U3
34

Differential Algebraic Methods for Obtaining Approximate Numerical Solutions to the Hamilton-Jacobi Equation

by

Gordon D. Pusch

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Physics

APPROVED:

Paul F. Zweifel

Paul F. Zweifel, Chairman

Marvin Blecher

Marvin Blecher

a. L. Ritter

A. L. Ritter

RFStreater

R. F. Streater

Clayton D. Williams

Clayton D. Williams

C. Thomas Mottershead

C. Thomas Mottershead
(Los Alamos National Laboratory)

March, 1990

Blacksburg, Virginia

c.2

LD
5655
V856
1990
P873
c.2

Differential Algebraic Methods for Obtaining Approximate Numerical Solutions to the Hamilton-Jacobi Equation

by

Gordon D. Pusch

Committee Chairman: Paul F. Zweifel

Physics

(ABSTRACT)

I present two differential-algebraic (DA) methods for approximately solving the Hamilton-Jacobi (HJ) equation. I use the "automatic differentiation" property of DA to convert the nonlinear partial-differential HJ equation into a initial-value problem for a DA-valued first-order ordinary differential equation (ODE), the "HJ/DA equation". The solution of either form of the HJ/DA equation is equivalent to a perturbative expansion of Hamilton's principle function about some reference trajectory (RT) through the system. The HJ/DA method also extracts the equations of motion for the RT itself. Hamilton's principle function generates the canonical tranformation, or mapping, between the initial and final state of every trajectory through the system. Since the map is represented by a generating function, it must automatically be symplectic, even in the presence of round-off error.

The DA-valued ODE produced by either form of HJ/DA is equivalent to a heirarchically-ordered system of real-valued ODEs without "feedback" terms; therefore the heirarchy may be truncated at any (arbitrarily high) order without loss of self consistency. The HJ/DA equation may be numerically integrated using standard algorithms, if all mathematical operations are done in DA. I show that the norm of the DA-valued part of the solution is bounded by linear growth. The generating function may be used to track either particles or the moments of a particle distribution through the system.

In the first method, all information about the perturbative dynamics is contained in the DA-valued generating function. I numerically integrate the HJ/DA equation, with the

identity as the intial generating function. A difficulty with this approach is that not all canonical transformations can be represented by the class of generating functions connected to the identity; one finds that with the required initial conditions, the generating function becomes singular near caustics or foci. One may continue integrating through a caustic by using a Legendre transformation to obtain a new (but equivalent) generating function which is singular near the identity, but nonsingular near the caustic. However the Legendre transformation is a numerically costly procedure, so one would not want to do this often. This approach is therfore not practical for systems producing periodic motions, because one must perform a Legendre transformation four times per period.

The second method avoids the caustic problem by representing only the nonlinear part of the dynamics by a generating function. The linearized dynamics is treated separately via matrix techniques. Since the nonlinear part of the dynamics may always be represented by a near-identity transformation, no problem occurs when passing through caustics.

I sucessfully verify the HJ/DA method by applying it to three problems which can be solved in closed form. Finally, I demonstrate the method's utility by using it to optimize the length of a lithium lens for minimum beam divergence via the moment-tracking technique.

Acknowledgements

It is my pleasure to acknowledge a few of the many people who have contributed their friendship and support to me during the research and writing of this dissertation.

First, I would like to thank my parents, Glenn and Jeanne Pusch, for their love, support, and patience, during the many years I have been in school. Their active encouragement of my childhood curiousity about absolutely *everything* in Heaven and on Earth resulted in a lifelong interest in the physical sciences. I dedicate this dissertation to them.

Next, I would like to thank my advisor, Dr. Paul Zweifel, and the rest of my advisory committee for their support and encouragement on a difficult and ambitious dissertation. I also thank them for their longsuffering patience with the many, many setbacks which occurred during the time my research and dissertation was "two weeks from completion" — for the better part of a year. I thank Christa Thomas for her support and encouragement, as well.

I thank the crew of AT Division, Los Alamos National Laboratory, for their support, hospitality, and friendship during the year I spent working for the optics code developement group of AT-3. The time I spent at LANL represented one of the most stimulating and educational years of my life. The topic of this dissertation was conceived there; indeed, most of what I know about accelerator physics was learned during that year. Special thanks to Ed Heighway for granting me the opportunity to work for AT-3. Thanks also to Tom

Mottershead, Walter Lysenko, and Andy Jason for their supervision and mentorship while I was "coming up to speed". Finally, to all of the above, and to Paul Chanell, Barbara Blind, Dave Wolf, and Mary Jo Mottershead, thanks for many long conversations about virtually everything, and for keeping mealtimes interesting.

I thank my colleague Martin Berz for taking the time to carefully explain differential algebra to me, and working through several examples. His detailed demonstration (despite my obvious skepticism) of how DA turns a "particle pusher" into a transfer matrix extracter is directly responsible for this dissertation; the "aha!" experience in which the HJ/DA method was born occurred less than twenty-four hours after I finally understood what he was talking about. The time I spent working with Martin while he was visiting AT-3 proved to be immensely fruitful and valuable, and I found our conversations to be a constant source of information and inspiration. He is also a good friend.

I thank Alex Dragt of the University of Maryland for putting me in touch with Ed Heighway. I also thank him for alerting me to the caustic problem HJ/DA had, and how to cure it. I thank Etienne Forest, for further clarification on this issue. Finally, I thank Fillipo Neri for many interesting discussions, for teaching me the basics of reducing a map to normal form, and explaining how his symplectic tracking method works.

I thank the Instrumentation and Controls group at CEBAF for their hospitality during my summer internship there, particularly Robert Rossmanith for inviting me. Thanks to David Douglas and Jorg Kewisch, for teaching me all the linear machine theory I'd never needed while I was at Los Alamos, and therefore never learned. I might never have aquired a true feel for beta and dispersion functions without their tutelage. Thanks also to Geoff Krafft, for discussions on space-sharge effects and beam disruption. And again, thanks to all three of the preceding for many long conversations about virtually everything.

I thank the people of Los Alamos Christian Fellowship for their friendship, love and prayer support. Special thanks to Ralph and Diane Dahlstrom, and the Dahlstrom family, and to

Jim and Jane Patterson, for serving as my "surrogate families" while I was a stranger in an (initially) strange land. All are shining examples of Christian hospitality in action. The Dahlstroms accepted me as their house-sitter even though they'd only just met me, knowing only that I shared their Faith, and needed a place to live. They then provided me with a "home away from home" for the first half of my stay in Los Alamos, after AT-3 offered to extend my summer internship to a full year. The Pattersons invited me (and many others) to dinner virtually every Sunday there wasn't a Pot-Luck, and housed me during the period between the end of my lease and my return to Blacksburg. I will remember you all for the rest of my days, with fondness.

I also thank the people of Blacksburg Christian Fellowship for their friendship, love and prayer support, especially those from my House Group. The members of House Group have been a constant source of comfort and support during very trying times. Special thanks to Mark Wyer, Hope Phetteplace, Mark Smith, Grace and Wayne Stinchcomb, and Doug and Kim Walsh for their support and encouragement. Thanks to Marie Hetherington for taking time from her Christmas break to carefully edit the incomplete first draft of this dissertation, and for attempting the hopeless task of teaching me an entire course of technical writing in a single evening, so she could explain to me what I'd done wrong. Her help in getting me started writing was invaluable.

In addition to the Dahlstroms and Pattersons, I would like to thank Dan Solie, Grace and Wayne Stinchcomb, Frank and Shirleigh Marvin, Doug and Kim Walsh, Detlef Ulfers, and Mark Smith for providing housing during the perpetually indeterminant time I was completing my research and dissertation. Also to Sam and Patty Metcalf, for providing housing while I was at CEBAF. I will always remember your hospitality and graciousness toward what amounted to "the guest who wouldn't leave". May Christ's Love and the Holy Spirit continue to fill you always!

I thank Lee Radigan for her friendship and encouragement, for many "pleasantly distracting" conversations, and for always keeping her fingers crossed (which sometimes made it

hard for her to type. . . ;-).

Finally, I would like to thank Jeff Brandenburg, Geoff Knobl, and all the rest of the "Friday Night" and "Singalong" crowds for friendship, folk-songs, and frequent flights of fancy. I'm going to miss the n-part harmony, people . . .

I know I've left a lot of people out, but this is already too long as it is. To all those unsung heroes: though I haven't named you, it doesn't mean I didn't appreciate you!

Take care, and best wishes, always.

Partial support for this research was received under a Los Alamos National Laboratory Institutional Supporting Research and Developement grant, under the auspices of the United States Department of Energy, and under grants NSF DMS-8701050 and DOE DE-FG05-87ER25033 from the Center for Transport Theory and Mathematical Physics. I also thank the Virginia Tech Physics Department and CEBAF for use of their computing facilities.

Contents

List of Figures

List of Tables

Chapter 0

Introduction

Scientists and engineers are often interested in the response of a dynamical system to perturbations from some idealized state. Unfortunately, an analytical solution to such a problem usually can only be obtained if the system in question is *linear*, or may be so approximated; for a nonlinear system, one must frequently resort to numerical methods.

For example, this dissertation is particularly concerned with the response of a beam of charged particles to applied electric and magnetic fields; the dynamics of such systems become nonlinear when the field has sextupole or higher components, or when coulomb interactions between the charged particles are included. If one is only interested in a small number of specific trajectories through a system, then the "brute-force" method of direct numerical integration may suffice. However, if one is interested in tracking hundreds of particles through the system, or one particle for hundreds of orbits, the computational expense of direct numerical integration may become prohibative. Furthermore, the analyst may be far more interested in properties characterizing the *system*, rather than some small set of trajectories passing through it.

