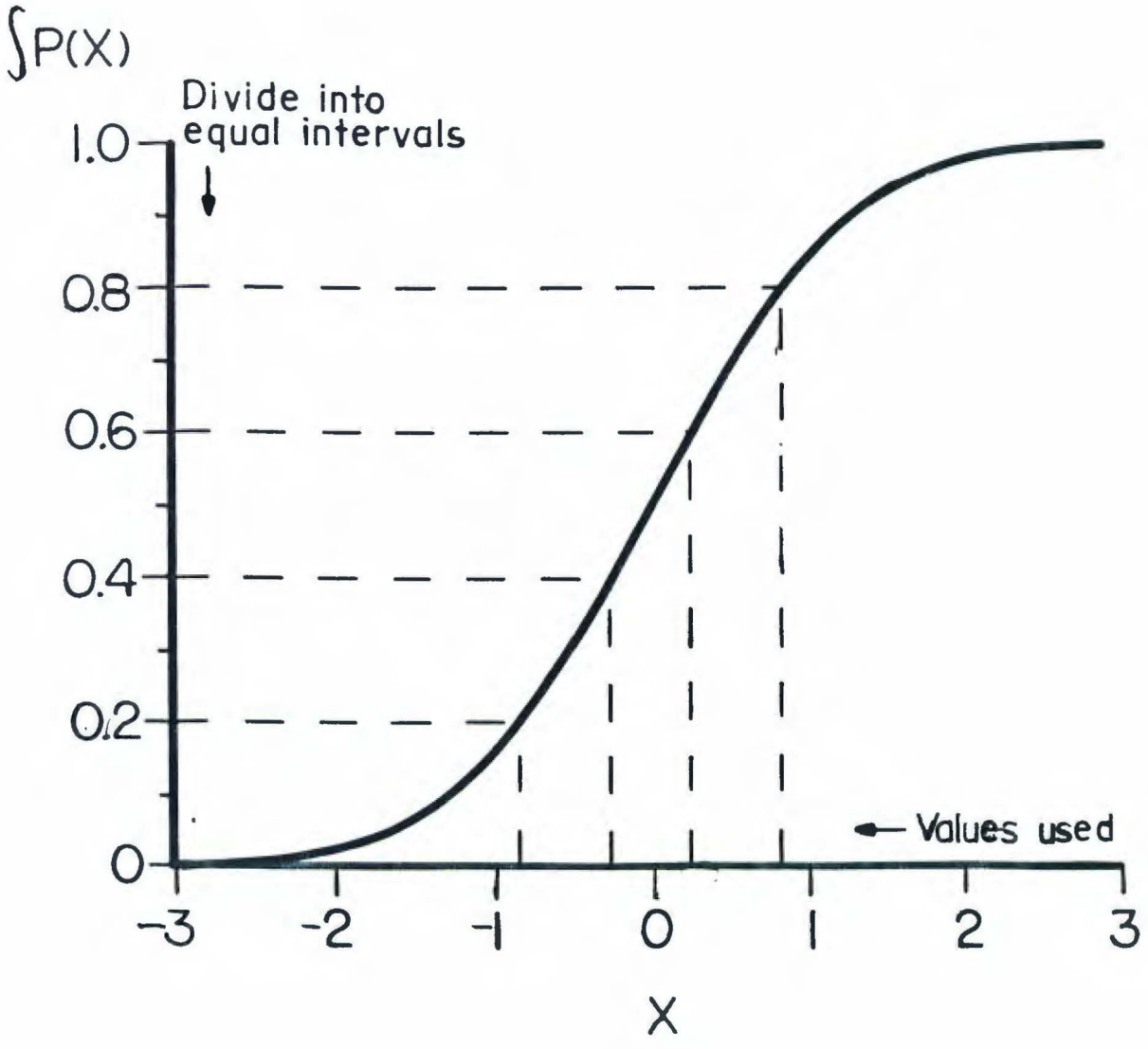


Figure A.4 Generation of a Regularly Distributed Sample of a Gaussian

of errors on the tune, to include higher B orders in the calculation, at least near trace values of ±2 where the arccosine function becomes very nonlinear. Second, in doing a sampling with varying parameters, it is better to generate them regularly than randomly, at least with a pseudo-random number generator of the type commonly available. Last, the validity of a sample remains roughly constant if the number of points sampled goes as the power of the number of parameters, b^n where n is the number of parameters and b is the base, and it should be at least 5 or 10 for a maximum likelihood ratio above 0.9. Thus, for two varying parameters, 25 would be the smallest reasonable sample size, and for four, 625 would be the smallest.

The method of propagation of errors extends the Lie algebraic accelerator design techniques so that not only is behavior at a particular set of parameters known, but behavior nearby is known as well. It has potential for development because of its advantages over a Monte Carlo simulation: it requires far less computation for equivalent validity of the answer, it is not subject to the possible unreliability of a random number generator, and it gives directly useful information, to wit, the mean and variance of desired quantities. Regular simulations are better than Monte Carlo in that they do not rely on a random number generator and give a more reliable answer with the same number of samples, but still are not as advantageous as the propagation of errors method.

Table A.2 Comparison of Propagation of Errors with Random and Regularly Generated Samples

Horizontal defocusing quad strength K = -2.19652710 , σ_K^2 = .0001

Horizontal focusing quad strength K = 3.7453240 , σ_K^2 = .0001

Regular sample used 10 points of each strength (10 x 10 = 100), random sample used 1000 points total. Seed for the random number generator was 6548804.

One term in propagation of errors formula for tune:

	Theory	Regular (mlr)	Random (mlr)
Tune x mean	0.4318793499	0.4319077293 (0.9903)	.4319438640 (0.6219)
variance	0.4177951×10^{-5}	0.4186980×10^{-5} (0.9877)	0.4382925×10^{-5} (0.5317)
Tune y mean	0.0670334734	0.0669517969 (0.9716)	0.06697461177 (0.8738)
variance	0.1162271×10^{-4}	0.1170186×10^{-4} (0.9822)	0.1285695×10^{-4} (0.0648)

Three terms in propagation of errors formula for tune:

	Theory	Regular (mlr)	Random (mlr)
Tune x mean	0.4319082160	0.4319082160 (1.0000)	0.4319438640 (0.8649)
variance	0.4190343×10^{-5}	0.4186980×10^{-5} (0.9908)	0.4382925×10^{-5} (0.5922)
Tune y mean	0.06695118513	0.06695179690 (1.0000)	0.06697461177 (0.9789)
variance	0.1172369×10^{-4}	0.1170186×10^{-4} (0.9917)	0.1285695×10^{-4} (0.1014)

Appendix B. MARYLIE 3.1

The techniques for treating accelerator errors Lie algebraically that are described in the body of the text have been implemented and tested in a computer code called MARYLIE 3.1. The core of this code is an extension of MARYLIE 3.0, written by Douglas (Douglas [1982], Dragt et. al. [1985]), that implements Lie algebraic techniques for accelerators without errors through third order (polynomials through fourth order). In addition, MARYLIE 3.1 has a completely different user interface.

The first section of this appendix shows how to use some of the features of the code, with some examples. It is not intended to be a manual or to replace one, but rather to show how the formalism of treating accelerator errors looks to the user with a real problem to solve. The second section describes the implementation and testing of the techniques described in the body of the text into code, including some of the practical aspects, to give a better understanding of them. Finally, the third section contains a listing of some important routines of MARYLIE 3.1, along with descriptions of other routines that the listed ones call.

a. Usage and Examples

In this section, I shall describe how to specify misalignments, how to find the closed orbit and the map nearby, and how to correct for the closed orbit. Also, a description of the fitting routine which may be used for closed orbit correction is given, and some applications related to closed orbit correction demonstrated.

i. Misalignment

Misalignments are specified for an element in the same way as other parameters. The user gives the six parameters specifying the misalignment at the fiducial point of the element. The example given in Table B.1 is of a horizontally defocusing quadrupole that has been misaligned by a horizontal translation of 0.001 meters and rotated in the midplane by 0.0015 radians, at the fiducial point. After it has been specified, and the magnetic rigidity and speed have been specified, we may see what the transfer map looks like. The matrix and polynomial coefficients are given; the first six rows are the matrix, the following lines are the polynomial coefficients, with the monomial name in brackets (all upper case phase space variables used above have become lower case in the coding, and there no are subscripts). Note that there are coefficients of x, px, and pt in the map, present because of the misalignment. The coefficients of y and py present are insignificant.

Misalignments may be specified for almost any of the elements available in MARYLIE. The only exceptions are those for which a misalignment does not make sense, such as a drift or a phase advance.

Concatenation of maps, including those with first-order polynomials, is straightfoward. For example, suppose one has a simple bending cell (Table B.2) consisting of a horizontally defocusing quadrupole, a parallel face bending magnet, and a horizontally focusing quadrupole, with drift spaces interspersed. With the defocusing quadrupole misaligned as before, and the focusing quadrupole also misaligned, the map for each will contain a first-order part. When they are concate-

nated together into a cell called 'nsex,' the whole transfer map may be obtained, as easily as the map for a single element. The result is printed out in the table.

ii. Closed Orbit Determination

Closed orbit determination is specified as a change in the form of the map. As constructed from the element library or from the concatenator, the polynomials of the map are kept in ascending form, i.e.,

$$e^{:f_1:}\, e^{:f_2:}\, e^{:f_3:}\, e^{:f_4:} \,. \tag{B.1}$$

However, it is possible to put this in another form, the fixed point, or closed orbit, form

$$e^{-:g_1:}\, e^{:g_2:}\, e^{:g_3:}\, e^{:g_4:}\, e^{:g_1:}, \tag{B.2}$$

as shown in Chapter 6. This is acheived by specifying the function 'fix' of the map, and is demonstrated for the bending cell in the table. The first-order terms (x, px, y, py and pt) are the coefficients of the polynomial that gives the fixed point, that is,

$$z \equiv e^{:g_1:}\zeta = [g_1,\zeta] \tag{B.3}$$

is the fixed point. The map around z is given by the matrix and the other polynomial coefficients.

Table B.1 Example of MARYLIE: Misaligned Element

```
ML3.1> ! The magnetic rigidity is specified by brho, the speed by beta
ML3.1> brho=4.8691523
ML3.1> beta=0.8412103393
ML3.1> ! The following command specifies a defocusing quadrupole
ML3.1> ! that has been misaligned horizontally
ML3.1> ! by 0.001m and rotationally in the midplane by 0.0015 radians.
ML3.1> hdq:quad,l=0.5,k1=-1.93,delx=0.001,theta=0.0015
ML3.1> ! The following command prints out the matrix and polynomials
ML3.1> ! of the map
ML3.1> print hdq
1.0500      0.50830     -2.17602E-17 -1.12543E-17 0.00000E+00  6.84590E-05
0.20148     1.0500       4.46093E-18  -2.17602E-17 0.00000E+00  8.68903E-06
2.40280E-17  1.16322E-17  0.95086       0.49178  0.00000E+00  1.34754E-20
4.61074E-18  2.40280E-17  -0.19493      0.95086  0.00000E+00  1.93515E-20
4.66974E-06  6.74624E-05  -2.05329E-20 1.90462E-21  1.0000       0.20658
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.0000
hdq[x]            =-2.0209359818933772E-04
hdq[px]           =-5.6241980046548858E-05
hdq[y]            =-2.3290344846676129E-20
hdq[py]           =2.5559573247026451E-20
hdq[pt]           =3.5615143293835734E-06
hdq[x x x]        =4.6214861290262093E-05
hdq[x x px]       =-2.1917207889858402E-04
hdq[x x y]        =-3.4403977226576909E-21
hdq[x x py]       =-7.8909047684250828E-21
hdq[x x pt]       =-3.9687203779944684E-03
hdq[x px px]      =2.1717934959245682E-04
hdq[x px y]       =-1.8734093274946567E-20
hdq[x px py]      =1.9823610538152742E-20
hdq[x px pt]      =6.0870890129539595E-02
hdq[x y y]        =9.4400528189299484E-05
hdq[x y py]       =2.3905829251721772E-04
hdq[x y pt]       =-7.0562531767054717E-21
hdq[x py py]      =-1.6587874102751662E-04
hdq[x py pt]      =2.6969894874270836E-18
hdq[x pt pt]      =1.1950494045847783E-05
hdq[px px px]     =-6.8911612664317048E-05
hdq[px px y]      =1.2596808369202098E-20
hdq[px px py]     =-1.0170543647415764E-20
hdq[px px pt]     =-0.3071991624156031
hdq[px y y]       =1.0849232448812077E-04
hdq[px y py]      =-1.2513631221717794E-04
hdq[px y pt]      =2.6969565312405407E-18
hdq[px py py]     =2.7355807298381209E-05
hdq[px py pt]     =-8.9871782719335052E-19
hdq[px pt pt]     =-9.4004651982864068E-05
hdq[y y y]        =1.6090534230346103E-21
hdq[y y py]       =1.0602879804796601E-20
hdq[y y pt]       =-3.8144076084688497E-03
hdq[y py py]      =2.0250154137856189E-21
hdq[y py pt]      =-5.6979315032469157E-02
hdq[y pt pt]      =3.3097850160976309E-21
```

```
hdq[py py py]     =-4.7017994764514440E-21
hdq[py py pt]     =-0.2875645362891140
hdq[py pt pt]     =-6.4304978660762564E-21
hdq[pt pt pt]     =-0.1227849184152725
hdq[x x x x]      =-7.7232642324291876E-03
hdq[x x x px]     =1.88187766812041116E-02
hdq[x x x y]      =1.0889899520524252E-18
hdq[x x x py]     =-2.5415729608614839E-19
hdq[x x x pt]     =7.3091304098399213E-05
hdq[x x px px]    =-6.9031150958326567E-02
hdq[x x px y]     =-7.1797832985174532E-19
hdq[x x px py]    =-4.7444936325076198E-18
hdq[x x px pt]    =-6.8658144637723236E-04
hdq[x x y y]      =-3.9239410422525906E-02
hdq[x x y py]     =2.9924810965651031E-02
hdq[x x y pt]     =4.0447676825624633E-22
hdq[x x py py]    =4.4441235473335375E-02
hdq[x x py pt]    =-2.4718374141706286E-20
hdq[x x pt pt]    =-5.4555487090434210E-03
hdq[x px px px]   =5.6012685311786729E-02
hdq[x px px y]    =-2.2687191016075999E-19
hdq[x px px py]   =3.5908287253772188E-18
hdq[x px px pt]   =7.8681653765253221E-04
hdq[x px y y]     =2.8952662776853766E-02
hdq[x px y py]    =-1.9623869573405422E-02
hdq[x px y pt]    =-2.8252097436597739E-20
hdq[x px py py]   =3.7644254888865468E-03
hdq[x px py pt]   =4.1049697579192941E-20
hdq[x px pt pt]   =8.4131986462893835E-02
hdq[x y y y]      =-1.2875753337517813E-18
hdq[x y y py]     =1.0598906378722498E-18
hdq[x y y pt]     =8.7459329853886297E-05
hdq[x y py py]    =7.8280697226996276E-20
hdq[x y py pt]    =3.1429609575511220E-04
hdq[x y pt pt]    =-1.3862827713345885E-20
hdq[x py py py]   =1.0866438676299725E-18
hdq[x py py pt]   =3.9999323702321767E-06
hdq[x py pt pt]   =3.6762370541639613E-18
hdq[x pt pt pt]   =1.8712483635342766E-04
hdq[px px px px] =-7.1394465692224513E-02
hdq[px px px y]   =1.1994565036443791E-18
hdq[px px px py] =-7.8051354607002803E-19
hdq[px px px pt] =-1.5170823668890271E-03
hdq[px px y y]    =-5.4262436099911447E-02
hdq[px px y py]   =3.5997297571197237E-03
hdq[px px y pt]   =3.4275268890774707E-20
hdq[px px py py] =-0.1257360230031477
hdq[px px py pt] =-3.8163347387641213E-20
hdq[px px pt pt] =-0.4246497735280909
hdq[px y y y]     =3.3846527835228803E-19
hdq[px y y py]    =-4.4393298079974429E-18
hdq[px y y pt]    =-2.1920055390121209E-04
hdq[px y py py]   =3.2524204809391448E-18
hdq[px y py pt]   =-2.3483138660921952E-04
```

```
hdq[px y pt pt]  =3.6762751579009847E-18
hdq[px py py py] =-7.3546950511144508E-19
hdq[px py py pt] =-1.2757873870120113E-03
hdq[px py pt pt] =-1.7590983373390017E-18
hdq[px pt pt pt] =-1.8920035329196499E-03
hdq[y y y y]     =-5.5538277818209808E-03
hdq[y y y py]    =1.4162700637201822E-02
hdq[y y y pt]    =1.7125150669872634E-21
hdq[y y py py]   =3.2918877310855554E-02
hdq[y y py pt]   =7.6744897737884513E-22
hdq[y y pt pt]   =-5.1528002253951172E-03
hdq[y py py py]  =-4.3719368943504521E-02
hdq[y py py pt]  =7.3507699392116773E-21
hdq[y py pt pt]  =-7.6510909155198873E-02
hdq[y pt pt pt]  =1.0159315603294272E-20
hdq[py py py py] =-5.4833622754068101E-02
hdq[py py py pt] =-2.2253064289553352E-20
hdq[py py pt pt] =-0.3862154810733600
hdq[py pt pt pt] =-2.6518178631959275E-20
hdq[pt pt pt pt] =-0.1566346259487956
```