Now the propagation of a state through a system may be thought of as being represented by an "operation", or *mapping*, which takes initial states into final states. Since the application of the map to a state may be orders of magnitude "cheaper" than direct numerical integration (because one is spared the expense of computing the intermediate states), it often makes sense to invest one's effort "up front" by looking for some means of numerically representing and computing the map produced by a system. When the perturbation is "small", for example, one could respresent the map by a Taylor-series expansion of the final coordinates in terms of the initial coordinates. In the region where this series is sufficiently accurate, tracking becomes a matter of evaluating a set of polynomials having a few hundred coefficients, as opposed to the several thousand complicated evaluations of the equations of motion which might be required by direct numerical integration.

The subclass of *Hamiltonian* systems pose particular problems, because the evolution-map produced by a Hamiltonian system has a special property known as *symplecticity*. One would like the numerical representation of such a map to respect symplecticity exactly, if possible; otherwise, unphysical behavior will occur. One such exactly symplectic representation is provided by the so-called "mixed generating functions". A Mixed generating function satisfies the Hamilton-Jacobi (HJ) equation associated with each Hamiltonian system.

HJ equations may be formulated for any system governed by a variational principle depending on no higher than first derivatives of the dynamical or "state" variables with respect to some continous parameter; this category contains almost all systems in physics, and a significant fraction of the systems in engineering. Examples from physics include certain problems in *celestial mechanics*, *paraxial light optics*, and *paraxial charged-particle optics.* In engineering and applied mathematics, examples of such variational principles arise in *optimal control theory.* In fact, essentially *any* first-order differential system may be imbedded in a larger Hamiltonian system, either by appending one or more "trivial" auxiliary equations [LN88]., or by using the theory of optimal controls [Kir70, Sag67]. The resulting extended system may sometimes exhibit pathological behavior not seen in the original system, however, so caution must be used [LN88].

This dissertation presents a new method for obtaining approximate numerical solutions to the HJ-equation using the recently developed method of "differential algebra". The resulting generating function provides a perturbative expansion for deviations from a specified "reference trajectory". Since much of my recent experience has been in charged-particle optics, I shall draw most of my examples and terminology from this field. However, the methods I shall describe may be applied to any of the problems mentioned above.

The organization of this dissertation is as follows:

In Chapter 1, I introduce those elements and notations of modern dynamical systems theory needed for this dissertation, with particular emphasis on *Hamiltonian* systems. I then discuss the importance of the *symplectic* condition in Hamiltonian dynamics, and why it should be respected by numerical simulations. I then briefly sketch the relationship between Hamiltonian dynamics and optics.

In Chapter 2, I review current numerical methods for Hamiltonian systems, and discuss their respective advantages and liabilities. I then briefly sketch my *HJ/DA* method, describing where it fits in with the aforementioned schemes.

In Chapter 3, I give a tutorial on *differential algebra*, a new method which allows one to compute the numerical values of the *analytical* derivatives of functions, to arbitrarily high order and machine precision, without resorting to an explicit analytical formula.

In Chapter 4, I introduce those elements of Hamilton-Jacobi and perturbation theory needed for this dissertation. I also briefly discuss the theory of *perturbative eikonals*.

In Chapter 5, I develop and present three forms of my *HJ/DA method*. HJ/DA is a technique for obtaining approximate numerical solutions to the Hamilton-Jacobi equation via differential algebra.

In Chapter 6, I describe an implementation of the HJ/DA method, and verify its accuracy for three test problems solvable in closed form: a particle in a uniform relativistic drift, a two-dimensional harmonic oscillator in polar coordinates, and a relativistic charged particle in a uniform magnetic field. I then apply it to a new problem, for which no closed form solution exists: optimization of a "lithium lens".

In Chapter 7, I summarize my results and conclusions.

Finally, I present my numerical results and the FORTRAN code used in appendices A–G.

Chapter 1

Modern Dynamics, Perturbative Methods, and Optics

"... Among all mathematical disciplines the theory of differential equations is the most important ... It furnishes the explanation of all those elementary manifestations of nature which involve time ..." — Sophus Lie (1895)

This chapter summarizes relevant concepts from modern dynamics and optics, and their connection to perturbative methods, in order to establish terminology, notation, and provide a framework.

1.1 Systems, Flows, and Mappings

A dynamical system may be defined in the abstract as a *tangent vector field* $\boldsymbol{U} \in TM$ on a *manifold of states* M; One may intuitively think of $\boldsymbol{U}$ as a *"velocity"* (see [AMR88, chap. 4], also [Omo86, chap. 2]). TM is the *tangent bundle* over M which, loosely speaking, is the product of M with its tangent manifolds (where tangent vectors live) at every point $p \in M$.

Locally, the equations of motion (EOMs) governing the evolution of the system may always be expressed as a set of first-order ordinary differential equations (ODEs):

$$\dot{\xi}^{\mu} = U^{\mu}(\boldsymbol{\xi}, t). \tag{1.1}$$

Here the ξ^{μ} denote a local set of coordinates on the manifold M, (*i.e.* a chart), t is the evolution parameter, and, as usual, $\dot{\xi}^{\mu}$ denotes the total derivative of the ξ^{μ} with respect to t.

In principle, I can find the evolution of such a system with respect to t from any admissable initial condition $\xi_i(t_1)$ to its coresponding final state $\xi_f(t_2)$; geometrically, this is a ***mapping*** of M onto itself for each t_1, t_2:

$$\mathcal{U}: \quad M \to M; \quad \boldsymbol{\xi}_i(t_1) \mapsto \boldsymbol{\xi}_f(t_2) = \mathcal{U}(t_2, t_1)\, \boldsymbol{\xi}_i(t_1). \tag{1.2}$$

Do not be deceived by the simple appearance of (1.2); in general the evolution-map operator $\mathcal{U}$ will ***not*** be linear!

If $\boldsymbol{U}$ satisfies certain smoothness and uniqueness conditions, then $\mathcal{U}$ will be a ***diffeomorphism*** (i.e., a smooth, one-to-one, onto map having a smooth inverse) [AMR88, p.116].

The two-parameter family of diffeomorphisms $\mathcal{U}_{t_2,t_1}$ produced by the map $\mathcal{U}(t_2, t_1)$, and labeled by the continous parameters t_1 and t_2, is called the ***flow*** of the dynamical vector field $\boldsymbol{U}$ on M (see [AMR88, p.239]). The flow has the "group" properties:

$$\mathcal{U}_{t_3,t_1} = \mathcal{U}_{t_3,t_2} \circ \mathcal{U}_{t_2,t_1}, \quad \mathcal{U}_{t_1,t_2} = \mathcal{U}^{-1}_{t_2,t_1}, \quad \mathcal{U}_{t,t} = \mathcal{E}, \quad \forall\, t,$$

where $\mathcal{E}$ is the identity map.

A system is called ***autonomous*** if its evolution map depends only on the ***difference*** $t_2 - t_1$, so that $\mathcal{U}_{t_2,t_1} = \mathcal{U}_{(t_2-t_1),0}$, $\forall\, t_2, t_1$; this is true if, and only if, $\boldsymbol{U}$ is independent of t. The map and its associated flow therefore effectively reduce to one-parameter families, which I write as $\mathcal{U}(t)$ and $\mathcal{U}_t$, respectively.

By abuse of notation, in the text of this dissertation I will often use the undecorated symbol $\mathcal{U}$ to denote both the flow of $\boldsymbol{U}$ and the mapping operator which generates it, when the difference is clear from context (they are, after all, more or less the same thing).

1.2 Hamiltonian Systems and Poisson Brackets

A very special class of dynamical systems are the *Hamiltonian* systems. A member of this class lives on an even-dimensional manifold P, generally known as a *"phase-space"* [AMR88, pp. 560–583]. By Darboux's theorem [AMR88, p. 562], [Arn88, p. 230], for any Hamiltonian system a local chart $\phi : P \to R^n \times R^n \simeq R^{2n}$ may always be found in which the evolution equations take the *canonical form*:

$$\dot{q}^i = \frac{\partial}{\partial p_i} H(\boldsymbol{q},\boldsymbol{p}), \quad \dot{p}_i = -\frac{\partial}{\partial q^i} H(\boldsymbol{q},\boldsymbol{p}), \quad i = 1,\ldots,n. \tag{1.3}$$

The *generalized coordinates* q^i, and their corresponding *generalized* (or *conjugate*) *momenta* p_i provide a particular local parameterization of the manifold P known as *"canonical coordinates"*, while the Hamiltonian function $H(\boldsymbol{q},\boldsymbol{p}) = H(q^1,\ldots,q^n,p_1,\ldots,p_n)$ specifies the dynamics.

I shall use the same symbol $\mathcal{H}$ to denote both the flow and the evolution mapping operator defined by H. Again, context should be sufficient to resolve this ambiguity. To refer to the system itself, I shall use either $\mathcal{H}$ or H, depending on which is more appropriate.

Strictly speaking, it is not generally possible to cover P by a single chart without encountering some sort of coordinate singularity. However, as a matter of convenience such "defective charts" are often used anyway (*e.g.* spherical polar coordinates), since they provide perfectly servicable representations as long as one remembers that some points are not properly represented. (The equations of motion may become numerically ill-conditioned near the singularities, however, requiring some type of regularization.)

The *Poisson Bracket* (PB) between functions on P plays a fundamental role in the theory of Hamiltonian systems. The PB is a skew-symmetric bilinear map:

$$\{\cdot,\cdot\}: \quad C^n(P) \times C^n(P) \to C^{n-1}(P)$$

where $C^n(P)$ is the set of all continuous, n-times differentiable functions over P.

The PB has a number of important algebraic properties [SM74, p. 39] which I will state here without proof:

I.	**Bilinearity:**	$\{(\alpha f_1 + \beta f_2), g\} = \alpha\{f_1, g\} + \beta\{f_2, g\}$
		$\{f, (\alpha g_1 + \beta g_2)\} = \alpha\{f, g_1\} + \beta\{f, g_2\}$
II.	**Anti-Symmetry:**	$\{f, g\} = -\{g, f\}$
III.	**Jacobi's Identity:**	$\{\{f, g\}, h\} + \{\{g, h\}, f\} + \{\{h, f\}, g\} \equiv 0$
IV.	**Derivation Property:**	$\{f, gh\} = \{f, g\}h + g\{f, h\}$.

In terms of canonical coordinates, the PB is given by [Gol80]:

$$\{f, g\} := \sum_{i=1}^{n} \{\frac{\partial f}{\partial q^i}\frac{\partial g}{\partial p_i} - \frac{\partial f}{\partial p_i}\frac{\partial g}{\partial q^i}\}. \tag{1.4}$$

While the above definition appears to be tied to a particular coordinate system, it is actually invariant under the class of coordinate transformations (called *canonical transformations*) which leave the canonical equations (1.3) form-invariant.

Using the Poisson Bracket, I can write the canonical equations (1.3) in a more symmetrical form:

$$\dot{q}^i = \{q^i, H\}, \qquad \dot{p}_i = \{p_i, H\}, \tag{1.5}$$

and indeed for *any* $f(\boldsymbol{q}, \boldsymbol{p}; t) \in C^1(P)$, one can show that the total derivative is given by:

$$\frac{d}{dt}f = \{f, H\} + \frac{\partial}{\partial t}f \tag{1.6}$$

along every trajectory $(q^i(t),\ p_i(t))$ satisfying the canonical equations (1.3).

1.3 Poisson Brackets and Poisson Manifolds

From the above properties of PBs, one sees that for functions in $C^\infty(P)$, the PB satisfies the defining properties of an (infinite-dimesional) Lie-algebra. These properties hold independently of the canonical coordinate representation (1.4), and indeed, a completely coordinate-free treatment is possible using Lie derivatives and differential forms [AMR88, Arn88]. Modern workers in Mechanics hold that it is the above "Lie properties" which are crucial; as in General Relativity, the coordinatization of the system is physically irrelevent, and a coordinate-free treatment should be employed whenever possible.

The arena of Hamiltonian dynamics is a *Poisson manifold*: a manifold of states with a Poisson Bracket defined on it [AMR88, pp.110]. The bracket structure plays a defining role in the geometry of this manifold analogous to the role of the metric tensor in a Riemannian manifold. In General Relativity, we distill the physical essence of coordinates into the metric tensor; in Hamiltonian dynamics, the physics of the canonical equations resides in the Poisson bracket structure. In General Relativity, we attach special importance to those transformations which preserve the metric; in Hamiltonian dynamics, we attach similar importance to the canonical tranformations which leave the bracket structure invariant.

For the purposes of this dissertation, the full machinery of coordinate-independent mechanics will not be necessary. However some of the concepts of *tensor analysis on phase-space* [SM74] will be useful.

1.4 Phase-Space Tensors

Consider a set of $2n$ functions $\xi^\mu(q,p)$, $\mu = 1, \ldots, 2n$. If they are smooth and invertable, I may just as easily parameterize the phase-space P by the ξ^μ as by the (q^i, p_i). In particular, the Poisson bracket (1.4) of two functions so parameterized will become:

$$\{f,g\} = \sum_{\mu,\nu} \frac{\partial f}{\partial \xi^\mu}\{\xi^\mu, \xi^\nu\}\frac{\partial g}{\partial \xi^\nu} = \partial_\mu f \; J^{\mu\nu} \; \partial_\nu g. \tag{1.7}$$

Here I have introduced the Einstein summation convention, and the common notation $\partial_\mu := \partial/\partial\xi^\mu$; I have also introduced the object $J^{\mu\nu} := \{\xi^\mu, \xi^\nu\}$. One can easily show that under a coordinate transformation $J^{\mu\nu}$ must transform like a rank-2 contravariant tensor.

The EOMs of the system in terms of the new paramertization are obtained by computing the total derivative of the ξ^μ; From (1.6) and (1.7), one finds:

$$\dot{\xi}^\mu = \{\xi^\mu, H\} + \frac{\partial}{\partial t}\xi^\mu = \frac{\partial \xi^\mu}{\partial \xi^\lambda} J^{\lambda\nu}\partial_\nu H = \delta^\mu_\lambda J^{\lambda\nu}\partial_\nu H = J^{\mu\nu}\partial_\nu H \tag{1.8}$$

(assuming, of course, that the ξ^μ have no explicit time-dependence). In canonical coordinates, with the particular labeling $\xi^\mu = (q^i, p_i)$, $J^{\mu\nu}$ takes on the following special "***block anti-symmetric unit***" form:

$$J^{\mu\nu} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{-1} & \mathbf{0} \end{bmatrix}. \tag{1.9}$$

This form sugests that canonical coordinates on a Poisson manifold play a role analogous to orthogonal cartesian coordinates in euclidian geometry, with $J^{\mu\nu}$ being somewhat analogous to the metric. The analogy may be somewhat strained though, since the metric is a symmetric bilinear form which maps a pair of vectors to a scalar, while the PB maps a pair of functions to another function. But *any* analogy falls over if you push it hard enough.

1.5 The Importance of Being Symplectic

Canonical transformations are those transformations $\boldsymbol{\xi} \to \bar{\boldsymbol{\xi}}$ which preserve the form of the canonical equations (1.3). This is equivalent to requiring that $J^{\mu\nu}$ be ***invariant***:

$$\bar{J}^{\alpha\beta} = \{\bar{\xi}^\alpha, \bar{\xi}^\beta\} = \frac{\partial\bar{\xi}^\alpha}{\partial\xi^\mu} J^{\mu\nu} \frac{\partial\bar{\xi}^\beta}{\partial\xi^\nu} \equiv J^{\alpha\beta}. \tag{1.10}$$

This apparently innocent identity actually has very far-reaching implications for the would-be numerical analyst intent on tackling a Hamiltonian system. It is called the ***symplectic condition***, because it implies that when $J^{\mu\nu}$ is in the "standard form' (1.9), then the Jacobian matrix of a canonical transformation, $[\mathrm{M}]^\alpha_\mu := \partial\bar{\xi}^\alpha/\partial\xi^\mu$, must be an element of the ***symplectic group*** $Sp(2n)$, where $2n = \dim\{P\}$. The symplectic group $Sp(2n)$ is the set of all $2n \times 2n$ real matrices which satisfy the condition, $\mathbf{MJM}^{\mathrm{T}} = \mathbf{J}$. It is easy to show [Gol80, p.403] that $\det\{\mathbf{M}\} = 1$, so $\mathbf{M}$ is volume-preserving; an immediate consequence of this is *Liouville's theorem*: the volume of any closed region of phase-space is preserved as it is convected along by the Hamiltonian flow [Gol80, §9.8].

If the tensor $J^{\mu\nu}$ is nondegenerate, its inverse defines a closed, nondegenerate two-form $\omega := \frac{1}{2} J^{-1}_{\mu\nu}\, d\xi^{\mu} \wedge d\xi^{\nu}$ called a ***symplectic structure***, and restricts our Poisson manifold M to a *Symplectic Manifold* (see [Arn88, p.201]). The term "symplectic" appears to have been coined by Herman Weyl. The symplectic groups are closely related to ***complex projective geometry*** [Arn88, App. 3], and $\boldsymbol{\omega}$ provides an "almost complex structure" on M. To eliminate possible confusion, Weyl transmogrified the Latin roots "*com*" and "*plex*" to their Greek equivalents "*sym*" and "*plectic*".[1] The symplectic condition (1.10), is an extrordinarily restrictive condition on the geometry and dynamics of a Hamiltonian system. It represents a set of ***global nonlinear partial differential constraints*** on ***every*** possible transformation one might apply to H, including those which represent its own evolution. In fact, the constraints (1.10) are so restrictive that they will probably not be satisfied if one attempts to approximate the flow $\mathcal{H}$ of the canonical equations (1.3) with some simpler system, unless the flow of that simpler system is ***also*** symplectic.

The symplectic condition (1.10) is a fundamental consequence of the Poisson bracket structure on the phase-space of the Hamiltonian system $\mathcal{H}$. It resembles in many ways the condition of ***analyticity*** in complex function theory, although the symplectic condition is understood far less clearly. If anything the symplectic condition may be ***more*** restrictive than complex-analyticity, since in Hamiltonian mechanics one is usually working in a space of more than two dimensions. Just as a function cannot be only "a little bit analytic" over some domain, a tranformation on P cannot be only "a little bit symplectic": it either is or it isn't. Moreover, the symplectic condition is a ***global*** constraint; if the Hamiltonian

[1] The only prior use of the word "symplectic" in English is as the name of a small bone in the head of a fish; amusingly, "***poisson***" happens to be French for "fish".

character of $\mathcal{H}$ is to be preseved, then the symplectic condition (1.10) must be satisfied at *every* point in P, and not just over a limited region.

Symplecticity will also impose an *infinite hierarchy of constraints* on certain approximation schemes (such as finite-difference schemes, or so-called perturbation "theory"), because even if the symplectic condition is *explicitly* satisfied *through* a given order, there is no guarantee that it will not be *implicitly* violated by *higher-order-terms* (abbreviated as "{HOTs}"). If the approximation scheme does *not* satisfy the symplectic condition, it may cause violations of important *conservation laws*, such as *Liouville's theorem* or the *Poincaré invariants* (both of which are consequences of the symplectic structure on P).[2] In particular, replacement of (1.3) by a finite difference scheme on a computer will almost certainly violate the symplectic condition, because the *global* structure of the finite *difference* equations may be very different from the orginal *differential* equations, even in the limit of *vanishing stepsize* (a fact often overlooked in numerical simulations).