Table B.2 Example of MARYLIE: Concatenation with Misaligned Element and Determination of Fixed Point

```
ML3.1> ! Parameters
ML3.1> brho=4.8691523
ML3.1> beta=0.8412103393
ML3.1> ! Define drifts
ML3.1> drs:drift,0.45
ML3.1> drl:drift,2.28646
ML3.1> ! Bending magnet
ML3.1> pbnd:rbend,angle=0.6283185307,k0=1.2
ML3.1> ! Quads
ML3.1> hdq:quad,0.5,-1.92,delx=0.001,theta=0.0015
ML3.1> hfq:quad,0.5,2.72,dely=-0.002,phi=0.0005
ML3.1> ! Define the cell nsex
ML3.1> nsex:{drl hdq drs pbnd drs hfq drl}
ML3.1> print nsex
1.0491        6.3609     -8.15777E-04 4.15067E-03  0.00000E+00     -2.4108
-0.27947    -0.74130     -4.32238E-04 -2.55449E-04 0.00000E+00     -0.37315
-3.56525E-03 -1.96288E-02 -0.70776     6.1585     0.00000E+00   2.00561E-05
-4.42108E-04 -2.99573E-03 -0.27890     1.0139     0.00000E+00   2.83648E-04
1.0652        4.1607       9.32781E-04  4.38151E-04  1.0000         3.4959
0.00000E+00   0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00   1.0000
nsex[x]              =-2.0172525808729309E-04
nsex[px]             =-5.1902716457048420E-04
nsex[y]              =-6.5748730501532581E-07
nsex[py]             =2.0522902938269129E-03
nsex[pt]             =3.3216685097319039E-06
nsex[x x x]          =-1.1685467853048130E-02
nsex[x x px]         =4.9115827700417679E-02
nsex[x x y]          =-3.7508846463591030E-04
nsex[x x py]         =1.3515587093583536E-03
nsex[x x pt]         =-0.2921899009584240
nsex[x px px]        =0.1541293598102544
nsex[x px y]         =1.8459840991815392E-03
nsex[x px py]        =-5.8109273418567862E-03
nsex[x px pt]        =-1.053266552300671
nsex[x y y]          =-6.2567929197728024E-02
nsex[x y py]         =0.7199593929369587
nsex[x y pt]         =-1.8782870284975775E-03
nsex[x py py]        =-1.905835214730898
nsex[x py pt]        =1.0349651051635205E-02
nsex[x pt pt]        =-1.418381223580777
nsex[px px px]       =9.2273820159942378E-02
nsex[px px y]        =-5.7687961928001209E-03
nsex[px px py]       =2.2412575998832867E-02
nsex[px px pt]       =-3.066265995682219
nsex[px y y]         =-2.0895818716566702E-02
nsex[px y py]        =0.1761480429306482
nsex[px y pt]        =-1.9658215829637552E-03
nsex[px py py]       =2.5641756864780207E-02
nsex[px py pt]       =1.4344371769626050E-02
nsex[px pt pt]       =-3.087690940763916
nsex[y y y]          =-3.8682917243934783E-04
```

```
nsex[y y py]       =4.9328086194929029E-03
nsex[y y pt]       =-0.4971576016263518
nsex[y py py]      =-2.2633644651601651E-02
nsex[y py pt]      =4.414549844753906
nsex[y pt pt]      =-5.2034938613428767E-03
nsex[py py py]     =3.5869993137844974E-02
nsex[py py pt]     =-12.42649316007642
nsex[py pt pt]     =3.0178405085538552E-02
nsex[pt pt pt]     =-4.350038301724557
nsex[x x x x]      =-1.4006356421361361E-02
nsex[x x x px]     =8.2995446808865904E-02
nsex[x x x y]      =-7.5594388156762432E-05
nsex[x x x py]     =1.9236304176885807E-04
nsex[x x x pt]     =-7.7171281470707818E-02
nsex[x x px px]    =-0.7017707554868090
nsex[x x px y]     =3.3335985118406195E-04
nsex[x x px py]    =3.5079341329712458E-04
nsex[x x px pt]    =-0.3754456037467961
nsex[x x y y]      =-0.1299619123656771
nsex[x x y py]     =0.8400508327057035
nsex[x x y pt]     =-9.4199818018458537E-04
nsex[x x py py]    =-1.537169654433486
nsex[x x py pt]    =4.0009463272032682E-03
nsex[x x pt pt]    =-0.8195286205449076
nsex[x px px px]   =-5.1444402135832124E-02
nsex[x px px y]    =-1.8449049177966154E-03
nsex[x px px py]   =8.4056416703511137E-03
nsex[x px px pt]   =-1.797369248474826
nsex[x px y y]     =0.2237088369735704
nsex[x px y py]    =-0.6164200724772446
nsex[x px y pt]    =2.9266080076750372E-04
nsex[x px py py]   =-0.2196249527114565
nsex[x px py pt]   =2.5532874205083786E-03
nsex[x px pt pt]   =-2.831565236049871
nsex[x y y y]      =-5.2396933337770688E-05
nsex[x y y py]     =1.6675770125039356E-03
nsex[x y y pt]     =-0.4520924714228340
nsex[x y py py]    =-1.0456491049240043E-02
nsex[x y py pt]    =4.449141065244286
nsex[x y pt pt]    =-4.2798098789980154E-03
nsex[x py py py]   =1.8925799840188203E-02
nsex[x py py pt]   =-11.05749117982887
nsex[x py pt pt]   =2.4253106861265555E-02
nsex[x pt pt pt]   =-3.232368510198176
nsex[px px px px]  =-2.636812513968043
nsex[px px px y]   =9.2667477419446441E-03
nsex[px px px py]  =-3.5740554914225448E-02
nsex[px px px pt]  =-4.756024007088811
nsex[px px y y]    =-2.214163774812210
nsex[px px y py]   =18.42249270581254
nsex[px px y pt]   =1.1174047777924586E-02
nsex[px px py py]  =-40.72939786089794
nsex[px px py pt]  =-2.8721138973226055E-02
nsex[px px pt pt]  =-7.805283953512858
```

```
nsex[px y y y]     =-1.8875983256306527E-04
nsex[px y y py]    =3.8065596561437331E-03
nsex[px y y pt]    =-1.736046318494450
nsex[px y py py]   =-2.4641015781689582E-02
nsex[px y py pt]   =15.83800454819271
nsex[px y pt pt]   =1.4835421580820193E-02
nsex[px py py py]  =4.9913067760440442E-02
nsex[px py py pt]  =-36.26137089715613
nsex[px py pt pt]  =-3.9843257329945709E-02
nsex[px pt pt pt]  =-6.041142472460865
nsex[y y y y]      =-7.0539192719261391E-02
nsex[y y y py]     =1.204687843698255
nsex[y y y pt]     =-1.3159298582876629E-04
nsex[y y py py]    =-8.059116752940898
nsex[y y py pt]    =6.5894987840005124E-03
nsex[y y pt pt]    =-1.649095603672400
nsex[y py py py]   =24.49769714159961
nsex[y py py pt]   =-5.1207525118099098E-02
nsex[y py pt pt]   =16.46375453710148
nsex[y pt pt pt]   =-1.8652344236122379E-03
nsex[py py py py]  =-28.65309774423115
nsex[py py py pt]  =0.1099918173529561
nsex[py py pt pt]  =-43.97665684879331
nsex[py pt pt pt]  =2.9842432479775850E-02
nsex[pt pt pt pt]  =-7.378314076666465
ML3.1> ! The following command defines 'newmap' as the fixed point
ML3.1> ! form of nsex
ML3.1> newmap:fix(nsex)
ML3.1> print newmap
1.0491        6.3610        -7.27624E-04 3.94080E-03  0.00000E+00  -2.4109
-0.27939     -0.74088      -4.77134E-04 -1.43323E-04 0.00000E+00  -0.37298
-3.66227E-03 -2.05917E-02 -0.70734     6.1587    0.00000E+00  -8.36462E-05
-4.48541E-04 -3.06390E-03 -0.27882     1.0139    0.00000E+00  -2.33892E-04
1.0649        4.1588        7.36833E-04  3.17107E-03  1.0000        3.4953
0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  0.00000E+00  1.0000
newmap[x]            =-1.2163785669934853E-04
newmap[px]           =-7.4219599423172437E-04
newmap[y]            =-3.3784480106511362E-04
newmap[py]           =2.0679431223401606E-03
newmap[x x x]        =-1.1634876685829601E-02
newmap[x x px]       =4.8753596724315318E-02
newmap[x x y]        =-6.3480566207657209E-04
newmap[x x py]       =2.0812061540395292E-03
newmap[x x pt]       =-0.2920291725814404
newmap[x px px]      =0.1551366432380444
newmap[x px y]       =2.5464030142382336E-03
newmap[x px py]      =-7.1422496322291046E-03
newmap[x px pt]      =-1.052947588815111
newmap[x y y]        =-6.2345209613934738E-02
newmap[x y py]       =0.7186082862251167
newmap[x y pt]       =-2.1514329065539798E-03
newmap[x py py]      =-1.903506496648869
newmap[x py pt]      =1.1505487962646018E-02
newmap[x pt pt]      =-1.417236209970535
```

```
newmap[px px px]    =9.1026880641269469E-02
newmap[px px y]     =-8.7032929551061216E-03
newmap[px px py]    =3.3006283839682483E-02
newmap[px px pt]    =-3.066702791933980
newmap[px y y]      =-2.1592677307057182E-02
newmap[px y py]     =0.1809702474085750
newmap[px y pt]     =-3.9150268038297156E-03
newmap[px py py]    =1.6310976313448819E-02
newmap[px py pt]    =2.3554983623730158E-02
newmap[px pt pt]    =-3.086482726702191
newmap[y y y]       =-5.6244363788545997E-04
newmap[y y py]      =6.9397617783938767E-03
newmap[y y pt]      =-0.4969668714604797
newmap[y py py]     =-3.0975835542627141E-02
newmap[y py pt]     =4.412817299343964
newmap[y pt pt]     =-5.5754776523087279E-03
newmap[py py py]    =4.7420662700775558E-02
newmap[py py pt]    =-12.42280589920402
newmap[py pt pt]    =2.9114489647151106E-02
newmap[pt pt pt]    =-4.347915962023496
newmap[x x x x]     =-1.4006356421361361E-02
newmap[x x x px]    =8.2995446808865904E-02
newmap[x x x y]     =-7.5594388156762432E-05
newmap[x x x py]    =1.9236304176885807E-04
newmap[x x x pt]    =-7.7171281470707818E-02
newmap[x x px px]   =-0.7017707554868090
newmap[x x px y]    =3.3335985118406195E-04
newmap[x x px py]   =3.5079341329712458E-04
newmap[x x px pt]   =-0.3754456037467961
newmap[x x y y]     =-0.1299619123656771
newmap[x x y py]    =0.8400508327057035
newmap[x x y pt]    =-9.4199818018458537E-04
newmap[x x py py]   =-1.537169654433486
newmap[x x py pt]   =4.0009463272032682E-03
newmap[x x pt pt]   =-0.8195286205449076
newmap[x px px px]  =-5.1444402135832124E-02
newmap[x px px y]   =-1.8449049177966154E-03
newmap[x px px py]  =8.4056416703511137E-03
newmap[x px px pt]  =-1.797369248474826
newmap[x px y y]    =0.2237088369735704
newmap[x px y py]   =-0.6164200724772446
newmap[x px y pt]   =2.9266080076750372E-04
newmap[x px py py]  =-0.2196249527114565
newmap[x px py pt]  =2.5532874205083786E-03
newmap[x px pt pt]  =-2.831565236049871
newmap[x y y y]     =-5.2396933337770688E-05
newmap[x y y py]    =1.6675770125039356E-03
newmap[x y y pt]    =-0.4520924714228340
newmap[x y py py]   =-1.0456491049240043E-02
newmap[x y py pt]   =4.449141065244286
newmap[x y pt pt]   =-4.2798098789980154E-03
newmap[x py py py]  =1.8925799840188203E-02
newmap[x py py pt]  =-11.05749117982887
newmap[x py pt pt]  =2.4253106861265555E-02
```

```
newmap[x pt pt pt]  =-3.232368510198176
newmap[px px px px] =-2.636812513968043
newmap[px px px y]  =9.2667477419446441E-03
newmap[px px px py] =-3.5740554914225448E-02
newmap[px px px pt] =-4.756024007088811
newmap[px px y y]   =-2.214163774812210
newmap[px px y py]  =18.42249270581254
newmap[px px y pt]  =1.1174047777924586E-02
newmap[px px py py] =-40.72939786089794
newmap[px px py pt] =-2.8721138973226055E-02
newmap[px px pt pt] =-7.805283953512858
newmap[px y y y]    =-1.8875983256306527E-04
newmap[px y y py]   =3.8065596561437331E-03
newmap[px y y pt]   =-1.736046318494450
newmap[px y py py]  =-2.4641015781689582E-02
newmap[px y py pt]  =15.83800454819271
newmap[px y pt pt]  =1.4835421580820193E-02
newmap[px py py py] =4.9913067760440442E-02
newmap[px py py pt] =-36.26137089715613
newmap[px py pt pt] =-3.9843257329945709E-02
newmap[px pt pt pt] =-6.041142472460865
newmap[y y y y]     =-7.0539192719261391E-02
newmap[y y y py]    =1.204687843698255
newmap[y y y pt]    =-1.3159298582876629E-04
newmap[y y py py]   =-8.059116752940898
newmap[y y py pt]   =6.5894987840005124E-03
newmap[y y pt pt]   =-1.649095603672400
newmap[y py py py]  =24.49769714159961
newmap[y py py pt]  =-5.1207525118099098E-02
newmap[y py pt pt]  =16.46375453710148
newmap[y pt pt pt]  =-1.8652344236122379E-03
newmap[py py py py] =-28.65309774423115
newmap[py py py pt] =0.1099918173529561
newmap[py py pt pt] =-43.97665684879331
newmap[py pt pt pt] =2.9842432479775850E-02
newmap[pt pt pt pt] =-7.378314076666465
```

iii. Fitting Routines and Closed Orbit Correction

MARYLIE 3.1 has an option for fitting any set of parameters so that an equal number of target functions are minimized. For instance, one may adjust the vertical and horizontal quadrupole strengths in a focusing cell so that the two tunes assume certain desired values. Table B.4, discussed below, has a demonstration of such a calculation.

The routines that do the fitting are rather involved and so are not listed here. The process is essentially the following, however. First, initial guesses of the parameters that are to be varied are given by the user. The target functions are evaluated at these initial values and then at nearby values so that the Jacobian matrix may be calculated. This gives a linear approximation to the targets as a function of the parameters. From this approximation, it is possible to extrapolate to the desired values. The new value will not be exact but will be closer if the functions are reasonably linear. This process may be continued until the target functions are close enough to the desired values, as dictated by some previously specified tolerance parameter.

With the closed orbit (fixed point) determination implemented as described in Chapter 6 and the fitting facility available, we may easily implement a closed orbit correction scheme such as the beam bump method (Myers [1984], Guignard [1970]). This method is designed to insure that the design beam passes through the center of each quadrupole, even the misaligned ones. Let us consider the example of a mythical four-sided ring (four right-angle bends) for storing protons at 797 MeV. Suppose each cell from the center of one side to the center of the next is a FODO cell. Suppose further that the horizontally focusing quadrupole in

one of the cells (call it number 1) has been moved horizontally by 1 cm. With kickers (orbit correction dipoles in the short-length approximation, see (10.38)) adjacent to each quadrupole, we wish to know what strengths to give these correction magnets so that the beam goes through the center of each quadrupole. Since we are only misaligning the horizontal plane, we may ignore the corrections in the vertical. Thus we have four correctors for four quadrupoles.

The first step is to set up a file with the definitions of the beam line elements and the cells (Table B.3). The next step is to fit the quadrupole strengths so that the tunes are the desired values (ν_X=0.19, ν_Y=0.185). The actual process is shown in Table B.4. Once this has been done, we may perform the beam bump process. First, we work with the collection of elements consisting of cell 4, the one prior to the one with the misalignment, and the kicker adjacent to its horizontal quadrupole. We fit so that the design trajectory is sent to 1 cm on the X axis at the misaligned quadrupole. Because 'adj1' is defined as the concatenated map of these elements factored in descending order, this is accomplished by demanding that 'adj1' be -0.01 px. When this map is applied to the phase-space values zero, the linear and higher-order part will be not change it, and the -0.01 px term will take zero to 0.01 x, the center of the misaligned magnet.

Once this has been done, and we are assured the beam is going through the center of the misaligned quadrupole, we may look at the fixed point of the half ring from the horizontal kicker prior to cell 1 to the beginning of cell 2, and fit so that this is zero. Then a beam on the design trajectory starting into the cell 1 will be deflected so that it enters the misaligned quad at the center, then deflected so that

at cell 2 it is back on the design trajectory. This final fitting is shown in the last part of Table B.4, together with the resultant strengths of each of the horizontal kickers.

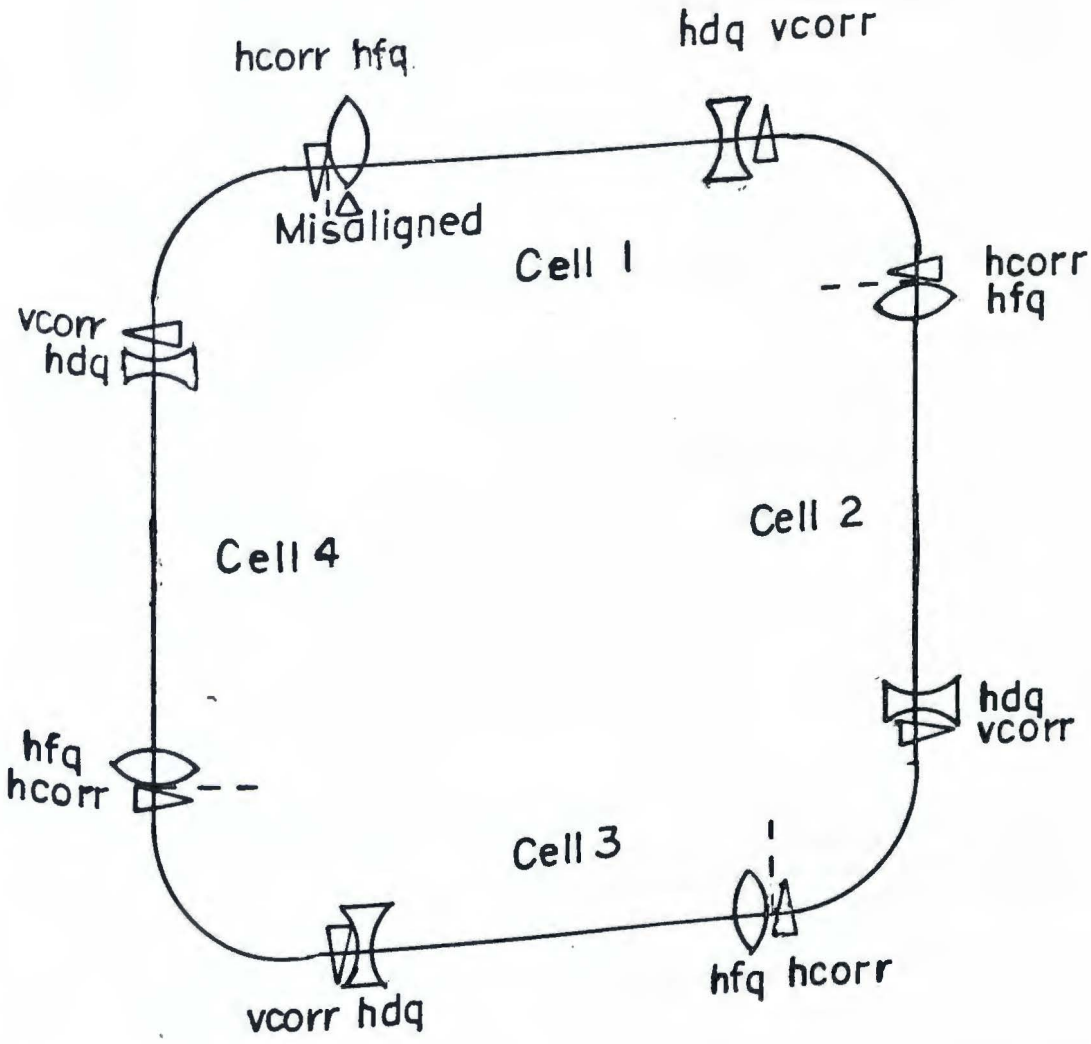


Figure B.1 A Mythical Four-Sided Ring for Demonstration of Closed Orbit Correction

Table B.3 Example of Fitting and Orbit Correction: file FOURSIDE

```
! These following describes a mythical four-sided ring
! that stores protons at 797 MeV.
! One of the horizontally focusing quadrupoles has been moved 1 cm
! horizontally.

! Global parameters
brho=4.8691523159713
beta=.84121033934903

! Drifts
drks:drift,0.2  ! for use with kickers of length .25 in drs
drs:drift,0.45
drm:drift,1.48646
drl:drift,2.28646

! Bends
nbnd:sbend,l=2.5494821908441938,angle=3.141592653589793/4
hcorr1:hkick,l=0.25,k0=strh1; strh1=0
vcorr1:vkick,l=0.25,k0=strv1; strv1=0
hcorr2:hkick,l=0.25,k0=strh2; strh2=0
vcorr2:vkick,l=0.25,k0=strv2; strv2=0
hcorr3:hkick,l=0.25,k0=strh3; strh3=0
vcorr3:vkick,l=0.25,k0=strv3; strv3=0
hcorr4:hkick,l=0.25,k0=strh4; strh4=0
vcorr4:vkick,l=0.25,k0=strv4; strv4=0

! Quads
klf=0.4
hfq:quad,0.5,klf                    ! Aligned horizontal focusing quad
badhfq:quad,0.5,klf,delx=0.01       ! This quad has been misaligned
                                    ! 1 cm in the horizontal direction
kld=-1.6
hdq:quad,0.5,kld                    ! Aligned horizontal defocusing quad

! Single Cell
dcf: drl hdq drs nbnd drs hfq drl       ! defocusing-corner-focusing
dcbf: drl hdq drs nbnd drs badhfq drl   ! defocusing-corner-bad focusing
cell:fix(dcf)

! Single rotated bad cells with and without kickers
bfdc: badhfq drl drl hdq drs nbnd drs   ! bad focusing-defocusing-corner
bfdck: badhfq drl drl hdq vcorr1 drks nbnd drks hcorr1

! Single rotated good cells with and without kickers
! The good cells are numbered 2, 3, and 4.  The horizontal
! kicker that preceeds each horizontal kicker has the number one less
! (mod 4), e.g., the kicker for the horizontal quad in cell 4 is hcorr3.
fdc: hfq drl drl hdq drs nbnd drs       ! focusing-defocusing-corner
fdck2: hfq drl drl hdq vcorr2 drks nbnd drks hcorr2
fdck3: hfq drl drl hdq vcorr3 drks nbnd drks hcorr3
fdck4: hfq drl drl hdq vcorr4 drks nbnd drks hcorr4
```

```
! Maps on which to check adjustments (see table B.4)
adj1:des(hcorr3 fdck4)
adj2:fix(hcorr3 fdck4 bfdck)
```

Table B.4 Sample Run of Fitting and Correction

```
ML3.1> ! Demonstration of fitting and orbit correction in Marylie 3.1
ML3.1> ! Example is a four-sided ring composed of FODO cells.
ML3.1> ! First, set the tunes to the desired values of .19 and .185.
ML3.1> <fourside
ML3.1> pr cell[tune]
cell[tune x]=0.1975049037232380
cell[tune y]=0.1866276261612764
cell[tune t]=0.0000000000000000E+00
ML3.1> targx=.19
ML3.1> targy=.185
ML3.1> fit cell[tune x]-targx,cell[tune y]-targy : kl?
ML3.1> pr cell[tune]
cell[tune x]=0.1899999999999998
cell[tune y]=0.1850000000000001
cell[tune t]=1.1857967309049915E-09
ML3.1> pr kl?
klf=0.2031540176634698
kld=-1.455819033088247

ML3.1> ! Begin correction for single misaligned horizontal quad in
ML3.1> !  cell 1.
ML3.1> ! Fit the horizontal kicker at horizontal quad previous to the
ML3.1> ! one with the misalignment so that the beam
ML3.1> ! goes through the center of the misaligned quad.
ML3.1> fit adj1[px]+0.01 : strh3
ML3.1> pr adj1[?]
adj1[x]           =-2.1407162629898209E-04
adj1[px]          =-1.0000000000000000E-02
adj1[pt]          =-8.7344303359367659E-03
ML3.1> pr strh?
strh1=0.0000000000000000E+00
strh2=0.0000000000000000E+00
strh3=-2.1856600497176117E-02
strh4=0.0000000000000000E+00

ML3.1> ! Now adjust the strengths of the kickers at the horizontal quad
ML3.1> !  with the misalignment and at the next horizontal quad so that
ML3.1> ! beam returns to the design trajectory at the latter horizontal
ML3.1> ! quad.
ML3.1> fit adj2[x],adj2[px] : strh4,strh1
ML3.1> pr adj2[?]
adj2[x]           =3.7177677481727441E-21
adj2[px]          =-1.9176431230311452E-19
ML3.1> pr strh?
strh1=-2.1644003957606226E-02
strh2=0.0000000000000000E+00
strh3=-2.1856600497176117E-02
strh4=1.9915047851789408E-02
```

b. Implementation and Testing

Adapting MARYLIE to include the first-order transformation mechanism involves essentially two modifications: the tracking routines, and the concatentation routines.

i. Tracking and Concatenation

Tracking is implemented in a straightforward manner by the scheme described in Chapter 2. The initial coordinates v_o are offset by the amount $[f_1,\zeta]|_{v_o}$, and then ordinary tracking proceeds from there. This involves minor modification to the MARYLIE 3.0 routines for non-symplectic and symplectic tracking.

Testing merely involves checking that the initial conditions are shifted by the amount expected based on the value of f_1. The higher-order portions of the map may be set to the identity to facilitate checking.

Concatenation is more difficult, implementing the process described in Chapter 4. The routines for concatenating matrices and polynomials third-order and higher, as explained in Section 4b, already exist in MARYLIE 3.0. The routines principally responsible for this are CONCAT and XFORM, the former for concatenating the polynomials and multiplying the matrices, the latter for transforming a polynomial by a matrix. To these must be added a facility for concatenating when a first-order term is present. A special routine GlMOVE, called by CONCAT, performs the computation needed to move the first-order term to the left (4.72), or optionally to move the first-order term to the right (4.82). This routine in turn calls various utilities, including MATIFY which turns

the calculated second-order polynomial h_2 into a matrix JS according to the equation (1.99), and EXPM which symplectically exponentiates this matrix, by any of the three methods described in Chapter 5. It was found after coding all three methods that, for the same degree of accuracy, the first two methods ran at approximately the same speed but the Cayley method ran approximately twice as fast. Therefore, this method was used. GIMOVE also uses two Poisson bracket routines, FIPBKT and PBKT.

One easy test of the values of h_n in the concatenation (4.72) is afforded by applying the result to zero on both sides. First, we simplify the problem by saying $f_1 = g_2 = g_3 = g_4 = 0$ so

$$\underbrace{e^{:f_2:} e^{:f_3:} e^{:f_4:}}_{M_f} e^{:g_1:} \cdot|_0 = e^{:h_1:} e^{:h_2:} e^{:h_3:} e^{:h_4:} \cdot|_0 \qquad (B.4)$$

On the left side, the application of the higher-order part M_f to the initial condition 0 yields 0. Then all that remains is

$$v_a = e^{:g_1:} \cdot|_0 = [g_1, \zeta] \qquad (B.5)$$

For example, if $g_1 = .1\ P_X$, then the left side is $[.1\ P_X, \zeta] = (-.1,0,0,0,0,0)$. On the right side, track in the fashion described above, applying $e^{:h_1:}$ first then using the usual tracking method for higher order terms. We should get the same result for both sides to order ε^3.