Cases have been found [How74] where the structure and character of global features, such as fixed-points and separatrices, change markedly when the system is approximated by a finite-difference scheme, such as the Runge-Kutta or Adams families of integrators [HNW87]: stable orbits may become unstable, fixed points may become attractors or repellors, and closed regions bounded by separatrices may become open. Also, Channell and Scoval [ChSc88] found that with Runge-Kutta and Adams numerical integrators, the constants of motion of the original system typically drift away from their initial values, with the deviation growing like some power of time. In contrast, Channell and Scoval found that, when using their special integrators which preserve symplecticity *exactly* even though they are only of finite accuracy in the time-step, the constants of motion typically show only small, apparently bounded fluctuations about some mean value; in some cases, the constants were even preserved to machine accuracy! The accuracy with which the constants were preserved becomes even more remarkable when one realizes that no *explicit* effort had been made by Chan-

[2] An example of such unphysical behavior may be seen in [Ser85], where gross violations of Liouville's theorem were observed in a long-term tracking study.

nell and Scoval to incorporate conservation of the constants into the difference scheme; the remarkable stability properties of these algorithms apparently came "for free".

As a physicist, I am biased toward approximation schemes which respect the structure of a system and its conserved quantities "as much as possible". The work of Channell and Scoval appears to vindicate this prejudice; it also shows that explicitly respecting symplecticity may perhaps be even *more* important than respecting, say, energy-conservation.

1.6 Optics and Hamiltonian Dynamics

One important application of Hamiltonian Dynamics is *optics*, both geometrical and corpuscular. Geometrical optics deals with the manipulation of light-beams using configurations of curved refractive surfaces. Corpuscular, or *charged-particle*, optics deals with the manipulation of charged particle beams using configurations of electric and magnetic fields. While superficially similar, the character and geometry of these two problems is fundamentally very different. In the former, the geometry is essentially ***Riemannian***, and the underlying variational principle is ***Fermat's principle of "least time"***; the "metric" is the Euclidian metric, conformally rescaled by the reciprocal of the refractive index. In the latter, it is intrinsicly ***Finslerian***, and the variational principle is ***Hamilton's principle of "least action"***. The significant difference between Reimannian and the more general Finslerian geometries is that in Finslerian, or "Hamiltonian" geometry as Synge [Syn60] prefers to call it, the analog of the "metric" (Hessian of the Lagrangian) in general depends nontrivially on the velocity. However Hamiltonian Dynamics provides a common and natural framework for both types of optics.

1.7 Hamiltonian Dynamics and Perturbation Methods

The fundamental problem faced by the numerical analyst in working with nonlinear systems is one of finding a convenient, yet accurate, method of representing the dynamics. One is then faced with a problem: while the *analyst* may understand the concept of functions and mappings, the *computer* certainly does not. All the computer "understands" are finite-precision numbers. So how does one *numerically* characterize a map?

First, practical considerations force one to restrict the scope of the problem. The computer can hold only a finite number of parameters, while the map $\mathcal{U}$ may very well depend on an infinite number. Hence, one needs some sort of approximation scheme. Furthermore, given that one is often most concerned about the action of the map on a *small neighborhood* about some initial reference point ξ_{0i}, one is often content to ask, "Can I *approximately* describe where the points in this neighborhood will be mapped to by $\mathcal{U}$, using only a finite number of parameters?" While many more than "four and twenty ways" to formulate this question exist, Stephen Omohundro has shown that "perturbation theory" provides a particularly convenient and useful framework [Omo86].

Consider a smooth, one parameter map $\mathcal{U}(t)$ of some manifold M onto itself, and reducing to the identity at $t = 0$. $\mathcal{U}(t)$ represents a flow $\mathcal{U}_t$ on M; this flow is the integral of some smooth vector field $\boldsymbol{U}$ defining the dynamical equations of the map, which may be obtained by differentiating $\mathcal{U}$ if they are not already known.

Choose some initial reference point ξ_{0i} in M. I shall call the curve $\xi_0(t) = \mathcal{U}(t)\,\xi_{0i}$ traced out by the image of the reference point ξ_{0i} under $\mathcal{U}(t)$ the *reference trajectory*, or RT. I shal call the image of ξ_{0i} at $t = t_f$, ξ_{0f} . Assume the RT remains in a "regular" region of the flow, i.e., it does not pass too close to a separatix, etc., so that near the RT, the flow is well-behaved, not chaotic. Then $\mathcal{U}(t_f)$ will map a small neighborhood Ξ_i about ξ_{0i} onto a corresponding small neighborhood Ξ_f about ξ_{0f}. The "smallness" of the neighborhood may be represented by the following *ansatz*: we formally introduce an artificial "ordering parameter" ϵ, and assume that the form for a general trajectory may be written as an asymptotic series (though in rare cases it may actually prove to be convergent):

$$\xi^\mu(t) = \xi^\mu_0(t) + \epsilon\xi^\mu_1(t) + \frac{1}{2!}\epsilon^2\xi^\mu_2(t) + \frac{1}{3!}\epsilon^3\xi^\mu_3(t) + \dots \tag{1.11}$$

Here $\xi^\mu_0(t)$ is again the RT, and the "corrections" ξ^μ_i, $i = 1, 2, 3, \dots$ are assumed independent of ϵ, although they may depend on other parameters (such as the boundary conditions) which I have suppressed. Hence, in the limit of vanishing ϵ, all deviations from the RT must vanish; *i.e.* the trajectories all collapse onto the RT.

Now substitute (1.11) into the dynamical vector field $\boldsymbol{U}$; if desired, $\boldsymbol{U}$ may also be allowed to have an explicit ϵ-dependence to represent any other dynamical effects which are thought of as "small" (or "large", if there are some "fast" variables we want to "average out"). By formally expanding the result in powers of ϵ and equating like terms, one now has an infinite hierarchy of equations having the same form as a typical problem in "perturbation theory":

$$\begin{aligned}
\dot{\xi}^\mu_0 &= U^\mu(0, \boldsymbol{\xi}_0) \\
\dot{\xi}^\mu_1 &= \xi^\alpha_1 \frac{\partial}{\partial \xi^\alpha_0} U^\mu(0, \boldsymbol{\xi}_0) + \frac{\partial}{\partial \epsilon} U^\mu(\epsilon, \boldsymbol{\xi}_0)\bigg|_{\epsilon=0} \\
\dot{\xi}^\mu_2 &= \xi^\alpha_2 \frac{\partial}{\partial \xi^\alpha_0} U^\mu(0, \boldsymbol{\xi}_0) + \xi^\alpha_1 \xi^\beta_1 \frac{\partial^2}{\partial \xi^\alpha_0 \partial \xi^\beta_0} U^\mu(0, \boldsymbol{\xi}_0) + 2\xi^\alpha_1 \frac{\partial^2}{\partial \epsilon \partial \xi^\alpha_0} U^\mu(\epsilon, \boldsymbol{\xi}_0)\bigg|_{\epsilon=0} \\
&\quad + \frac{\partial^2}{\partial \epsilon^2} U^\mu(\epsilon, \boldsymbol{\xi}_0)\bigg|_{\epsilon=0} \\
&\vdots
\end{aligned} \tag{1.12}$$

The first-order terms are just the linearized system; the higher-order terms describe the effects of nonlinearities (see Omohundro's dissertation [Omo86, pp.84-89] for the details).

There are several important thing one should note about the perturbation expansion (1.12). First, the n^{th} term in the series does not appear until the n^{th}*-order equation*, so while ξ^μ_n feeds "up", it does not feed "down"; therefore the series may be truncated at any point without changing any lower-order results. Second, the n^{th} term in the series appears *linearly* in the n^{th}-order equation, so an explicit solution is always possible, merely by quadratures.

Third, the right hand side of each equation in the series is "homogenous in order", *i.e.* in the second-order term, $\boldsymbol{U}$ always appears differentiated *twice*, and the sum of the subscripts of the $\boldsymbol{\xi}_n$ is always equal to the order minus the number of ϵ-differentiations. Finally, the initial conditions (ICs) of *all* the $\boldsymbol{\xi}_n$ (except for the reference trajectory $\boldsymbol{\xi}_0$) are at the analyst's disposal, because they are "nonphysical" degrees of freedom, introduced solely for constructing the perturbative solution, and have no effect on the unperturbed problem (again, *all* solutions of (1.12) collapse onto the RT in the limit of vanishing ϵ). In a sense, the IC's one chooses for the $\boldsymbol{\xi}_n$ represent a "gauge-like" degree of freedom and are all in some sense "equivalent", because the $\boldsymbol{\xi}_n$ are *not* characteristics of the system! Instead, the $\boldsymbol{\xi}_n$ describe the *deformation* of a system and its initial conditions resulting from the perturbation.

Omohundro advocates a radical reinterpretation of perturbation expansions. An asymptotic series, such as (1.11) is assumed to be, has a zero radius of convergence. Therefore he argues that one should view the coefficients ξ_n^μ as describing things that are characteristic of the *unperturbed* dynamical system in an *infinitesimal neighborhood about the RT*, rather than of the perturbed system at finite ϵ, as is usually assumed. He may therefore place perturbation "theory" on a *rigorous geometrical footing*, the natural geometric framework being the theory of *jet bundles and prolongations*. (I will briefly discuss jets in §4.4 of this dissertation.) Omohundro's geometric interpretation allows one to understand the physical meaning of the series (1.11) even though in general it does not converge.

1.8 Perturbation Methods and Paraxial Optics

When thinking of "optics", most physicists generally think of its simplest (and most limited) form: *Gaussian* (or *paraxial*) *geometrical optics*. One considers only rays inclined at small angles to the optical axis, and deviating but a small distance (relative to any other length scale) transversely to it. The surfaces of all optical elements are idealized, having the simplest possible geometry: spheres, or perhaps at worst a quadric. All indices of refraction are homogenous, so that rays travel in straight lines between optical surfaces, and optical elements are frequently regarded as "infinitesimally thin" (again, relative to any other length scale). Then one may expand to first order in deviations and angles; *i.e.* one ***linearizes*** the problem to make analytical solution possible. One has a simple problem in plane right triangles.