Additionally, we may apply both sides to $\zeta_b = e^{-:h_1:} \cdot|_0 = [-h_1, \cdot]$. Then, on the right-hand side, the application of $e^{:h_1:}$ to these initial

conditions yields 0, and the application of the higher-order h_n does not change this. Thus, to test we may apply $M_f\zeta_b$ and expect to get $-\zeta_a$. Note that this method does not check the h_2, h_3 or h_4.

Both the above methods were used, and proved the routines correct. In fact, the two sides of the second method are exactly the same, as shown in Section 4d.

Another check on the formulas and coding is to move the g_1 to the left

$$e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:g_1:} = e^{:h_1:}e^{:h_2:}e^{:h_3:}e^{:h_4:} \qquad (B.6)$$

and then move h_1 back to the right

$$e^{:h_1:}e^{:h_2:}e^{:h_3:}e^{:h_4:} = e^{:k_2:}e^{:k_3:}e^{:k_4:}e^{:k_1:} \qquad (B.7)$$

and see how the polynomials k_n compare with f_n and g_1. Considering that the total order must correct through order 4, one would expect that

$$\|g_1 - k_1\| \sim O(\varepsilon^4) \qquad (B.8a)$$

$$\|f_2 - k_2\| \sim O(\varepsilon^3) \qquad (B.8b)$$

$$\|f_3 - k_3\| \sim O(\varepsilon^2) \qquad (B.8c)$$

$$\|f_4 - k_4\| \sim O(\varepsilon). \qquad (B.8d)$$

These were all borne out by scaling checks, although $f_4 - k_4$ is zero because the computation of h_4 and k_n is exact.

An important check of the concatenation routines is to compare the result of tracking a map with an f_1 present with one where it is missing but simulated it with a "kick" of the coordinates. The user-specified subroutines of MARYLIE allow one to alter the coordinates in tracking any way one pleases, so we may add constants in the same way that the application of a first-order map does. If ε and δ are both scaled by a factor s, the difference of the end results of the two runs should scale by a factor s^4. This was indeed the case with the test lattices tried.

Finally, a check may be made on the end-result by translating a parallel-face bending magnet in the direction along its pole faces, the X direction. This should make no difference in the result up through the appropriate total order in matrix and coefficients. This was tested and confirmed to be the case.

ii. The Fixed Point (Closed Orbit) Finder

There are four routines that form the fixed point (closed orbit) finder. CLORB is the main routine implementing the iteration scheme, and is called on the user's instructions during lattice construction in MARYLIE. It replaces the current map with the shift to the closed orbit, and the map around the closed orbit. It calls three routines: SANDWC, which constructs the map "sandwich" MNM^{-1} or $M^{-1}NM$; the logical function TINDEP which determines whether the map is time independent or not; and the subroutine GET4X4 which extracts the transverse part of the matrix that represents $e^{:g_2:}$ in the cases of time independence. CLORB has been written so that the user may dictate that the procedure in the case of time independence or for time dependence be followed, or to let the code determine this automatically.

In practice, the convergence is not quadratic as was demonstrated theoretically in Chapter 6 because $N^{(n+1)}$ is not calculated by acting on $N^{(n)}$ with the $r_1^{(n+1)}$ maps, as in (6.15) but by acting on M with the $f_1^{(n)}$ maps as in (6.10). This introduces some round-off error in the computation, because of the relatively large, fixed first-order component of M, $e^{:g_1:}$, that washes out the quadratic nature of the convergence. Even so, convergence to ten digits precision occurs in ten iterations for the lattice shown (Table B.) but with the horizontally defocusing quadrupoles misplaced by 10 cm horizontally. If $N^{(n+1)}$ is calculated with $N^{(n)}$ and, the convergence is seen to be quadratic. A routine NCLORB (not listed) performs the calculation this way, and the quadratic converage was verified.

To test the closed orbit finder routines, we merely check both sides of (6.7)

$$e^{:g_1:}e^{:g_2:}e^{:g_3:}e^{:g_4:} = e^{-:f_1:}e^{:f_2:}e^{:f_3:}e^{:f_4:}e^{:f_1:} \qquad (6.7)$$

by putting the right side into the standard factorization with the concatenation tools available. The testing proved the routines correct.

iii. Specification of Misalignments and Generation of Maps

There are several routines to implement the misalignment specification and map generation described in Chapters 7, 11, and 12. The routine SHIFID takes Euler angle parameters given at a fiducial point and computes what they should be at the entry or exit polefaces according to the process described in Chapter 11. It uses the routines EUCPR and INVEUC for doing multiplication and inversion respectively in

the Euclidean group. Once the Euclidean group parameters are known at the face, the routine SHIFT computes the map according to the method of Chapter 12. It uses the routines AROT and TPROT1 for the rotations and LATSHF and DRIFT1 for the translations.

iv. Dipoles

The map computation for the correction dipole has been coded in the kicker approximation (10.38). It is called KICKER. It first computes the map of an ideally powered half-parallel face magnet, and concatenates this with the coordinate transformation explained in Chapter 10.

The ideally powered half-parallel face magnets and the general bending magnet have been coded into routines HPF and GBODY (not listed). This mispowered versions, as well as the mispowered version of the normal-entry bending magnet, await coding.

c. Listings of Important Routines

In the following pages are tables giving listings of some of the important routines used in MARYLIE 3.1 for concatenation, fixed point finding, and so on. They are written in Fortran (ANSI X3.9-1978), with the exception of the "include" statements.

Table B.5 is a brief summary of routines not listed that are called by the listed routines. Tables B.6 through B.12 are listings of misalignment concatenation routines described in Section Bb, along with various matrix symplectification and symlecticity checking routines (see Chapter 5). Table B.13 lists the closed orbit finder routine. Tables B.14 through B.18 list the routines for computation of the misaligned maps as described in Chapter 12. Tables B.19 through B.22 describe the misalignment specification and Euclidean group computation described in Chapters 7 and 11.

Table B.5 Routines Not Listed

MINV (M)	: Inverts a symplectic matrix M.
MMULT (A,B,C)	: Matrix multiplication A = B C.
MTMULT (A,B,C)	: Matrix transpose and multiply A = Btranspose C.
CMULT (A,X,B)	: Matrix multipy by constant A = X B.
VCLEAR (F,N)	: Clear (set to 0) the order N part of the polynomial F.
VCADD (F,X,G,N)	: Cumulative addition of polynomial coefficients F = F + XG for scalar X and Nth order part of F and G.
VADD (F,G,H,N)	: Polynomial coefficient addition F = G + H, Nth order part.
VASS (F,G,N)	: Polynomial assignment F = G, copy Nth order part of G into F.
ALLOC (N)	: Allocate a block of data of length N from the common storage 'BLDATA'. Value returned is a pointer to the start.
DEALL (PTR,LEN)	: Deallocate data block.
SVPBKT (F,N,I,G)	: Single variable Poisson bracket G_N = $[F_N,\zeta_I]$.
MAPELT (TYPE, PARMS, KCK)	: Create the map on the map ring stack level specified by KICK of TYPE with parameters PARMS.
LES (X,DIM,M,Y,S)	: Linear equation solver Y = MX. S is scratch area.
SANDWC	: "Sandwich" the two maps on top of the stack, either $B^{-1}AB$ or BAB^{-1}.
MPPUT, MPGET, MPTOWN	: Put map on right stack, get it from stack, place or remove temporary ownership on map to prevent deletion.

Table B.6 Subroutine GlMOVE

```
      subroutine glmove(out,mout, f,mf,g,which)
c  The routine "glmove" moves the exponential of a first order
c  polynomial to the right (which=1):
c       exp(:gl:)exp(:f2:)exp(:f3:)exp(:f4:)exp(:f1:)
c                = exp(:h2:)exp(:h3:)exp(:h4:)exp(:h1:)
c  or to the left (which=-1):
c       exp(:f1:)exp(:f2:)exp(:f3:)exp(:f4:)exp(:g1:)
c                = exp(:h1:)exp(:h2:)exp(:h3:)exp(:h4:)
c  Written by Liam Healy, June 13, 1985.
c
c----Variables----
c  out = array of polynomials returned
c  mout = matrix returned
      double precision out(*),mout(6,6)
c  f, mf = polynomial and matrix of first map
c  g     = polynomial of second map
c  mfinv = inverse of mf
      double precision f(*),mf(6,6),g(*),mfinv(6,6)
c  which, dw = which way to move g1 (see above)
      integer which
      double precision dw
c  hut = h1, untransformed
      double precision hut(6)
c  glf3 = [g1,f3] etcetera
      double precision glf3(27),g1glf3(6),g1f4(83),g1g1f4(27),
     # ggglf4(6),ggfgf(6),ggf3f3(27),f3glf3(83)
c  gt = first order part of g transformed by mf
      double precision gt(6)
c  emh = the symplectic exponentiation of the matrix calculated
      double precision mh(6,6),emh(6,6)
      integer i
c
c----Functions and Subroutines called----
c  flpbkt, pbkt,
c  matify, mmult, mass, minv,expM
c  vass, vadd, vcadd, vclear,
c  xform
c
c----Routine----
      do 100 i=1,6
        if (g(i).ne.0.) goto 120
  100 continue
      call vass(out,f,0)
      call mass(mout,mf)
      return

c--------Calculate gt, gl transformed by the matrix mf----------
  120 continue
      if (which.gt.0) then
        call mass(mfinv,mf)
        call minv(mfinv)
```

```
        call xform(g,1,mfinv,0,gt)
      else
        call vass(gt, g,1)
      endif

c--------Create all the Various Poisson Brackets Needed--------------
c  [g1,f3]
      call flpbkt(g1f3, gt,f,3)
c  [g1,g1,f3]
      call flpbkt(g1g1f3, gt,g1f3,2)
c  [f3,g1,f3]
      call pbkt(f,3,g1f3,2, f3g1f3)
c  [g1,f4]
      call flpbkt(g1f4, gt,f,4)
c  [g1,g1,f4]
      call flpbkt(g1g1f4, gt,g1f4,3)
c  [g1,g1,g1,f4]
      call flpbkt(gg g1f4, gt,g1g1f4,2)
c  [[g1,g1,f3],[g1,f3]]
      call flpbkt(ggfgf, g1g1f3,g1f3,2)
c  [[g1,g1,f3],f3]
      call flpbkt(ggf3f3, g1g1f3,f,3)
      dw=dble(which)

c-------Calculate the matrix part of the factored exponential-------
c  There are two uncombined exponentials representing a linear map
c  The first is simply mf, as supplied.
c  The second is:
      call vclear(out, 2)
      call vcadd(out, dw,g1f3,2)
      call vcadd(out, dw*.25d0,ggf3f3,2)
      call vcadd(out, .5d0,g1g1f4,2)
c  Now we must make this into a matrix 'mh',
c  and multiply the matrices together to get the final one.
      call matify(mh, out)
      call expM(emh, mh)
      call mmult(mout, emh,mf)

c--------Calculate the third-order part of the factored exponential-----
      call vclear(out, 3)
      call vcadd(out, 1.d0,f,3)
      call vcadd(out, dw*.5d0,f3g1f3,3)
      call vcadd(out, dw,g1f4,3)

c--------Fourth-order part is trivial--------
      call vass(out, f,4)

c-------Calculate first-order part of the factored exponential--------
      call vclear(out, 1)
      call vass(hut, gt,1)
      call vcadd(hut, .5d0,g1g1f3,1)
      call vcadd(hut, dw*1.d0/6.d0,gg g1f4,1)
      call vcadd(hut, .25d0,ggfgf,1)
      if (which.lt.0) then
```

```
  call xform(hut,1,mf,0,out)
  call vadd(out, out,f,1)
else
  call vadd(out, hut,f,1)
endif
return
end
```

Table B.7 Subroutine MATIFY

```
      subroutine matify(matrix,f2)
c  Computes the matrix that corresponds to :f2:.
c  It is written in a simple-minded manner to keep execution time short.
c  Written by Liam Healy, May 29, 1985.
c
c----Variables----
c  matrix = matrix supplied
      double precision matrix(6,6)
c  f2 = array of coefficients giving f2 values (others are ignored)
      double precision f2(*)
c
c----Routine----
      matrix(1,1)=-f2(8)
      matrix(1,2)=-2.*f2(13)
      matrix(1,3)=-f2(14)
      matrix(1,4)=-f2(15)
      matrix(1,5)=-f2(16)
      matrix(1,6)=-f2(17)
      matrix(2,1)=2.*f2(7)
      matrix(2,2)=f2(8)
      matrix(2,3)=f2(9)
      matrix(2,4)=f2(10)
      matrix(2,5)=f2(11)
      matrix(2,6)=f2(12)
      matrix(3,1)=-f2(10)
      matrix(3,2)=-f2(15)
      matrix(3,3)=-f2(19)
      matrix(3,4)=-2.*f2(22)
      matrix(3,5)=-f2(23)
      matrix(3,6)=-f2(24)
      matrix(4,1)=f2(9)
      matrix(4,2)=f2(14)
      matrix(4,3)=2*f2(18)
      matrix(4,4)=f2(19)
      matrix(4,5)=f2(20)
      matrix(4,6)=f2(21)
      matrix(5,1)=-f2(12)
      matrix(5,2)=-f2(17)
      matrix(5,3)=-f2(21)
      matrix(5,4)=-f2(24)
      matrix(5,5)=-f2(26)
      matrix(5,6)=-2.*f2(27)
      matrix(6,1)=f2(11)
      matrix(6,2)=f2(16)
      matrix(6,3)=f2(20)
      matrix(6,4)=f2(23)
      matrix(6,5)=2.*f2(25)
      matrix(6,6)=f2(26)
      return
      end
```

Table B.8 Subroutine FlPBKT

```
      subroutine flpbkt(pb, left,right,ords)
c  fl Poisson Bracket.
c  Takes the Poisson bracket of the first order part of 'left' with
c  the order 'ord' part of 'right'. The result is left in pb.
c  If ords<0, then do all orders from 1 up to -ords.
c  Written by Liam Healy, June 13, 1985.
c
c----Variables----
c  pb = Poisson Bracket, returned
      double precision pb(*)
c  left, right = coefficients of monomials to be concatenated
      double precision left(*),right(*)
c  onetrm = result of concatenating 'right' with a single phase
c  space    variable
      double precision onetrm(0:83)
c  ords, ord, ordm = order supplied, order of 'right' to be
c                  concatenated, ord-1
c  psv = phase space variable (1...6)
      integer ords,ord,psv,ordm
c
c----Functions and Subroutines called----
c  svpbkt, vcadd, vclear
c
c----Routine----
      call vclear(pb,sign(abs(ords)-1,ords))
      do 140 ord=max(ords,1),abs(ords)
        ordm=ord-1
        do 100 psv=1,6
          call svpbkt(right,ord,psv,1, onetrm)
          call vcadd(pb, -left(psv),onetrm,ordm)
  100   continue
  140 continue
      return
      end
```