However, the "real world" is not quite so accommodating. Desirable characteristics of the instrument, such as small physical size and large usable aperture, seldom allow the luxury of "thin" elements. At the same time, practical aspects of manufacturing impose the conflicting demand that the majority of the optical surfaces be spherical. But thick spherical lenses of short focal-length and large aperture fail to satisfy the paraxial condition; nonlinear effects appear, and must somehow be treated.

The successes of linearized optics encourages one to attempt a perturbative approach. One rescales all the transverse deviations and angles, which are considered to be "small", by an artificial "ordering parameter". One then formally expands the dynamics in terms of this parameter, hoping that the coefficients of the perturbation expansion will provide some information about nonlinear effects, even though at most they will provide an asymptotic description of the optical system. This approach proves amazingly successful; one may use the perturbation coefficients both to classify and correct the various distortions, or "aberrations" (coma, astigmatism, distortion, etc.), resulting from the "third-order" terms in the expansion [BrnWlf70]. Prior to the developement of these techniques (largely by German opticians), light optics was largely a matter of "cut and try"; now it is a *science* (but largely a German and Japanese science, to the regret of American camera manufacturers).

Charged-particle optics almost forces one into these techniques from the start. The "lenses" of charged-particle optics consist of carefully shaped electromagnetic fields in free space (free, because a beam of charged particles usually must propagate through an ultrahigh vacuum if it is to be useful), and Maxwell's equations in free space strongly constrain the types of fields one may produce. For example, “fringe fields” will always exist, extending beyond the lens for a distance comparable to its aperture; in essence, the “thickness” of an electric or magnetic lens must be at least as great as its aperture. Furthermore, Schertzer’s theorem [Sch36] and its extension by Moses [Mos66] show that it is physically impossible for a system containing only quadrupole or solenoid lenses to be free of third-order aberrations. Hence, prior to the development of perturbative charged-particle optics, the only option available to, for example, electron microscope designers was to use very small apertures and very large focal lengths, and accept the design penalties this imposed.

Chapter 2

Current Numerical Methods for Nonlinear Dynamics

"... Before beginning any numerical calculation, it is vitally important that you first know the correct answer ... " — Wheeler's First Moral Principle

I now review the currently used methods in charged-particle optics, starting first with non-symplectic methods, followed by their symplectic generalizations. Again, while this dissertation focuses on optical systems, the techniques are applicable to *any* dynamical system.

2.1 Non-Symplectic Methods

Until recently, the two most commonly used methods in charged-particle optics have been *ray-tracing* and *extended transfer matrices.* I will briefly describe the two methods, and enumerate their respective advantages and drawbacks.

2.1.1 Ray-Tracing

Ray-tracing is a fancy name for brute-force numerical integration. The principal advantages of ray-tracing are simplicity and generality. It makes few particular assumptions about the nature of the dynamical system other than that ODEs can describe it. One merely writes down the equations of motion (EOMs) in a convenient form, then discretizes them according to one's favorite method (usually high-order Runge-Kutta or Adams). After selecting the desired initial conditions (ICs), one numerically integrates the EOMs in order to determine the corresponding final conditions, and, if desired, the trajectory connecting them. One then repeats this procedure using various "sufficiently nearby" ICs until the "pencil of rays" generated "adequately" explores the neighborhood surrounding the reference trajectory.

Unfortunately, one generally does not know beforehand just what constitutes "sufficiently nearby", nor be able to quantitatively define what one means by "adequately". One must, with significant computational expense, explore many trajectories deviating by varying amounts from the RT, frequently with little information regarding the importance of accumulated errors.

Under the best of circumstances, ray-tracing leaves one quite literally with reams of numbers: each initial condition and its corresponding final condition. Taken *collectively*, they should in principle provide *some* sort of characterization of the system; however *individually* each ray has little significance.

Various techniques have been developed to aid in extracting useful information from a ray-tracing run:

- **Scatter-Plots.** In this popular method, one generates a set of intial conditions (sometimes regularly, but usually according to some probability distribution via a Monte-Carlo method), propagates this set through the system, and plots various two-dimensional projections of the resulting final states While certain characteristics, such as effective beam spot-size or momentum spread can fairly easily be read off from such plots, the more complicated effects resulting from nonlinearities are much more difficult to quantify. One often merely relies on the unparalleled pattern-recognition ability of the human eye to gain some sort of "feel" for the system. Such an approach is necessarily a rather subjective proceedure, and one's perspective may be severely limited by the low dimensionality of the projection, unless the motions in the various planes are not coupled; however, in the hands of an expert scatter-plots may be quite illuminating.

- **Poincaré Sections.** In this method, one chooses a surface in phase-space (generally a two-dimensional plane), and some set of initial conditions. One then allows the system to evolve; each time the resulting trajectory intersects the surface, one plots a "hit" at the intersection point [Arn88]. (If the surface is two-dimensional and orientable, one may choose to plot only those "hits" which intersect the surface from a given side.) This technique is extensively used in the study of nonlinear oscillators; in optics, it applies only to "multi-pass" systems, such as storage-rings. Poincaré sections are again a visually oriented method, also limited to a low-dimensional surface of section, and may miss important dynamics [ChSc88].

- **Moments.** For a more objective characterization of the system, one may again generate and transport a distribution of particles. One then studies not the scatter-plot, but the effect the system has on the *moments* of the particle distribution with respect to some set of "basis functions", usually polynomials in the coordinate deviations from the RT. (The term "basis" is somewhat of a misnomer here, as the finite set of functions will not be complete.) Three main factors have limited the use of moments:

  ▹ **Interpretation.** With the exception of a "gaussian-ellipsoidal beam", which is completely parameterized by its "zeroth", first, and second moments (normalization, centroid, and "sigma-matrix", respectively), we have no adaquate interpretive framework for assessing the physical significance of moments. Quite simply, taking moments is a "many-to-one" mapping— one can't uniquely recover a distribution given only a finite number of its moments.

▹ **Statistics.** Due to statistical fluctuations, to get an accurate estimate of the moments may require the use of hundreds, perhaps thousands, of particles. Furthermore, the number of particles needed increases rapidly with both the order of the moment and the dimensionality of the system.

▹ **Truncation.** One can derive a set of evolution equations for the moments directly from the EOMs, which the ray-tracing moments ought to satisfy as the number of particles tends to infinity. These equations exhibit a "feed-down" phenomenon analogous to that which occurs in the BBGKY-hierarchy [LLb] of transport theory: the truncated moment-equations are not closed; rather, the lower order equations depend on higher-order moments. While this "feed-down"effect is not *directly* relevant when using ray-tracing on some particle set to compute the evolution of the moments rather than the moment-equations themselves, its existence does cast some doubt on the very validity of moments as a method of beam description: can one *really* trust a parameterization whose evolution in principle depends, however insensitively, on unknown, and perhaps unknowable, parameters?

Finally, moments do not characterize the *system* directly, because the final moments depend on both the system and the initial beam. Despite these problems, moments remain a promising tool, under developement at Los Alamos [LO88], the University of Maryland [DNR88a], and elsewhere.

- **Aberration Expansions.** Alternatively, one might attempt to use a small but carefully-chosen set of rays to numerically extract the "aberration coefficients" via a finite-difference scheme. (Aberration coefficients are basically a Taylor-series representation of the map; see §2.1.2 below.) While this is in principle possible, in practice this method results in intolerable round-off errors, even when using multiple precision arithmetic. Numerical differencing is inherently a "noise-amplifier"; one seeks small differences between large numbers, making loss of precision inevitable.

For Scatter-plots, Poincaré Sections, and any other method based on human visual capabilities, the limitation to 2-D or at most 3-D graphics may really be rather serious. Channell and Scoval [ChSc88] found that even for as simple a system as the Hénon-Hieles oscillator [HH66] (which has merely four degrees of freedom), important geometrical and dynamical features clearly emerge upon taking 3-D sections; these features appear to have been entirely unsuspected by workers using only 2-D Poincaré sections, despite many previous studies.

Finally, most numerical integration algorithms violate the important *symplectic condition* (1.10) even at *first order* in the time-step. This is not immediately fatal; since the map the integrator produces must reduce to the *identity map* in the limit of vanishing step size, the leading-order term in the map is not the first, but the *zeroth* order (identity) term, and symplecticity returns in the limit of *infinitesimal* step size. Hence, for a sufficiently small time step exact symplecticity may be approached as closely as one pleases. In particular, when using finite-precision arithmetic there must be some step-size which is sufficiently small that the violation of (1.10) will be indistinguishable from round-off error. However such a policy clearly conflicts with the minimization of both accumulated round-off error and expenditures. Furthermore, numerical integration may also not respect certain global features of the dynamics, as discussed earlier.

Overall, while ray-tracing may be sufficient to study the behavior of an *individual* ray, it is intrinsically ill-suited to the study of more global features, even so limited a feature as how a small neighborhood about some initial state transforms under the system's evolution-map.

2.1.2 Extended Transfer Matrices

The term "extended transfer matrix" (or "transfer matrix", in more common usage) is somewhat of a misnomer. A transfer matrix is actually nothing more nor less than a truncated Taylor-series representation of the perturbative initial-to-final-state evolution map; it transforms or "transfers" incoming rays into outgoing ones. The term originated in the early days of both light and corpuscular optics, when only the *linearized* paraxial approximation was in common use. Since the map for any linear system is itself linear, it could be represented by a matrix; calculating the map of a multi-element optical system could therefore be done by matrix multiplication.

More precisely, an extended transfer matrix represents the ***deviation*** of the final state of a particle from the final state of the reference trajectory in terms of a ***set of Taylor expansions*** in the ***deviation*** of the initial state of that particle from the initial state of the reference trajectory, truncated to finite order. In other words, it is a ***polynomial approximation*** to the perturbative evolution map. It is therefore more "global" than a ray-trace, because in some sense it characterizes *every* trajectory which is "sufficiently nearby" the reference trajectory.

At one time the term "aberration" meant "deviation from the reference trajectory", so the Taylor expansion was known as an "aberration expansion". Nowadays, "aberration" has come to refer to any of the image defects resulting from nonlinearities, and the terms "transfer matrix coefficient", "aberration-expansion coefficient", and "aberration coefficient" are now virtually synonomous.

Transfer matrices are simple to use, and the effects of each term are easy to interpret; moreover, they allow the useful classification of optical effects by the order of the terms producing them. The cascading of two optical-elements into a composite system results in a composition or "concatenation" of their respective maps; in terms of transfer matrices, this is substitution of one set of polynomials into another, truncated to the order of the approximation. This process may be efficiently carried out via a generalization of "Horner's rule" for evaluating polynomials, and reduces to "ordinary" matrix multiplication in first order, so that the "old" linearized theory is obtained

The transfer matrix is an expansion of the final *deviation* from the RT in terms of the initial *deviation* from the RT, so this representation is *origin-preserving*; *i.e.* the origin is a *fixed point* in this representation. (Without loss of generality, we are free to choose coordinates on phase-space such that the reference particle occupies the origin at all times.) Unfortunately, such a map cannot represent "misaligned" elements (elements whose "optical axis" is either not coincident and/or not tangent to the reference trajectory at both entrance and exit planes). As a result, the Taylor coefficients exhibit "feed-down" as well as feed-up; however this problem is common to *all* perturbative optical methods, including the author's, and, as usual, we adopt the pious hope of perturbation "theory" that the higher-order coefficients are negligibly small.

Unfortunately, *obtaining* the transfer matrix of a given optical element is difficult. Though heroically laborious hand-calculation, several workers [Bro77] were able to obtain analytical expressions for the transfer matrix coefficients of the more common optical elements through third-order, but it was not until the invention of the digital computer and FORTRAN that the use of these expressions was really practical. The most familiar of the charged-particle optics codes such as the ubiquitous TRANSPORT, and the more user-friendly GIOS are based on these calculations.

Extension to fifth order using hand calculations was considered impractical; hence, until Martin Berz developed his special-purpose symbolic manipulation program HAMILTON [BrzWln87], no fifth-order code existed. As of this writing, Berz's COSY-5.0 [BHW87] is still the only such code which is fully operational. However the University of Maryland group is hard at work on the Lie-algebraic codes MARYLIE-5.0 and MARYLIE-5.1 (see §2.3.1 below), and "beta-test" versions have recently been released to selected sites for evaluation.[1]

[1] Alex J. Dragt, (private communication).

Analytic extension of transfer matrices beyond fifth order is considered impractical by *any* method; however, Berz's "Differential Algebra", the numerical method on which this dissertation is based, in principle allows one to numerically obtain the transfer matrix *directly from the equations of motion*, to as high an order as one desires; Berz uses this approach in his recently released COSY INFINITY.[2]

A further problem with transfer matrices is that the coefficients are not all independent; they are related by the *symplectic identities* which proceed from the symplectic condition (1.10). This interdependency leads to order-of-magnitude increases in storage and computational overhead (see §2.3.1 below), and one must check to see if the symplectic identities are satisfied through the order of the calculation. This curse is not unmixed; using the accuracy to which the symplectic identities are satisfied serves as as an independent check of the accumulated errors in the calculation. Alternatively, one may choose an independent set of coefficients and use the identities to eliminate the remainder, but then the check is lost. GIOS uses the second option, while COSY allows both. TRANSPORT provides neither. (Because the non-matrix-based Lie-algebraic maps of the MARYLIE family of codes satisfy symplecticity *identically*, the identities provide no independent check of accuracy.)

Even if all of the coefficients of the transfer matrix satisfy the symplectic *identities*, it does not necessarily follow that the symplectic *condition* (1.10) will be satisfied. The matrix only provides a *polynomial approximation* to the map. While I know of no published investigation,[3] and have not attempted the proof myself, I think it likely that if one demands that the symplectic identity be satisfied exactly, then the transfer matrix coefficients are badly overdetermined by (1.10). In fact, I suspect that except perhaps for restricted classes of maps (such as the *Cremona maps* mentioned in § 2.3.1, footnote 4), the symplectic condition (1.10) might only be satisfied to one order *less* than the order of the transfer matrix, due to the derivative in the Jacobian matrix. Numerical experiments in long-term

[2]Martin Berz, (private communication)

[3]Studies of the symplectic identities have of course been done (*e.g.* [BHW87]), but only through the order of the transfer matrix. The question I am posing here is whether *additional* constraints are imposed on the coefficients of a polynomial map by the neglected HOTs.

tracking using matrix codes do indeed appear to verify that *some* sort of violation of the symplectic condition is occuring [Ser85]; whether or not the mechanism behind this is the one I propose above would require further study.

2.2 Need for Symplectic Methods

Given the critical importance of the symplectic condition (1.10), one is strongly impelled to seek methods which will satisfy it *identically*, or at least to the accuracy of the calculation. Yet because of the nature of the condition (a nonlinear partial-differential constraint), one might despair of ever accomplishing this. Remarkably, several exactly symplectic methods do exist; I summarize them in the following section.

2.3 Symplectic Methods

The methods which satisfy the symplectic condition (1.10) exactly may be again divided into broad classes similar to the non-symplectic methods: *transfer maps* (analogous to transfer matrices), and *canonical integrators* (analogous to ray-tracing). Each of these may be further broken into sub-classes based on the use of *Lie-transformations* and *"mixed" generating functions.*

2.3.1 Lie-Algrebraic Transfer Maps

For every $f \in C^1(P)$, one may naturally associate a derivation operator by means of the Poisson bracket. In the notation of Dragt and Finn [DF76],

$$:f:g \quad := \quad \{f, g\}. \tag{2.1}$$

(Dragt calls $:f:$ a *Lie operator*, and refers to it as "a Poisson-bracket waiting to happen".) Powers of a Lie operator are defined by:

$$\begin{aligned} :f:^0 g &= g, \\ :f:^2 g &:= :f:(:f:g) = \{f,\{f,g\}\}, \\ :f:^3 g &:= :f:(:f:^2 g) = \{f,\{f,\{f,g\}\}\}, \\ &\vdots \end{aligned} \tag{2.2}$$

An analytic function of $:f:$ may be defined by its power series expansion; an important special case is the *Lie transformation* generated by a function f:

$$\mathcal{F} \;:=\; \exp(:f:) \;=\; \sum_{n=0}^{\infty} \frac{1}{n!} :f:^n \tag{2.3}$$

Dragt and Finn have shown that every Lie transformation is a symplectic map, and that a broad class of symplectic maps can be written as Lie transformations. In particular, all *analytic* symplectic maps can be written as Lie transformations, and such analytic maps are exactly what "perturbation theory" produces.

The relation of Lie transformations to Hamiltonian systems is this: for a Hamiltonian which is *independent of time*, the evolution map is given by

$$\mathcal{H}(t) = \exp(-t{:}H{:}) \tag{2.4}$$

(For time *dependent* Hamiltonians a similar expression holds, involving instead the *anti*-chronologically ordered exponential of the integral of $:H:$ with respect to t, strongly reminiscent of quantum mechanics; the ordering is required because, in general, for any two times t_1 and t_2, $:H(t_1):$ and $:H(t_2):$ will not commute.)

Since the composition (or, in particle-optics circles, "concatenation") of two symplectic maps is itself a symplectic map, we might hope to build up arbitrarily complicated symplectic maps using simple Lie transformations. This is indeed possible, and Dragt and

Finn have shown that any *analytic* origin-preserving symplectic map can be written in the following *factored form*:

$$\mathcal{F} = \exp(:f_2^c:)\,\exp(:f_2^a:)\,\exp(:f_3:)\,\exp(:f_4:)\,\exp(:f_5:)\,\exp(:f_6:)\cdots \tag{2.5}$$

Here the f_n are ***homogenous polynomials*** of order n in the canonical variables; the "c" and "a" superscripts on the quadratic pieces denote parts which "commute" and "anticommute" with $J^{\mu\nu}$. Dragt and Finn call the coefficients appearing in the above polynomials "Lie coefficients".

To sketch how the Lie transformation corresponding to an analytic symplectic map may be constructed, note that from (1.4), the result of a homogenous polynomial Lie operator acting on a homogenous polynomial is another homogenous polynomial:

$$:f_n:f_m = \{f_n, f_m\} = f_{n+m-2}. \tag{2.6}$$

Since the canonical coordinates ξ^μ are themselves trivially homogenous polynomials of order one, one sees that the Taylor series expansion of $\exp(:f_k:)\,\xi^\mu$ contains only terms of order $n(k-2)+1$, $n = 0, 1, 2, \ldots$ in the ξ^μ, with the leading term after ξ^μ being of order $(k-1)$. Since Lie transformations satisfy the symplectic constraint (1.10) ***identically***, it appears we may have found a way to describe a symplectic map with ***no redundant parameters***; for if one could find at each order $(n-1)$ an f_n which annihilates the leading term in the remainder (*i.e.*, an f_n such that $(\mathcal{M} - \mathcal{F}_{(n)})\xi^\mu = \mathcal{O}(n+1)$, where $\mathcal{F}_{(n)}$ is a map of Dragt-Finn factored form, (2.5), terminated at the factor $\exp(:f_n:)$), then one has in fact constructed constructed a Lie-transform representation of $\mathcal{M}$. The condition for such an f_n to exist is just the symplectic condition, as shown by Dragt and Finn [DF76], and also by Forest [For84]. Therefore one may freely interconvert between the transfer map and transfer matrix representations of a symplectic map, using whichever is most convenient for the calculation at hand.