Table B.9 Function TINDEP and Subroutine GET4X4

```
      logical function tindep
c  Is the linear part of the map
c  represented by the matrix time independent, i.e. does it leave
c  energy untouched?
c  Written by Liam Healy, October 20, 1985.

c----Variables----
      include 'map:mappcs.inc'
c  ptrs = pointers to current map
      integer ptrs(NPCS),base,i
      include 'bldata'

c----Routine----
      call mpget(ptrs,0)
      base=ptrs(MATRIX)-1
      tindep= bldata(base+36).eq.1.
      do 100 i=1,5
  100   tindep=tindep.and. bldata(base+6*i).eq.0.
      return
      end



      subroutine get4x4(matout, matin)
c  Collect only the transverse piece of matin into matout.
c  Written by Liam Healy, October 20, 1985.

c----Variables----
c  matout, matin = matrices returned and supplied
      double precision matout(4,4),matin(6,6)
c  row, col = row and column indeces
      integer row,col


c----Routine----
      do 100 row=1,4
      do 100 col=1,4
  100    matout(row,col)=matin(row,col)
      return
      end
```

Table B.10 Routines for Exponentiation of Matrices

```
      subroutine expM (matout, matin)
c  Exponentiate a matrix by the Cayley method.
c  This program written by Liam Healy, June 1, 1985.
c
c----Variables----
c  matin, matout = the matrix to be exponentiated and the result.
      double precision matin(6,6),matout(6,6)
c  matin2, matin3  = square and cube of matin
      double precision matin2(6,6), matin3(6,6)
c  term1, term2 = first and second terms in tanh series
      double precision term1(6,6),term2(6,6)
c  num, den, JWaprx = numerator and denominator of M calc, J*Wapprox
      double precision num(6,6),den(6,6),JWaprx(6,6)
c  ident =identity
      double precision ident(6,6)
      common/id/ident
c
c----Routine----
      call mmult(matin2, matin,matin)
      call mmult(matin3, matin,matin2)
      call cmult(term1, dble(1./2.),matin)
      call cmult(term2, dble(-1./24.),matin3)
      call madd(JWaprx, term1,term2)
      call madd(num, ident,JWaprx)
      call msub(den, ident,JWaprx)
      call mdiv(matout, num,den)
      return
      end



      subroutine expMtaya (matout, matin,lterm)
c  Exponentiates a matrix the conventional way : Taylor series
c  The norm of each term is printed in this routine (as opposed to c
expMtay1).
c  This routine for testing purposes only.
c
      double precision matout(6,6),matin(6,6)
      double precision term(6,6),new(6,6)
      double precision norms(0:4),mxclsu
      double precision fact
c  ident = identity matrix
      double precision ident(6,6)
      common/id/ident
c  lterm = last term in Taylor series.
      integer lterm
c
      fact=1
      call mass(new,ident)
      call mass(matout,ident)
      print *,'Norm of Taylor terms:'
      do 100 i=1,lterm
```

```
        call mmultd(new, new,matin)
        fact=fact*i
        call cmult(term, 1.d0/fact,new)
        norms(mod(i,5))=mxclsu(term)
        if (mod(i,5).eq.0) print 800,(norms(j),j=1,4),norms(0)
  800   format (5g13.6)
        call madd(matout, matout,term)
  100 continue
      if (mod(lterm,5).gt.0) print 800,(norms(j),j=1,mod(lterm,5))
      print *
      return
      end



      subroutine expMtayl (matout, matin,lterm)
c  Exponentiates a matrix the conventional way : Taylor series
c  This routine for testing purposes only.
c
      double precision matout(6,6),matin(6,6)
      double precision term(6,6),new(6,6)
      double precision norms(0:4),mxclsu
      double precision fact
c  ident = identity matrix
      double precision ident(6,6)
      common/id/ident
c  lterm = last term in Taylor series.
      integer lterm
c
      fact=1
      call mass(new,ident)
      call mass(matout,ident)
c     print *,'Norm of Taylor terms:'
      do 100 i=1,lterm
        call mmultd(new, new,matin)
        fact=fact*i
        call cmult(term, 1.d0/fact,new)
c       norms(mod(i,5))=mxclsu(term)
c       if (mod(i,5).eq.0) print 800,(norms(j),j=1,4),norms(0)
c 800   format (5g13.6)
        call madd(matout, matout,term)
  100 continue
c     if (mod(lterm,5).gt.0) print 800,(norms(j),j=1,mod(lterm,5))
c     print *
      return
      end
```

Table B.11 Subroutines for Matrix Symplectification by Furman's Method

```
      subroutine corr(cormat, mat)
c  Gives the correction matrix C of M. Furman's prescription for
c  symplectification.
c  Written by Liam Healy, June 11, 1985.
c
c----Variables----
c  cormat, mat = the correction matrix and the input matrix.
      double precision cormat(6,6), mat(6,6)
c  err = the matrix E
      double precision err(6,6)
c  ident = identity matrix
      double precision ident(6,6)
      common/id/ident
c
c----Routine----
      call symper(err,mat)
      call cmult(err, .5d0,err)
      call msub(cormat, ident,err)
      return
      end



      subroutine iter(m,niter)
c  Iterates to converge on a sympectic matrix by the prescription of
c  M. Furman.
c  Written by Liam Healy, June 11, 1985.
c
c----Variables----
c  m = matrix input and symplectic matrix returned
      double precision m(6,6)
c  cormat= correction matrix C
      double precision cormat(6,6)
c  niter = number of iterations
      integer niter
c
c----Routine----
      do 100 i=1,niter
        call corr(cormat,m)
        call mmultd(m, cormat,m)
  100 continue
      return
      end
```

Table B.12 Routines for Determining the Symplecticity of a Matrix

```
      subroutine symper(err, mat)
c  Gives SYMPlectic ERror: The deviation of a matrix from symplecticity
c  according to the formula of A. Dragt and M. Furman:
c                  E = -1 -N.J.Ntranspose.J
c  where N = mat, the suppplied matrix, E = err
c
c-----Variables----
c  err = error matrix returned
c  mat = matrix to be tested, supplied
      double precision err(6,6),mat(6,6)
c  ntj, nj, term2 = Ntranspose.J, N.J, N.J.Ntranspose
      double precision ntj(6,6),nj(6,6),term2(6,6)
c  jm = matrix J
      integer jm(6,6)
      common/symp/jm
c  ident = identity matrix
      double precision ident(6,6)
      common/id/ident
c
c----Routine----
      call mtmult(ntj, mat,jm)
      call mmult(nj, mat,jm)
      call mmult(term2, nj,ntj)
      call madd(err, ident,term2)
      call cmult(err, -1.d0,err)
      return
      end


      double precision function mxclsu(m)
c  Computes the MaXimum CoLumn SUm norm for the matrix m.
c  Reference: L. Collatz, Functional Analysis & Numerical Mathematics,
c  p.177
c  Written by Liam Healy, June 6, 1985.
c
c ----Variables---
c  m =matrix
      double precision m(6,6)
c  sum = sum norms for the columns
      double precision sum(6)
c
c----Routine----
      mxclsu=0.
      do 100 j=1,6
  100   sum(j)=0.
      do 120 j=1,6
        do 110 i=1,6
  110     sum(j)=sum(j)+abs(m(i,j))
  120   mxclsu=max(mxclsu,sum(j))
      return
      end
```

Table B.13 Subroutine CLORB

```
      subroutine clorb(signal)
c  Closed orbit finder.  Based on the techniques of Chapter 6.
c  Written by Liam Healy, October 1, 1985, rewritten April 4, 1986.

c----Variables-----
      include 'map:mappcs.inc'
c  signal = error condition returned
      integer signal
c  orig = original map as supplied on top of the ring stack
      integer orig(NPCS),fl(NPCS),d(NPCS)
      integer idsubd,rnpll,scrach,isdred,zkick,opt,i
c  dtimin = d is time independent
      logical dtimin
c  identity matrix
      double precision iden(6,6)
      common/id/iden
      external ident
      include 'bldata'

c----Function-----
      integer alloc
      logical tindep

c----Routine----
      zkick=0
      idsubd=alloc(36)
      if (idsubd.le.0) goto 300
      rnpll=alloc(6)
      if (rnpll.le.0) goto 300
      call mptown(1,0)
      call mpget(orig,0)
      call mapelt(ident,0,zkick)
      if (zkick.eq.-1) goto 300
      call mptown(1,0)
      call mpget(fl,0)
      d(MATRIX)=0
      do 100 i=1,8
        if (d(MATRIX).gt.0) call remmap(d)
        call mpput(orig,0)
        opt=2
        call sandwc(opt,-1)
        if (opt.eq.0) goto 300
        dtimin=tindep()
        call mpget(d,0)
        call msub(bldata(idsubd),iden,bldata(d(MATRIX)))
        call mtran(bldata(idsubd))
        if (dtimin) then
          isdred=alloc(16)
          if (isdred.le.0) goto 300
          call get4x4(bldata(isdred),bldata(idsubd))
```

```
      scrach=alloc(20)
      if (scrach.le.0) goto 300
      call les(bldata(rnpll),4,bldata(isdred),
     &              bldata(d(POLYS)),bldata(scrach))
      bldata(rnpll+4)=0.
      bldata(rnpll+5)=0.
      call deall(isdred,16)
      call deall(scrach,20)
    else
      scrach=alloc(42)
      if (scrach.le.0) goto 300
      call les(bldata(rnpll),6,bldata(idsubd),bldata(d(POLYS)),
     &             bldata(scrach))
      call deall(scrach,42)
    endif
    call vadd(bldata(fl(POLYS)),bldata(fl(POLYS)),bldata(rnpll),1)
    call mpput(fl,0)
100 continue
    call vass(bldata(d(POLYS)),bldata(fl(POLYS)),1)
    orig(OWNED)=max(orig(OWNED),0)
    call remmap(orig)
    fl(OWNED)=0
    call remmap(fl)
    call deall(idsubd,36)
    call deall(rnpll,6)
    call mpput(d,0)
    return

300 continue
    signal=-1
    call remmap(orig)
    fl(OWNED)=0
    call remmap(fl)
    call deall(idsubd,36)
    call deall(rnpll,6)
    end
```

Table B.14 Subroutine SHIFT

```
      subroutine shift(parms,kick)
c  Generates the map for a coordinate transformation corrosponding to
c  a translation of coordinates in the X, Y, Z axes (delx, dely, delz),
c  and rigid body rotations with the Euler angles (phi, theta, psi) (see
c  Goldstein, sec 4-4 for definition).
c  Transformations are active motion of the beamline elements with
c  respect to the fixed coordinates; from the particles' point of view,
c  they are passive tranformations of the coordinates attached to the
elements.
c  Thus the routines LATSHF, AROT, TPROT1, DRIFT1
c  must be called so that they describe passive transformation
c  of the coordinates (tprot1 is called with negative of parameter).
c  Written by Liam Healy,  August 21, 1985.
c
c----Variables----
      double precision parms(*)
      integer kick,nokick,maxkck,opt
c  library routines called
      external latshf,drift1,arot,tprot1
c
c----Routine----
      opt=2
      nokick=0
      maxkck=max(1,kick)
      call mapelt(arot,parms(4),maxkck)
      if (maxkck.eq.-1) goto 300
c       The Y-axis rotation is opposite sign, because our rotation is
c       positive by right-hand rule, whereas tprot is positive by left-
c       hand.
      call mapelt(tprot1,-parms(5),nokick)
      if (nokick.eq.-1) goto 300
      call ccmap(opt)
      if (opt.eq.0) goto 300
      call mapelt(arot,parms(6),nokick)
      if (nokick.eq.-1) goto 300
      call ccmap(opt)
      if (opt.eq.0) goto 300
      call mapelt(latshf,parms,nokick)
      if (nokick.eq.-1) goto 300
      call ccmap(opt)
      if (opt.eq.0) goto 300
      call mapelt(drift1,parms(3),nokick)
      if (nokick.eq.-1) goto 300
      call ccmap(opt)
      if (opt.eq.0) goto 300
      if (kick.eq.0) call ptrdrp(nokick)
      if (nokick.eq.-1) goto 300
      return
  300 kick=-1
      return
      end
```

Table B.15 Subroutine TPROT

```
      subroutine tprot(parms,mh,h,flag)
c
c     subroutine to generate lie transformation
c     for trailing edge rotation to pole faces of parallel faced
c     bending magnet without fringe field
c     rho is the magnet design orbit
c     radius in metres, psi the angle between the design
c     orbit and the normal to the pole face
c
      implicit double precision (a-h,o-z)
      double precision parms(*),h(*),mh(6,6)
      include 'glparm.inc'
      integer flag

      goto 100
c
      entry tprot1(parms,mh,h,flag)
      h(1)=sin(parms(1))
 100  continue
c
c  Parameters
c   psi = rotation angle
      psi=parms(1)

      spsi=dsin(psi)
      cpsi=dcos(psi)
      tan=spsi/cpsi
      tan2=tan*tan
      sec=1.0d0/cpsi

c     trailing edge map, in absence of fringe fields
c
c     matrix arrays (containing linear effects)
      do 70 i=3,6
      mh(i,i)=+1.0d0
   70 continue
      mh(2,6)=-sec*spsi/beta
      mh(5,1)=+spsi/beta
      mh(1,1)=cpsi
      mh(2,2)=sec
c
c     arrays containing generators of nonlinearties
c
c     degree 3
c
      h(34)=-tan/2.0d0
      h(43)=-tan/2.0d0
      h(48)=-tan/(gamma**2*beta**2*2.0d0)
c
c     degree 4
c
```

```
h(105)=+tan2/4.d0
h(109)=-tan/(2.d0*beta)
h(114)=+tan2/4.d0
h(119)=+tan2/(4.d0*gamma**2*beta**2)
h(132)=-tan/(2.d0*beta)
h(139)=-tan/(2.d0*gamma**2*beta**3)
return
end
```

Table B.16 Subroutine AROT

```
      subroutine arot(parms,mh,h,flag)

c  Rotates axes in x-y plane by angle 'ang'.  Positive angle rotates by
c  right-hand rule (thumb in Z direction, the direction of beam).
c  This is a passive rotation; particle coordinates are given in terms
c  of new axes.
c  In order to get the map for an element, e.g., a quad,
c  rotated on its axis by theta clockwise looking in the direction
c  of the beam, the element map should be preceded a
c  by arot(theta) and followed by arot(-theta).
c  Written by Liam Healy, June 12, 1984.
      implicit double precision (a-h,o-z)
      double precision h(*),mh(6,6)
      double precision parms(*)
      integer flag

      ang=parms(1)
c  Rotate coordinates
      mh(1,1)=cos(ang)
      mh(1,3)=sin(ang)
      mh(3,1)=-sin(ang)
      mh(3,3)=cos(ang)
c  Rotate momenta
      mh(2,2)=cos(ang)
      mh(2,4)=sin(ang)
      mh(4,2)=-sin(ang)
      mh(4,4)=cos(ang)
c  Don't touch flight time
      mh(5,5)=1.
      mh(6,6)=1.
c  Polynomials are zero (bless those linear maps).
      return
      end
```