Dragt and Forest [For84, DF83] have also shown how to obtain a set of ODEs for the Lie coefficients, given the series expansion of the Hamiltonian about the RT. This requires that the analyst first perform a canonical transformation to coordinates centered on the RT, then expand about it; this procedure has been automated in Healy's SMP program ANALIE [Hea86]. Ryne's FORTRAN program GENMAP [Ryn87] may then be used to obtain the Lie coefficients by integrating the ODEs.

Let us compare the number of coefficients required for a transfer matrix *versus* a transfer map. The number of monomials of order less than or equal to n in v variables is:

$$ {}_nN_v := \frac{(n+v)!}{n!\,v!}; \tag{2.7}$$

it obeys the useful recursion relation:

$$ {}_nN_v = {}_nN_{v-1} + {}_{n-1}N_v. \tag{2.8}$$

as may be shown by induction, and easily verified by direct substitution of (2.7) into (2.8).

The transfer matrix representation consists of v sets of ${}_nN_v$ monomials; the transfer map, on the other hand, needs only one set, but of one order higher: ${}_{n+1}N_v$. Taking the quotient of these, we find that a transfer matrix will involve $v(n+1)/(n+v+1)$ times as many coefficients as the equivalent Lie-algebraic transfer map. One sees that for maps of order n greater than about v, the Lie algebraic approach can lead to substantial savings.

Lie transformations are analytically useful; for example, perturbation theory becomes compact and elegant in Dragt-Finn form, minimizing the number of coefficients one must calculate [Car81, § 5]; the treatment of symmetries and invariants also becomes straightforward and elegant [Car81, § 3]. While the physical meaning of a Lie-algebraic transfer map is not as transparent as that of a transfer matrix, the analyst now has access to powerful Lie-algebraic classification and analysis tools developed by Dragt and his collaborators (such as Dragt's *resonance-basis* [Dra87], and Forest's *normal-form algorithms* [FDL87, For89]).

However, one should not think there is anything "magic" about a Lie-algebraic transfer map. Order by order, a transfer map contains no more *information* than the equivalent transfer matrix; it simply provides a systematic way of calculating a set of HOTs satisfying the symplectic condition to all orders; indeed, one may freely convert between the two representations using the Dragt-Finn-Forest algorithm [DF76, For84]. However, given a set of Taylor-series coefficients through order n, the knowledge that (1.10) is satisfied is insufficient to uniquely determine the HOTs. There is no *unique* set of HOTs which satisfy the symplectic identities; many different truncation schemes are possible.[4] Furthermore, the concatenation of transfer maps requires repeated use of the Baker-Campbell-Hausdorff (BCH) [SW86] theorem to return it to the Dragt-Finn factored form. This process rapidly becomes more complex and unwieldy as one goes to higher orders, especially since no *explicit* formula for the concatenation process has yet been found, (although an algorithm to construct it does exist). By contrast, concatenation of high-order transfer matrices is perfectly straightforward, as the algorithm is entirely order-independent.

Finally, since Lie algebraic transfer maps are intrinsically an *operator* method, they provide no real advantage if one actually wishes to *track* particles though the map, because closed-form expressions for the series (2.3) have only been obtained in certain special cases.[2] One must either convert the map to a matrix, losing symplecticity, or invoke some sort of implicit solution method [Ner86]. For many applications, the explicit symplecticity of Lie transformations is largely an illusory advantage; the real gain is in the functional independence of the Lie coefficients.

[4]One area of active investigation by Dragt's group is whether the homogenous-polynomial basis of (2.5) may be replaced by a *nilpotent* basis, as the series (2.3) would then terminate. This is essentially the same question as the one I alluded to in §2.1.2: do *polynomial* (or, more generally, *algebraic*) symplectic maps exist? It turns out that the answer is in the affirmative: a class of algebraic symplectic maps called "Cremona maps" exist, and provide an interesting alternative basis for factoring Lie transformations [Ran].

2.3.2 Symplectic Integrators

Generating Function Based

Symplectic integrators are numerical integration algorithms which respect the condition (1.10) *exactly*; most of the "standard" integrators do not have this property, as stated in §1.5. The first such algorithms were developed in 1955 by R. De Vogelaére in a series of unpublished papers [DeV56]. No further work appears to have been done until 1983, when R. Ruth [Rut83] showed that for certain choices of the Runge-Kutta weights, the discretized canonical EOMs for Hamiltonians of the form:

$$H(\vec{x},\vec{p}) = \frac{1}{2m}\vec{p}^{\,2} + V(\vec{x}) \tag{2.9}$$

would preserve (1.10) exactly. Independently in the same year, P. Channell [Cha83] developed a systematic implicit method for general Hamitonian systems which reduced to Ruth's algorithm for Hamiltonians of the form (2.9).

Channell's "Runge-Kutta type" method uses the fact that a "mixed" canonical generating-function produces an exactly symplectic map [Gol80, LLa]. He chooses (using Goldstein's terminology) an F_3-type generating-function (subscripts refer to times):

$$\boldsymbol{p}_1 = -\frac{\partial}{\partial \boldsymbol{q}_1} K(\boldsymbol{q}_1,\boldsymbol{p}_2) \qquad \boldsymbol{q}_2 = -\frac{\partial}{\partial \boldsymbol{p}_2} K(\boldsymbol{q}_1,\boldsymbol{p}_2) \tag{2.10}$$

and assumes an expansion of K in powers of $\Delta t := (t_2 - t_1)$:

$$K = \sum_{m=0}^{\infty} \frac{1}{m!}\Delta t^m K_m(\boldsymbol{q}_1,\boldsymbol{p}_2) \tag{2.11}$$

Then he substitutes (2.10) and the canonical equations (1.3) into the following identity:

$$\dot{\boldsymbol{p}}_1 = \frac{\partial \boldsymbol{p}_1}{\partial t_2} + \frac{\partial \boldsymbol{p}_1}{\partial \boldsymbol{q}_1} \cdot \dot{\boldsymbol{q}}_1. \tag{2.12}$$

Defining the auxiliary quantity:

$$\Delta \boldsymbol{p} := (\boldsymbol{p}_2 - \boldsymbol{p}_1) = \sum_{n=1}^{\infty} \frac{\Delta t^n}{n!} \boldsymbol{P}_n(\boldsymbol{q}_1,\boldsymbol{p}_2), \tag{2.13}$$

Channell obtains:

$$\sum_{n=0}^{\infty} \frac{\Delta t^n}{n!} \boldsymbol{P}_{n+1} = \sum_{n=1}^{\infty} \frac{\Delta t^n}{n!} \frac{\partial^2 K_n}{\partial q \partial \boldsymbol{q}} \cdot \left(\sum_{m=0}^{\infty} \frac{1}{m!} \left(\Delta \boldsymbol{p} \cdot \frac{\partial}{\partial \boldsymbol{p}} \right)^m \right) \frac{\partial H}{\partial \boldsymbol{p}} - \sum_{n=0}^{\infty} \frac{1}{n!} \left(\Delta \boldsymbol{p} \cdot \frac{\partial}{\partial \boldsymbol{p}} \right)^n \frac{\partial H}{\partial \boldsymbol{q}} \tag{2.14}$$

Assuming the K_m may also be expanded in powers of $\boldsymbol{q}_1$ and $\boldsymbol{p}_2$, he truncates (2.14) to finite order, and equates like powers to determine K: the result is not very illuminating, and I shall not have need of it, anyway.

Once the series representation of K is obtained, one may use (2.10) to push the particle forward by the time-step Δt. Note that this is an *implicit* formula; some sort of iterative method (*e.g.* Newton-Raphson) must be used to solve it. Fortunately, an excellent initial guess is provided by any non-symplectic method of the same accuracy, so the result will converge rapidly. One might therefore use the Channell-Scoval Runge-Kutta algorithm to provide a symplectifying "corrector step" to one's favorite non-symplectic integrator.

Note the presence of high-order partial derivatives in (2.14). Unlike a standard Runge-Kutta algorithm, it will *not* be sufficient to approximate the partials by finite differences; the analytical formulas must be used. To grind these out by hand would be quite tedious and error-prone; fortunately, the advent of symbolic manipulators such as SMP, MACSYMA, and *Mathematica* capable of both evaluating (2.14) and automatically translating the results into FORTRAN code make use of (2.14) feasible. Nevertheless, this is a high price to pay, for unlike standard Runge-Kutta algorithms, which are "general purpose", the integrator must, in effect, be rederived each time. As deriving the integrator from even simple Hamiltonians may require tens of minutes (or perhaps even hours), and may result in several *thousand* lines of code, one can only afford to use this symplectic integrator when one absolutely *has* to.

To combat this, Channel and Scoval attempted to develope an "Adams-type" integrator, which uses the information obtained in previous evaluations of the right-hand-side of the EOMs to cut down the amount of analytical work required. Their strategy was to fit a local polynomial model to the true Hamiltonian; the symplectic integrator for their assumed polynomial form could then be derived 'once and for all". Unfortunately, they found that this strategy required storing on the order of v^{n-1} previous evaluations of the EOMs, and also on the order of v^n Hamiltonian coefficients, which must be determined by solving a similar number of linear equations. Hence, while the Adams-type integrator is faster for systems with only a few degrees of freedom, as the number of degrees of freedom becomes large, the Adams-type integrator becomes impractical.

An alternative symplectic integrator may be derived from the Hamilton-Jacobi equation:

$$\frac{\partial}{\partial t}S + H(\boldsymbol{q}_2, \frac{\partial}{\partial \boldsymbol{q}_2}S) = 0 \tag{2.15}$$

Here, $S(\boldsymbol{q}_1, \boldsymbol{q}_2; t)$ is an F_1-type rather than an F_3-type generating function:

$$\boldsymbol{p}_1 = -\frac{\partial}{\partial \boldsymbol{q}_1}S, \qquad \boldsymbol{p}_2 = \frac{\partial}{\partial \boldsymbol{q}_2}S \tag{2.16}$$

Channell and Scoval set $H = H_0 + \epsilon V$, where H_0 is chosen to be the *free* Hamiltonian, V is the remainder (assumed to depend only on $\boldsymbol{q}_2$ in their example), and ϵ is an ordering parameter which they set to unity at the end of the calculation.