Table B.17 Subroutine LATSHF

```
      subroutine latshf(parms,mh,h,flag)
c  Lateral shift map.
c  Liam Healy, March 18, 1986.

c     double precision mh(36),h(*)
c  parms = list of parameters
      double precision parms(*)
c  flag = what derivatives to calculate
      integer flag,i
      double precision ident(36)
      common/id/ident

c----Routine----
      do 100 i=1,36
100     mh(i)=ident(i)
      h(2)=parms(1)
      h(4)=parms(2)
      return
      end
```

Table B.18 Subroutine DRIFT

```
      subroutine drift(parms,mh,h,flag)
c
c
c     generates linear matrix mh and
c     array h containing nonlinearities
c     for the transfer map describing
c     a drift section of length 1 metres
c
      implicit double precision (a-h,o-z)
      double precision 1,lsc,mh
      dimension h(209)
      dimension mh(6,6,0:*)
      include 'glparm.inc'
c   parms = list of parameters
      double precision parms(*)
c  flag = what derivatives to calculate
      integer flag
c
      goto 100
      entry drift1(parms,mh,h,flag)
      h(6)=-parms(1)/(s1*beta)

  100 continue
      l=parms(1)
      lsc=l/s1
c
c
c     add drift terms to mh
c
      do 40 k=1,6
      mh(k,k,0)=+1.0d0
   40 continue
      mh(1,2,0)=+lsc
      mh(3,4,0)=+lsc
      mh(5,6,0)=+(lsc/((gamma**2)*(beta**2)))
c  Derivative of matrix with respect to length 1
      if (flag.gt.1) then
        mh(1,2,1)=1./s1
        mh(3,4,1)=1./s1
        mh(5,6,1)=1./(s1*(gamma*beta)**2)
      endif
c
c     add drift terms to h
c
c
c     degree 3
c
      h(53)=-(lsc/(2.0d0*beta))
      h(76)=-(lsc/(2.0d0*beta))
      h(83)=-(lsc/(2.0d0*(gamma**2)*(beta**3)))
c
c
c     degree 4
c
      h(140)=-lsc/8.0d0
```

```
h(149)=-lsc/4.0d0
h(154)=+(lsc*(1.0d0-(3.0d0/(beta**2))))/4.0d0
h(195)=-lsc/8.0d0
h(200)=+(lsc*(1.0d0-(3.0d0/(beta**2))))/4.0d0
h(209)=+lsc*(1.0d0-(5.0d0/beta**2))/(8.0d0*gamma**2*beta**2)
return
end
```

Table B.19 Subroutine SHIFID

```
      subroutine shifid(mface,misfid,parms,eltyp,face)
c  Get the Euclidean group (3 misplacements + 3 Euler angles) of
c  the face misalignment, given the misalignment at the fiducial
c  point, and the parameters describing the magnet.
c                -1       -1    -1  -1
c     C = A B A     or  C   = A   B   A          (entry face, exit face)
c  The Euler angles are active rotations of the beamline elements,
c  positive according to the right-hand rule.
c  Written by Liam Healy, March 16, 1986.

c----Variables----
c  mface = C = Euclidean group element of face misalignment (output)
c  misfid = B = Euclidean group element of fiducial point misalignment
c  parms = parameter of perfect elment, either:
c            bend radius, bend angle, entry angle, exit angle (bend)
c            length  (straight element)
      double precision mface(6),misfid(6),parms(4),theta
c  eltyp = -1:bend, 1:straight
c  face = 1: entry face, -1:exit face
      integer eltyp,face
c  ftof = A = Euclidean group element of fiducial to face map
      double precision ftof(6),invff(6),interm(6),invmf(6)
      integer i

c----Routine----
      do 100 i=1,6
  100   ftof(i)=0.
      if (eltyp.lt.0) then
        theta=parms(2)/2.
        ftof(1)=-2*parms(1)*sin(theta/2.)**2
        ftof(3)=-face*parms(1)*sin(theta)
        ftof(5)=face*theta
      else
        ftof(3)=-face*parms(1)/2.
      endif
      call inveuc(invff,ftof)
      if (face.gt.0) then
        call eucpr(interm,misfid,invff)
        call eucpr(mface,ftof,interm)
      else
        call inveuc(invmf,misfid)
        call eucpr(interm,invmf,invff)
        call eucpr(mface,ftof,interm)
      endif
      return
      end
```

Table B.20 Subroutine EUCPR

```
      subroutine eucpr(out, in1,in2)
c  Take the product of the Euclidean group (translations + SO(3))
c  elements parameterized by displacements alpha (elements 1,2,3)
c  Euler angles psi, theta, phi (elements 4,5,6) (see Goldstein sec 4-
c  1).
c  Written by Liam Healy, December 5, 1985.

c----Variables----
c  out, in1, in2 = returned parameters, two incoming parameter sets
      double precision out(6),in1(6),in2(6)
c  ph, th, ps = phi, theta, psi Euler angles
      double precision phpr,thpr,pspr,ph1,th1,ps1,ph2,th2,ps2
c  sines and cosines of angles
      double precision csum,ssum,cph1,sph1,cph2,sph2,cth1,sth1,cth2,
     &   sth2,cps1,sps1,cps2,sps2,arg,num,den
c  matrix = rotation matrix in R3
      double precision matrix(3,3)
c  i,j = indeces in R3
      integer i,j
c  hpi = half of pi
      include 'nature'
      double precision HPI
      parameter (HPI=PI/2.)

c----Function----
      double precision gen,a,b,c,nofuzz
      gen(a,b,c) = sin(a)*sin(b)*cos(c) + cos(a)*ssum*sin(c)
     &                  + sin(a)*csum*cos(b)*sin(c)
      nofuzz(a)=sign(1.d0,a)*min(abs(a),1.d0)

c----Routine----
c  Product of SO(3) elements, parameterized by Euler angles
      csum=cos(in1(6)+in2(4))
      ssum=sin(in1(6)+in2(4))
      ph1=in1(4)
      cph1=cos(ph1)
      sph1=sin(ph1)
      th1=in1(5)
      th2=in2(5)
      cth1=cos(th1)
      sth1=sin(th1)
      cth2=cos(th2)
      sth2=sin(th2)
      ps2=in2(6)
      cps2=cos(ps2)
      sps2=sin(ps2)
      arg=cth1*cth2-csum*sth1*sth2
      out(5)=acos(nofuzz(arg))
      if (out(5).ne.0.) then
c         The numerator and denominator of the atan2 are not both 0,
c         because elements 1,3 and 2,3 of the rotation matrix can
```

```
c          be both 0 only if theta=0 (see Dragt notes on Rotation, p.59)
          num=gen(ph1,th1,th2)
          den=gen(ph1+HPI,th1,th2)
          if (num.eq.0..and.den.eq.0.) then
            out(4)=0.
          else
            out(4)=atan2(num,den)
          endif
          num=gen(ps2,th2,th1)
          den=gen(ps2+HPI,th2,th1)
          if (num.eq.0..and.den.eq.0.) then
            out(6)=0.
          else
            out(6)=atan2(num,den)
          endif
        else
c           theta=0 : Rotation matrix is just Rz(phi)
          out(4)=acos( nofuzz(csum*(cph1*cps2*cth1*cth2-sph1*sps2)
     &           -ssum* (cph1*sps2*cth1+sph1*cps2*cth2)
     &           -cph1*cps2*sth1*sth2)  )
          out(6)=0.
        endif
c  The translation part
        do 120 j=1,3
  120     out(j)=in2(j)
        call euler(out, in1,in2(4))
        return
        end
```

Table B.21 Subroutine EULER

```
      subroutine euler(vecout, vecin,angs)
c  Returns the rotation matrix applied to vector for the Euler angles
c  supplied.
c  Translated vector is added to whatever is already in vecout.
c  Written by Liam Healy, December 6, 1985.

c----Variables----
c  vecin, vecout = vector supplied and returned
      double precision vecin(3),vecout(3)
c  matrix = matrix
      double precision matrix(3,3)
c  angs = Euler angles
      double precision angs(3)
c  cph, sph, cth, sth, cps, sps = cosine and sine of the angles
      double precision cph,sph,cth,sth,cps,sps
c  i,j = indeces in R3
      integer i,j

c----Routine----
c  Define trig quantities
      cph=cos(angs(1))
      sph=sin(angs(1))
      cth=cos(angs(2))
      sth=sin(angs(2))
      cps=cos(angs(3))
      sps=sin(angs(3))
c  Set matrix values
      matrix(1,1)=cph*cth*cps-sph*sps
      matrix(2,1)=sph*cth*cps+cph*sps
      matrix(3,1)=-sth*cps
      matrix(1,2)=-cph*cth*sps-sph*cps
      matrix(2,2)=-sph*cth*sps+cph*cps
      matrix(3,2)=sth*sps
      matrix(1,3)=cph*sth
      matrix(2,3)=sph*sth
      matrix(3,3)=cth
      do 100 j=1,3
      do 100 i=1,3
  100   vecout(i)=vecout(i)+matrix(i,j)*vecin(j)
      return
      end
```

Table B.22 Subroutine INVEUC

```
      subroutine inveuc(out,in)
c  Finds the inverse element of the element of the Euclidean group
c  specified by in (translation vector + Euler angles).
c  Written by Liam Healy, December 9, 1985.

c----Variables----
c  out, in = returned parameters, incoming parameters.
c  Group is parameterized by 3 translation vectors + 3 Euler angles.
      double precision out(6),in(6)
c  i = index in R3
      integer i

c----Routine----
      out(4)=-in(6)
      out(5)=-in(5)
      out(6)=-in(4)
      do 120 i=1,3
120     out(i)=0.
      call euler(out, in,out(4))
      do 100 i=1,3
100     out(i)=-out(i)
      return
      end
```

Table B.23 Subroutine KICKER

```
      subroutine kicker(parms,kick)

c    parms = list of parameters
       double precision parms(*)
c   b = field strength, len = length of dipole
c   rho = bend radius
c   theta = bend angle
c   hv = horizontal (0) or vertical (1) flag
       double precision b,len,rho,theta,hv,hpfprm(3),frgprm(2)
       integer kick,kck,nokick,opt

       include 'libr:glparm.inc'
       include 'nature'
       external hpfl,nfrng,arot,drift

c----Routine----
       opt=2
       len=parms(1)
       b=parms(2)
       if (b.eq.0.) then
         call mapelt(drift,parms,kick)
         if (kick.eq.-1) goto 300
       else
         hv=parms(3)
         rho=brho/b
         theta=asin(len/rho)
         nokick=0
         kck=max(1,kick)
         if (nint(hv).ge.1) then
           call mapelt(arot,PI/2,kck)
           kck=0
         endif
         frgprm(1)=rho
         frgprm(2)=1
         call mapelt(nfrng,frgprm,kck)
         if (kck.eq.-1) goto 300
         if (nint(hv).ge.1) then
           call ccmap(opt)
           if (opt.eq.0) goto 300
         endif
         hpfprm(1)=rho
         hpfprm(2)=theta
         hpfprm(3)=1
         call mapelt(hpfl,hpfprm,nokick)
         if (nokick.eq.-1) goto 300
         call ccmap(opt)
         if (opt.eq.0) goto 300
         frgprm(2)=-1
         call mapelt(nfrng,frgprm,nokick)
         if (nokick.eq.-1) goto 300
         call ccmap(opt)
         if (opt.eq.0) goto 300
```

```
      if (nint(hv).ge.1) then
        call mapelt(arot,-PI/2,nokick)
        if (nokick.eq.-1) goto 300
        call ccmap(opt)
      endif
      if (kick.eq.0) call ptrdrp(0)
    endif
    return

300 continue
    if (kick.eq.0) call ptrdrp(0)
    kick=-1
    return
    end
```

Appendix C: The Symbolic Computation Code ANNALIE

In this Appendix I shall briefly describe and give examples for ANNALIE (Analytical Lie Algebraic computations for charged particle beam transport), a code written in the language SMP to do some of the computations of MARYLIE described above analytically. I will assume that the reader is familiar with the SMP language (Inference Corp. [1983]), or can figure it out with examples and an explanation. Portions of this package were used to verify the concatenation formulae of Chapter 4, and to compute the maps in Chapters 9 and 10.

The package of routines called GENL (Table C.1) contains a number of general purpose routines and initialization routines. The function set-up creates the canonical variables, parameters such as the dimension of phase space, and quantities such as the matrix J (1.7). The function 'ndex' computes the index number for an array of exponents, and 'expon' does the reverse (see Appendix D). The other important functions are 'xform' which transforms a polynomial by a matrix by the transformation rule, and 'matify' which turns a polynomial into a matrix according to the rule (1.99). The package SETUP (Table C.2) involves the appropriate initialization and defintion.

MARYLIE, through working numerically is, in effect, doing symbolic computations. Storing each polynomial coefficient as a separate element of an array, for instance, it performs a Possion bracket on these polynomials by multiplying and adding the appropriate array elements. Given a symbolic manipulation program such as SMP, one has a broader range of choices in representing the maps. For example, a polynomial can be stored as an array by monomial number and coefficient the way it

is stored in MARYLIE,

$$[c_{28}, c_{29}, \ldots], \tag{C.1}$$

or as a symbolic polynomial,

$$c_{28}X^3 + c_{29}X^2P_X + \ldots, \tag{C.2}$$

where the index numbers are a particular way of storing the polynomial coefficients (the illustration here is the storage scheme of MARYLIE, the Giorgelli ordering, Appendix D or Dragt et. al. [1985]). For this reason, there are many routines for the representation and conversion of representation of polynomials. These are contained in the package POLYS (Table C.3).

A companion to POLYS is CREATE (Table C.4), which has several functions for making polynomials. Chief among these are 'npol' which makes a symbolic polynomial from a list of indices and 'mkind' which will make a list of indices based on some quality such as 'mps' for midplane symmetry or 'conserve' for time-independence of the Hamiltonian. Finally, 'pick' randomly picks monomials from a list to form a polynomial via 'ranpoly.'
This is so that relations may be tested on sample small polynomials, where the fully general polynomial may be too large for SMP to handle.

The two packages PB and LIE (Tables C.5 and C.6) contain the major functions of ANNALIE. They implement the definitions of Chapter 1. The function 'pb' computes the Poisson bracket axiomatically, by the rules (1.43-1.46). That is, computation proceeds by using linearity and the

derivation rule to break up a polynomial, until only fundamental Poisson brackets (those between phase space variables) remain. Once the fundamental Poisson brackets have been calculated, the final answer may be constructed.

The functions that remain in the package PB are devoted to the Lie operators, or the adjoint of the Poisson bracket. The function 'colon' turns a polynomial into a Lie operator (see section 1c), 'concat' concatenates (composes) then, and 'liepow' and 'lieplus' allow their exponentiation and addition.

The usefulness of these Lie operators is extended by the function 'allow' in the package LIE. This function is a general purpose routine that will take any mathematical function with a Taylor series and alter it so that when it is applied to a Lie operator, it is replaced by its Taylor series with addition replaced with 'lieplus,' multiplication by 'concat', and exponentiation by 'liepow'. In particular, when applied to 'Exp,' the SMP exponentiation function, it forms the Lie transformation (1.54). In practice the Taylor series is truncated at some suitable point, specified by the user as an argument to 'allow.'

There are other functions of ANNALIE not included here, primarily to make it compatible on input and output with MARYLIE. Much of what is part of MARYLIE, however, is not present in ANNALIE. In particular, there is nothing that is associated with purely numerical aspects of MARYLIE, for example, the element library or computations of tunes.

ANNALIE has proved quite useful as an aid to the implementation of the mathematics in this dissertation into MARYLIE. There is much that could be done to extend this usefulness, but from a practical standpoint

it would be limited by the capacity of most computers that SMP runs on, and the bugs in SMP. These considerations have restricted the usefulness of ANNALIE as it is.

Table C.1 Package GENL

(Note: the continuation symbol prints as '_' in these listings.)

```
/************** Utility ***************/

/* Sub list extraction */
subl[$list,$range] :: Cat[Ar[ $range ,$list]]

/* Substitute a list of values for a list of variables in an
expression */
<XList0
sl[$expr,$vbls_=onedeep[$vbls],
  $vals_=onedeep[$vals] & P[Len[$vbls]=Len[$vals]]] :: _
          Ap[S,Prep[$expr,Ldist[Rep[$vbls,$vals]]]]

save[$fun] :: (_$fun[master]:Rel[$fun]; restore[$fun])
restore[$fun] :: $fun:Rel[_$fun[master]]

/*************** General Purpose Routines *************/

/* Create constants based on the number of phase space dimensions
   and the maximum order of polynomials. */
setup[$psd_=Evenp[$psd],$polymaxord_=Natp[$polymaxord]] :: _
          (Lcl[%i,%j,%sh]; _
          Map[Set, 'psd,'polymaxord,'canvbl,'top, _
                'bottom,'id,'zmat,'J,'sm,'expon ]; _
    /*          $psd: Number of phase space dimensions   */_
          psd:$psd; _
    /*   $polymaxord: Maximum order retained in polynomials  */_
          polymaxord:$polymaxord; _
          If[$psd<=6,%sh:subl[cvn, 1,$psd ]; _
            Map[Set,%sh]; canvbl : Rel[%sh], _
            Set[z];canvbl:Map[z, 1..$psd ]]; _
    /* Set property to indicate canonical variables  */_
          Map[_$a[canvbl]:1,canvbl]; _
    /* Minimum, maximum indeces for each order  */_
          top:    Cat[Ar[  1,$polymaxord  ,Comb[$a+$psd,$a]-1]]; _
          bottom: Cat[Ar[  1,$polymaxord  ,Comb[$a+$psd-1,$a-1]]]; _
    /* Useful array & matrices  */_
          zarr: Repl[0,$psd] ; _
          zmat:  Repl[zarr,$psd] ; _
          id: Ar[ $psd,$psd ]; _
          J: Ar[ $psd,$psd ,genJ];
    /* Restore "memoed" functions */
          restore['expon]; restore['mkind]; _
    /* Create symbolic monomial from exponents
    (e.g. x 2 y from  2,0,1,0,0,0 ), and make a table of symbolic
    monomials based indexed by the standard indeces */_
          sm[0]:1; %j: expon[0] :  Repl[0,psd] ; _
          Do[%i,top[$polymaxord],%j:next[%j]; sm[%i]:symmon[%j]]; _
          Null)
```

```
/*  Template to generate matrix J  */
genJ[$i,$j] :: P[Ceil[$i/2] = Ceil[$j/2]]*Sign[$j + -$i]

/*  Index computation  */
obin[$m,$i] :: Comb[$m+psd-$i,1+psd-$i]
ndex[$j] :: (Lcl[cord,ind,ib]; ind:cord:$j[psd]; _
              Do[ib,psd-1,1,-1,Inc[cord,$j[ib]]; _
              Inc[ind,obin[cord,ib]]]; ind)

/*  Find the last non-zero exponent (except psd) in an array of
exponents */
lnzj[$j] :: (Lcl[i] ; i : psd ; Loop[$j[i] = 0 | i = psd,Dec[i],i > 0])

/* Get the next (in the index ordering) array of exponents from
    this one */
next[$1] :: (Lcl[out,i] ; out : $1 ; out[psd] : 0 ; i : lnzj[$1] ; If[i_
              > 0,Dec[out[i]]] ; Inc[out[i + 1],1 + $1[psd]] ; out)

/* The exponents for each index */
<XMSet
expon[$ind] ::: next[expon[$ind + -1]]
expon[0] :  Repl[0,'psd]
save['expon]

/*  Invert a symplectic matrix  */
sminv[$m_=isdbyd[$m]] :: -J.Trans[$m].J

/*  Transform a polynomial by a map (given as an array of psd functions,
    corresponding to what each of the image of each of the variables),
    or a matrix.  */
xform[$poly,$map_=onedeep[$map]] :: Ex[sl[$poly,canvbl,$map]]
xform[$poly,$mat_=isdbyd[$mat]] :: xform[$poly,$mat.canvbl]

/*  Definition: monomial basis element is a product of powers of
      the variables, e.g. x px py 2.  */
/*  Make a matrix for the transformation of monomial basis elements
    out of an array of such transformations  */

/*  Turn a second-order polynomial into a matrix
      6 phase space vbls)  */
matify[$poly] ::
  -$poly[8],-2*$poly[13],-$poly[14],-$poly[15], _
                         -$poly[16],- $poly[17] , _,$poly[11],$poly[12] , _
      2*$poly[7],$poly[8],$poly[9],$poly[10],$poly[11],$poly[12] , _
      -$poly[10],-$poly[15],-$poly[19],-2*$poly[22], _
                         -$poly[23],- $poly[24] , _,$poly[20],$poly[21] , _
      $poly[9],$poly[14],2*$poly[18],$poly[19],$poly[20],$poly[21] , _
      -$poly[12],-$poly[17], $poly[21],-$poly[24], _
                         -$poly[26],-2*$poly[27] , _,$poly[25],$poly[26]
      $poly[11],$poly[16],$poly[20],$poly[23],2*$poly[25],$poly[26]

/*************  Character Determination  ************/

/*  List goes to exactly a depth of one  */
```

```
onedeep[$1]:: Contp[$1] & P[Len[Dim[$1]]=1]

/*  Object is a list of n-long lists that do not contain any lists  */
lof1[$1,$n]:: (Lc1[%dim];
      Fullp[$1,2] & P[(%dim:Dim[$1])[2]=$n] & P[Len[%dim]=2])

/*  Object is a psd by psd matrix, with no lists as elements  */
isdbyd[$m] :: Fullp[$m,2] & P[Dim[$m]= psd,psd ]

/*  Two terms are in the Giorgelli order, or if the same exponents,
    in lexical order    */
sortind[$expr1_= Listp[$expr1],$expr2_= Listp[$expr2]]::
      (Lc1[%o];
      If[(%o:Sign[ndex[exps[$expr2]]-ndex[exps[$expr1]]])=0,
      Ord[$expr1,$expr2],%o])
issortind[$expr1,$expr2] :: Ceil[Theta[sortind[$expr1,$expr2]]]
```

Table C.2 Package SETUP

```
/***************  Initial Values  *****************/
/* Names of the canonical variables */
cvn :  'x,'px,'y,'py,'t,'pt

ntexpf2tay:6      /*  Number of terms in exponential to take for f2s  */

/*  Create dimension and maximum order for polynomials  */
If[ Natp[bottom[1]],setup[6,4]]

/*  Names to give to maps when they are read in  */
mapnames:  'mf,'f,'mg,'g,'mh,'h,'mk,'k,'ml,'l,mj,'j
```

Table C.3 Package POLYS

```
/**********  Representation and Conversion of Polynomials **********/

symmon[$j] :: Inner[Pow,canvbl,$j,Mult]

/* For a particular monomial, give a list consisting of the index and
the
   coefficient */
exps[$mon] :: Map[Expt[$a,$mon],canvbl]
co[$mon,$ex] :: Coef[symmon[$ex],$mon]

/*  Produce an  index, coeff  for a/each term in a polynomial
      or in a list of terms, or return a polynomial from
      a list of  index,coef  pairs.  */
ic_:Ldist
ic_:Tier
ic[$mon_ = Listp[$mon]] :: _
      (Lcl[pwrs]; pwrs:exps[$mon];  ndex[pwrs],co[$mon,pwrs] )
ic[$poly_=P[$poly[0]='Plus]] :: ic[terms[$poly]]
pic[$1_=onedeep[$1]] :: $1[2]*sm[$1[1]]
pic[$1_=lofl[$1,2]] :: poly[Map[pic,$1]]
/* <XContig  */
spic[$1] :: Contig[sepoly[pic[$1]], polymaxord ,0]

/*  Find the order of a term, or orders in list of terms or a polynomial
    (note total order will be taken if totalorder=1)   */
totalorder=0
ordi[$term] :: Ap[Plus,exps[$term]] + totalorder*Expt[e,$term]
ordc[$p] :: Map[ordi,terms[$p]]

/*  The order of any particular phase space variable  */
ordj[$term,$n] :: Expt[canvbl[$n],$term]

/*  A non-repeating list of the orders of the terms in a polynomial  */
orders_:Tier
orders[$p] :: Union[ordc[$p]]

/*  Is the polynomial homogeneous?  */
homog[$p] :: P[Len[orders[$p]]=1]
maxord_:Ldist
maxord[$p] :: Last[orders[$p]]

/* Seperate terms of a polynomial, into a list of monomials.    */
terms_:Tier
terms[$p_=Contp[$p]] ::  $p
terms[$p_= Contp[$p]] :: (Lcl[%p]; %p:Ex[$p];
      If[%p[0]='Plus, Ar[Len[%p],%p], %p , %p ])

/*  Make a polynomial out of a list of monomials  or
    select terms of a certain order from a polynomial or list of
monomials;
    return a polynomial.
    Any particular selection criterion can be picked for $sel,
```

```
    preceeded by a single quote. E.g., so[1,'Le,3] picks out
    all terms less than or equal to third order. */
Le[$n,$m] :: $m>=$n
Lt[$n,$m] :: $m>$n
poly_:Tier
poly[$1_=Contp[$1]] :: Ap[Plus,$1]
poly[$1_= Contp[$1]] :: $1
poly[$p,$ord _= Natp[$ord] | $ord=0] :: poly[$p,Eq,$ord]
<XMask
poly[$p,$sel_=Symbp[$sel],$ords] :: (Lcl[%t]; %t:terms[$p]; _
      poly[Mask[Map[$sel[ordi[$a],$ords],%t],%t]])

/*  Create a list of polynomials by seperating a polynomial or list
      of terms according to the value of $qual for each term.
      indexed by that value.
      Default  quality is order.    */
sepoly_:Ldist
sepoly_:Tier
sepoly[$p,$qual] ::
      (Lcl[%quals,%t]; %t:terms[$p]; %quals:Rel[Map[$qual,%t]];  _
      Ar[  Union[%quals]  ,poly[Mask[Map[$a=$b,%quals],%t]]])
sepoly[$p] :: sepoly[$p,'ordi]
```

Table C.4 Package CREATE

```
/************* Creation of Arrays ************/

/*  make symbolic polynomials (arrayed by order)
      for an array with indeces given in the list $inds.      */
mkpol[$name,$inds_=Contp[$inds]] ::
spic[Trans[ $inds,Ldist[$name[$inds]] ]]

/*  Pick $n elements at random from a list  */
/*  <XMask  */
pick[$list,$n] :: (Lcl[%len]; %len:Len[$list];
      Mask[Map[P[$n/%len>$a],Ar[%len, Rand[]]],$list])

/*  Make a list of indices satisfy a given property $qual
      e.g., mkind['expon[$a,2]=0] yields a list of indices for which
      the there is no px term  */
mkind[$qual] ::: Cat[Ar[ top[polymaxord] ,,Rel[$qual]]]
save['mkind]

/*  Some qualities to act on indices  */
mps[$ind] :: Smp[expon[$ind]][3]=0
conserve[$ind] :: Smp[expon[$ind]][5]=0
both[$ind] :: mps[$ind] & conserve[$ind]

ranpoly[$name,$qual,$n] :: mkpol[$name,pick[mkind[$qual],$n]]
```

Table C.5 Package PB

```
/*  Expression contains only constants (literal or
     declared symbolic)  */
constp[$expr] :: Ap[And,Map[P[_$a[Const]=1],Cont[$expr]]]

/*  Expression contains only constants or canonical variables  */
evalbl[$expr] ::
      Ap[And,Map[P[_$a[Const]=1|_$a['canvbl]=1], Cont[$expr]]]

/*  Non-null arguments  */
nna[$$r] ::  P[$$r=Null]

pb_:Ldist
pb_:Tier

/*  Fundamental Poisson Brackets  */
Inner[Set[cvnum[$a],$b],canvbl,Ar[psd]]
pb: Tier[Ar[  canvbl  , canvbl  ,J[cvnum[$a],cvnum[$b]]]]
pb[$f,$f]:0

/*  Constants  */
pb[$$r,$c_=constp[$c],$$u] : 0

/*  How to interpret quotients  */
pb[$$r,$f/$g,$$s] :: Rel[pb[$$r,'$g -1 $f,$$s]]

/*  Derivation property  */
pb[$$r,$f,$g $$t_=nna[$$t]] ::
      pb[$$r,pb[$f,$$t] $g] + pb[$$r,pb[$f,$g] $$t]
pb[$$r,$f $$t_=nna[$$t],$$u_=nna[$$u]] ::
      pb[$$r,$$t pb[$f,$$u]] + pb[$$r,$f pb[$$t,$$u]]
pb[$f $n,$g] :: $n $f ($n-1) pb[$f,$g]
pb[$f,$g $n] :: $n $g ($n-1) pb[$f,$g]

/*  Linearity  */
pb[$$r,$f+$$t,$$u] :: pb[$$r,$f,$$u] + pb[$$r,$$t,$$u]

/*  Unflatten pb chain if it's calculable.  */
pb[$$r,pb[$f,$$s]] :: pb[$$r,$f,$$s]
pb[$$r,$f_=evalbl[$f],$g_=evalbl[$g]] :: pb[$$r,pb[$f,$g]]
pb[$f_=P[_$f[Mgen] =1]] :: $f

/* Null pb is just argument */
pb[,$f]:: $f
pb[$$r,] :: pb[$$r]
pb[$f,] :: $f

/*  Identity Lie operator  */
iden[$obj] :: $obj

/*  Colonize  */
_colon[Extr,Mult]:concat
colon[$f,$obj] :: 'pb[$f,$obj]
```

```
/*  Is the object a Lie operator?  */
lo[$f] :: P[Len[$f]=1 & Ind[$f,1]=$obj]

/*  Concatenate maps  */
concat_:Tier
concat[$f_=lo[$f],$g_=lo[$g]] :: Rel[Ldist[$f[$g]]]
concat[$f,$g,$$r_=nna[$$r]] :: concat[$f,concat[$g,$$r]]
Sxset[".",concat,4,2]

/*  Raise them to powers  */
/* <XMSet  */
liepow_:Tier
liepow[$f_=lo[$f],0] : iden
liepow[$f_=lo[$f],1] :: $f
liepow[$f_=lo[$f],$n_=Natp[$n]& $n>1] ::: concat[$f,liepow[$f,$n-1]]
save['liepow]
Sxset[". ",liepow,3,1]

/*  Add maps  */
lieplus_:Tier
lieplus[$f_=evalbl[$f],$g_=lo[$g]] :: Rel[$f iden + $g]
lieplus[$f_=lo[$f],$g_=evalbl[$g]] :: Rel[$g iden + $f]
lieplus[$f_=lo[$f],$g_=lo[$g]] :: Rel[$f + $g]
lieplus[$f,$g,$$r_=nna[$$r]] :: lieplus[$f,lieplus[$g,$$r]]
```

Table C.6 Package LIE

```
/*  Allow or disallow use of a function on a Lie operator  */
/*  SMP bug: don't 'disallow' or look at properties of a system function
    once it's been used with a Lie operator (causes crash).  */
allow[$fun,$nterms] :: (Lcl[%dum,%liex];
      %liex:S[As[S[Dis[Ax[Ps[$fun[%dum],%dum,0,$nterms]],Inf],
      'Pow->'liepow,'Mult->'concat,'Plus->'lieplus],Inf],%dum->$c];
      Ap[Setd, '$fun[$c_=lo[$c]],%liex ]; $fun:Rel[Rel[$fun]];
      If[Valp[_$fun],_$fun[hold]:_$fun; _$fun[Ldist]:] )
disallow[$fun] :: (If[Valp[_$fun],_$fun:_$fun[hold]];
      $fun[$c_=lo[$c]]: )
```

Appendix D. Index Numbers for Monomial Coefficients Used by Marylie

Index	Exponents of						Variables
	x	px	y	py	t	pt	
1	1	0	0	0	0	0	x
2	0	1	0	0	0	0	px
3	0	0	1	0	0	0	y
4	0	0	0	1	0	0	py
5	0	0	0	0	1	0	t
6	0	0	0	0	0	1	pt
7	2	0	0	0	0	0	x x
8	1	1	0	0	0	0	x px
9	1	0	1	0	0	0	x y
10	1	0	0	1	0	0	x py
11	1	0	0	0	1	0	x t
12	1	0	0	0	0	1	x pt
13	0	2	0	0	0	0	px px
14	0	1	1	0	0	0	px y
15	0	1	0	1	0	0	px py
16	0	1	0	0	1	0	px t
17	0	1	0	0	0	1	px pt
18	0	0	2	0	0	0	y y
19	0	0	1	1	0	0	y py
20	0	0	1	0	1	0	y t
21	0	0	1	0	0	1	y pt
22	0	0	0	2	0	0	py py
23	0	0	0	1	1	0	py t
24	0	0	0	1	0	1	py pt
25	0	0	0	0	2	0	t t
26	0	0	0	0	1	1	t pt
27	0	0	0	0	0	2	pt pt
28	3	0	0	0	0	0	x x x
29	2	1	0	0	0	0	x x px
30	2	0	1	0	0	0	x x y
31	2	0	0	1	0	0	x x py
32	2	0	0	0	1	0	x x t
33	2	0	0	0	0	1	x x pt
34	1	2	0	0	0	0	x px px
35	1	1	1	0	0	0	x px y
36	1	1	0	1	0	0	x px py
37	1	1	0	0	1	0	x px t
38	1	1	0	0	0	1	x px pt
39	1	0	2	0	0	0	x y y
40	1	0	1	1	0	0	x y py

Index	Exponents of						Variables
	x	px	y	py	t	pt	
41	1	0	1	0	1	0	x y t
42	1	0	1	0	0	1	x y pt
43	1	0	0	2	0	0	x py py
44	1	0	0	1	1	0	x py t
45	1	0	0	1	0	1	x py pt
46	1	0	0	0	2	0	x t t
47	1	0	0	0	1	1	x t pt
48	1	0	0	0	0	2	x pt pt
49	0	3	0	0	0	0	px px px
50	0	2	1	0	0	0	px px y
51	0	2	0	1	0	0	px px py
52	0	2	0	0	1	0	px px t
53	0	2	0	0	0	1	px px pt
54	0	1	2	0	0	0	px y y
55	0	1	1	1	0	0	px y py
56	0	1	1	0	1	0	px y t
57	0	1	1	0	0	1	px y pt
58	0	1	0	2	0	0	px py py
59	0	1	0	1	1	0	px py t
60	0	1	0	1	0	1	px py pt
61	0	1	0	0	2	0	px t t
62	0	1	0	0	1	1	px t pt
63	0	1	0	0	0	2	px pt pt
64	0	0	3	0	0	0	y y y
65	0	0	2	1	0	0	y y py
66	0	0	2	0	1	0	y y t
67	0	0	2	0	0	1	y y pt
68	0	0	1	2	0	0	y py py
69	0	0	1	1	1	0	y py t
70	0	0	1	1	0	1	y py pt
71	0	0	1	0	2	0	y t t
72	0	0	1	0	1	1	y t pt
73	0	0	1	0	0	2	y pt pt
74	0	0	0	3	0	0	py py py
75	0	0	0	2	1	0	py py t
76	0	0	0	2	0	1	py py pt
77	0	0	0	1	2	0	py t t
78	0	0	0	1	1	1	py t pt
79	0	0	0	1	0	2	py pt pt
80	0	0	0	0	3	0	t t t

Index	Exponents of						Variables
	x	px	y	py	t	pt	
81	0	0	0	0	2	1	t t pt
82	0	0	0	0	1	2	t pt pt
83	0	0	0	0	0	3	pt pt pt
84	4	0	0	0	0	0	x x x x
85	3	1	0	0	0	0	x x x px
86	3	0	1	0	0	0	x x x y
87	3	0	0	1	0	0	x x x py
88	3	0	0	0	1	0	x x x t
89	3	0	0	0	0	1	x x x pt
90	2	2	0	0	0	0	x x px px
91	2	1	1	0	0	0	x x px y
92	2	1	0	1	0	0	x x px py
93	2	1	0	0	1	0	x x px t
94	2	1	0	0	0	1	x x px pt
95	2	0	2	0	0	0	x x y y
96	2	0	1	1	0	0	x x y py
97	2	0	1	0	1	0	x x y t
98	2	0	1	0	0	1	x x y pt
99	2	0	0	2	0	0	x x py py
100	2	0	0	1	1	0	x x py t
101	2	0	0	1	0	1	x x py pt
102	2	0	0	0	2	0	x x t t
103	2	0	0	0	1	1	x x t pt
104	2	0	0	0	0	2	x x pt pt
105	1	3	0	0	0	0	x px px px
106	1	2	1	0	0	0	x px px y
107	1	2	0	1	0	0	x px px py
108	1	2	0	0	1	0	x px px t
109	1	2	0	0	0	1	x px px pt
110	1	1	2	0	0	0	x px y y
111	1	1	1	1	0	0	x px y py
112	1	1	1	0	1	0	x px y t
113	1	1	1	0	0	1	x px y pt
114	1	1	0	2	0	0	x px py py
115	1	1	0	1	1	0	x px py t
116	1	1	0	1	0	1	x px py pt
117	1	1	0	0	2	0	x px t t
118	1	1	0	0	1	1	x px t pt
119	1	1	0	0	0	2	x px pt pt
120	1	0	3	0	0	0	x y y y

Index	Exponents of						Variables			
	x	px	y	py	t	pt				
121	1	0	2	1	0	0	x	y	y	py
122	1	0	2	0	1	0	x	y	y	t
123	1	0	2	0	0	1	x	y	y	pt
124	1	0	1	2	0	0	x	y	py	py
125	1	0	1	1	1	0	x	y	py	t
126	1	0	1	1	0	1	x	y	py	pt
127	1	0	1	0	2	0	x	y	t	t
128	1	0	1	0	1	1	x	y	t	pt
129	1	0	1	0	0	2	x	y	pt	pt
130	1	0	0	3	0	0	x	py	py	py
131	1	0	0	2	1	0	x	py	py	t
132	1	0	0	2	0	1	x	py	py	pt
133	1	0	0	1	2	0	x	py	t	t
134	1	0	0	1	1	1	x	py	t	pt
135	1	0	0	1	0	2	x	py	pt	pt
136	1	0	0	0	3	0	x	t	t	t
137	1	0	0	0	2	1	x	t	t	pt
138	1	0	0	0	1	2	x	t	pt	pt
139	1	0	0	0	0	3	x	pt	pt	pt
140	0	4	0	0	0	0	px	px	px	px
141	0	3	1	0	0	0	px	px	px	y
142	0	3	0	1	0	0	px	px	px	py
143	0	3	0	0	1	0	px	px	px	t
144	0	3	0	0	0	1	px	px	px	pt
145	0	2	2	0	0	0	px	px	y	y
146	0	2	1	1	0	0	px	px	y	py
147	0	2	1	0	1	0	px	px	y	t
148	0	2	1	0	0	1	px	px	y	pt
149	0	2	0	2	0	0	px	px	py	py
150	0	2	0	1	1	0	px	px	py	t
151	0	2	0	1	0	1	px	px	py	pt
152	0	2	0	0	2	0	px	px	t	t
153	0	2	0	0	1	1	px	px	t	pt
154	0	2	0	0	0	2	px	px	pt	pt
155	0	1	3	0	0	0	px	y	y	y
156	0	1	2	1	0	0	px	y	y	py
157	0	1	2	0	1	0	px	y	y	t
158	0	1	2	0	0	1	px	y	y	pt
159	0	1	1	2	0	0	px	y	py	py
160	0	1	1	1	1	0	px	y	py	t

Index	Exponents of						Variables
	x	px	y	py	t	pt	
161	0	1	1	1	0	1	px y py pt
162	0	1	1	0	2	0	px y t t
163	0	1	1	0	1	1	px y t pt
164	0	1	1	0	0	2	px y pt pt
165	0	1	0	3	0	0	px py py py
166	0	1	0	2	1	0	px py py t
167	0	1	0	2	0	1	px py py pt
168	0	1	0	1	2	0	px py t t
169	0	1	0	1	1	1	px py t pt
170	0	1	0	1	0	2	px py pt pt
171	0	1	0	0	3	0	px t t t
172	0	1	0	0	2	1	px t t pt
173	0	1	0	0	1	2	px t pt pt
174	0	1	0	0	0	3	px pt pt pt
175	0	0	4	0	0	0	y y y y
176	0	0	3	1	0	0	y y y py
177	0	0	3	0	1	0	y y y t
178	0	0	3	0	0	1	y y y pt
179	0	0	2	2	0	0	y y py py
180	0	0	2	1	1	0	y y py t
181	0	0	2	1	0	1	y y py pt
182	0	0	2	0	2	0	y y t t
183	0	0	2	0	1	1	y y t pt
184	0	0	2	0	0	2	y y pt pt
185	0	0	1	3	0	0	y py py py
186	0	0	1	2	1	0	y py py t
187	0	0	1	2	0	1	y py py pt
188	0	0	1	1	2	0	y py t t
189	0	0	1	1	1	1	y py t pt
190	0	0	1	1	0	2	y py pt pt
191	0	0	1	0	3	0	y t t t
192	0	0	1	0	2	1	y t t pt
193	0	0	1	0	1	2	y t pt pt
194	0	0	1	0	0	3	y pt pt pt
195	0	0	0	4	0	0	py py py py
196	0	0	0	3	1	0	py py py t
197	0	0	0	3	0	1	py py py pt
198	0	0	0	2	2	0	py py t t
199	0	0	0	2	1	1	py py t pt
200	0	0	0	2	0	2	py py pt pt

Index	Exponents of						Variables			
	x	px	y	py	t	pt				
201	0	0	0	1	3	0	py	t	t	t
202	0	0	0	1	2	1	py	t	t	pt
203	0	0	0	1	1	2	py	t	pt	pt
204	0	0	0	1	0	3	py	pt	pt	pt
205	0	0	0	0	4	0	t	t	t	t
206	0	0	0	0	3	1	t	t	t	pt
207	0	0	0	0	2	2	t	t	pt	pt
208	0	0	0	0	1	3	t	pt	pt	pt
209	0	0	0	0	0	4	pt	pt	pt	pt

References

Arnold [1978], V.I.: Mathematical Methods of Classical Mechanics, Springer-Verlag, New York.

Bevington [1969], Philip R.: Data Reduction and Error Analysis for the Physical Sciences, McGraw-Hill, New York.

Chow [1978], Yutze: General Theory of Lie Algebras, Vol. 1, Gordon and Breach, New York.

Dixmier [1977], Jacques: Enveloping Algebras, North Holland, Amsterdam.

Douglas [1982], D.R.: Ph.D. Thesis, University of Maryland, unpublished.

Dragt [1976], A.J., J. Finn: Lie Series and Invariant Functions for Symplectic Maps, J. Math. Phys. 17, 2215.

Dragt [1982], A.J.: Lectures on Nonlinear Orbit Dynamics, in Physics of High Energy Particle Accelerators (Proceedings of the 1981 Fermilab Summer School of High Energy Particle Physics), AIP Conference Proceedings, vol. 87, Am. Inst. Physics, New York.

Dragt [1982b], A.J.: Unpublished notes on fringe field calculations.

Dragt [1983], A.J., E. Forest: Computation of Nonlinear Behavior of Hamiltonian Systems Using Lie Algebraic Methods, J. Math. Phys., 24, 1734.

Dragt [1985], A.J., L.M. Healy, F. Neri, R. Ryne, D. Douglas, E. Forest: Marylie 3.0 Manual.

Dragt [1986], A.J.: Lecture notes on the rotation group, unpublished.

Eadie [1971], W.T., D. Drijard, F.E. James, M. Roos, B. Sodoulet: Statistical Methods in Experimental Physics, North-Holland, Amsterdam.

Forest [1984], E.: Ph.D. Thesis, University of Maryland, unpublished.

Furman [1985], M.: A Simple Method to Symplectify Matrices, SSC-TM-4001, SSC Central Design Group, Berkeley, Cal., unpublished.

Goldstein [1950], H.: Classical Mechanics; Addison Wesley, Reading, MA.

Guignard [1970], G.: Effects des Champs Magnetiques Perturbateurs d'un Synchrotron sur 1'Orbite Fermee et Les Oscillations Betatroniques, Ainsi Que Leur Compensation, CERN 70-24, Geneva, unpublished.

Hausner [1968], M., J. Schwartz: Lie Groups, Lie Algebras, Gordon and Breach, N.Y.

Helgason [1978], S: Differential Geometry, Lie Groups, and Symmetric Spaces; Academic Press, New York.

Inference Corp. [1983]: SMP Reference Manual, Inference Corp.

Jacobson [1962], Nathan: Lie Algebras; Dover, New York.

Myers [1984], Peter D.: SSC Magnet Alignment and Aperture, SSC Note-27, SSC Central Design Group, Berkeley, Cal., unpublished.

Richtmyer [1978], Robert D.: Principles of Advanced Mathematical Physics, Springer-Verlag, Berlin.

Schutz [1980], B.: Geometrical Methods of Mathematical Physics, Cambridge Univ. Press, Cambridge.

Spivak [1970], M.: Differential Geometry; Publish or Perish, Berkeley, CA.