They assume the ansatz:

$$S = S_0 + \sum_{n=1}^{\infty} \epsilon^n S_n \tag{2.17}$$

where S_0 is the *free* generating function; for a nonrelativistic particle, it is:

$$S_0 = \frac{(\boldsymbol{q}_2 - \boldsymbol{q}_1)^2}{2\,\Delta t} \tag{2.18}$$

They then insert (2.17) into (2.15) and collect like terms in ϵ. The result is the following recursion relation:

$$\frac{\partial S_1}{\partial t} + \frac{\partial S_0}{\partial \boldsymbol{q}_2} \cdot \frac{\partial S_1}{\partial \boldsymbol{q}_2} = -V \tag{2.19}$$

$$\frac{\partial S_n}{\partial t} + \frac{\partial S_0}{\partial \boldsymbol{q}_2} \cdot \frac{\partial S_n}{\partial \boldsymbol{q}_2} = -\frac{1}{2}\sum_{m=1}^{n} \frac{\partial S_m}{\partial \boldsymbol{q}_2} \cdot \frac{\partial S_{n-m}}{\partial \boldsymbol{q}_2}, \qquad n \geq 2 \tag{2.20}$$

Expanding $V(\boldsymbol{q}_2)$ about some *arbitrary* point $\bar{\boldsymbol{q}}$, inserting this into (2.19) and (2.20), and collecting like terms, they again obtain some rather complicated expressions, which are then integrated with respect to t by using the fact that the left-hand-sides of equations (2.19) and (2.20) both have the very simple characteristics: $\boldsymbol{q}_2 = \boldsymbol{q}_1 + \boldsymbol{v}_0 \Delta t$, where $\boldsymbol{v}_0$ is the initial velocity. The result is, again, not very illuminating, although one notices the following interesting symmetry:

$$S_i(\boldsymbol{q}_1, \boldsymbol{q}_2; t) = -S_i(\boldsymbol{q}_1, \boldsymbol{q}_2; -t)$$

which is a consequence of the time-reversibility of Hamiltonian systems.

The parameter $\bar{\boldsymbol{q}}$ is a sort of "gauge freedom", which may be chosen at will by the analyst. By choosing it to be either $\boldsymbol{q}_1$, or a non-symplectic estimate of $\boldsymbol{q}_2$, the terms in the final expressions involving $(\boldsymbol{q}_1 - \bar{\boldsymbol{q}})$ or $(\boldsymbol{q}_2 - \bar{\boldsymbol{q}})$ may be made small; another interesting choice would be the estimated value of $\boldsymbol{q}$ at $t = t_1 + \frac{1}{2}\Delta t$. Each choice seems to yield substantially similar results.

Lie-Algebra Based

There are also symplectic integrators based on Lie-algebraic techniques; the simplest of these is Neri's "leapfrog method" [Ner87], for time-independent Hamiltonians of the "A+B" type. By "A+B", I mean that $H = A + B$, where A and B are Hamiltonians whose maps can be solved exactly, but which do not "commute",[5] *i.e.* $\{A, B\} \neq 0$. The exact map for H acting over a time t is $\mathcal{H}(t) = \exp(-t{:}H{:})$; let $\mathcal{A}(t) = \exp(-t{:}A{:})$ and $\mathcal{B}(t) = \exp(-t{:}B{:})$ be the maps of A and B acting alone. Using the BCH theorem, it can be shown that:

$$\mathcal{A}(t)\,\mathcal{B}(t) = \mathcal{H}(t) + \mathcal{O}(t^2), \qquad (2.21)$$

$$\mathcal{B}(t)\,\mathcal{A}(t) = \mathcal{H}(t) + \mathcal{O}(t^2); \qquad (2.22)$$

but

$$\mathcal{A}(\tfrac{1}{2}t)\,\mathcal{B}(t)\,\mathcal{A}(\tfrac{1}{2}t) = \mathcal{H}(t) + \mathcal{O}(t^3). \qquad (2.23)$$

[5]Actually, it is the Lie operators ${:}A{:}$ and ${:}B{:}$ which do not commute; however because of the operator-identity $[{:}A{:}, {:}B{:}] := [{:}A{:}{:}B{:} - {:}B{:}{:}A{:}] = {:}\{A, B\}{:}$, one sometimes abuses language by saying that A and B do not commute. Technically, I should probably say "A and B are not in involution".

Formula (2.23) constitutes one time-step of Neri's "leapfrog" method. He has also obtained an order-$\mathcal{O}(t^5)$ "leapfrog" map:

$$\left[\mathcal{A}(\tfrac{1}{2}\alpha t)\mathcal{B}(\alpha t)\mathcal{A}(\tfrac{1}{2}\alpha t)\right]\left[\mathcal{A}(\tfrac{1}{2}\beta t)\mathcal{B}(\beta t)\mathcal{A}(\tfrac{1}{2}\beta t)\right]\left[\mathcal{A}(\tfrac{1}{2}\alpha t)\mathcal{B}(\alpha t)\mathcal{A}(\tfrac{1}{2}\alpha t)\right] = \mathcal{H}(t) + \mathcal{O}(t^5), \tag{2.24}$$

where

$$\alpha := \left(\frac{1}{2-\sqrt[3]{2}}\right), \qquad \beta := -\left(\frac{\sqrt[3]{2}}{2-\sqrt[3]{2}}\right)$$

I have pointed out to Neri that (2.24) is a "leapfrog map made out of leapfrog maps", *i.e.* a map of "ABA" form having factors "A" and "B" that are themselves of "ABA" form. It is tempting to speculate that this pattern continues to arbitrarily high order: perhaps one could go on eliminating two more powers of t in the error-term at each step, by constructing an $\mathcal{O}(t^{2n+3})$ "ABA" map out of factors having the same form as the $\mathcal{O}(t^{2n+1})$ "ABA" map obtained in the previous step.

Note that in both (2.23) and (2.24) (since $2\alpha + \beta = 1$), the sum of the arguments of the $\mathcal{A}$ maps, and also the $\mathcal{B}$ maps, each add up to the total length of the system, t. Marsden (private communication) has commented on the similarity of (2.23) to the "Trotter product" formula [AMR88, p.287]; also [CHMM78].

The "leapfrog" method also has a generalization of (2.23) to $H = A + B + C$:

$$\mathcal{A}(\tfrac{1}{2}t)\,\mathcal{B}(\tfrac{1}{2}t)\,\mathcal{C}(t)\,\mathcal{B}(\tfrac{1}{2}t)\,\mathcal{A}(\tfrac{1}{2}t) = \mathcal{H}(t) + \mathcal{O}(t^3). \tag{2.25}$$

Clearly, one could continue splitting exactly solvable pieces off of H until it has been completely decomposed into maps one can evaluate exactly, resulting in a map of "ABC ... Z ... CBA" form.

Neri claims ([Ner87]; see also [For87, footnote 16]) that all of Ruth's symplectic integrators may be factored into a composition of "drifts, rotations, and kicks",[6] each of which may be evaluated exactly in closed form.

[6]"rotations" also include rotations between p's and q's. Both "drifts" and "kicks" are types of *shears* and may be nonlinear. Artin [Art57, p.137] has shown that all *linear* symplectic transformations may be factored into a product of shears, so a nonlinear generalization should perhaps be not too surprising; this nonlinear generalization is closely related to the *Cremona maps* mentioned in footnote 2, this chapter.

Neri also claims that similar formulas may be derived for time-*dependent* Hamiltonians, if the "A" and "B" maps are evaluated at the proper times; *i.e.*, the map satisfies:

$$\mathcal{A}_1\mathcal{B}_2\mathcal{A}_3 = \overline{\mathbf{T}}\exp\left\{-\int_{t_i}^{t_f} :H(t):\,dt\right\} + \mathcal{O}(t^3) \tag{2.26}$$

where

$$\mathcal{A}_1 := \exp(-\tfrac{1}{2}\Delta t{:}H(t_1){:}),\quad \mathcal{B}_2 := \exp(-\Delta t{:}H(t_2){:}),\quad \mathcal{A}_3 := \exp(-\tfrac{1}{2}\Delta t{:}H(t_3){:});\quad \Delta t := (t_f - t_i)$$

where the intermediate values of the independent variable, $t_i < t_1 < t_2 < t_3 < t_f$, have been chosen to cancel offending terms *a la* Gaussian quadrature. $\overline{\mathbf{T}}$ is the "anti-chronological ordering operator" mentioned in §2.3.1.

2.3.3 Generating-Function Methods

Neri's Symplectic Tracking Algorithm

As discussed in §2.3.1, Lie-algebraic maps are not immediately useful for tracking purposes. One method for "symplectic tracking", developed by Neri for the MARYLIE code, is to convert the nonlinear part of the factored transfer map (2.5) into an "equivalent" canonical generating function which is a polynomial of order n in its (mixed) variables. These generating functions are "equivalent" to the map in the sense that the Taylor expansion of the map generated by the canonical transformation and that of the transfer map agree through order $(n-1)$. The implicit equations for the canonical transformation are solved numerically via a Newton-Raphson iteration method.

Neri does this by "un-factoring" the nonlinear part of the map, to obtain a polynomial of order 3 through n, the "pseudo-Hamiltonian", h. He uses this pseudo-Hamiltonian to compute a generating function *via* an HJ-equation, in a manner very similar to the canonical integrators of Channel and Scoval:

$$\partial_\tau S(\boldsymbol{q},\bar{\boldsymbol{p}};\tau) = -h(\boldsymbol{q},\partial_{\boldsymbol{q}} S(\boldsymbol{q},\bar{\boldsymbol{p}};\tau)) \tag{2.27}$$

The coefficients of the generating function are then determined by "Picard iteration", with the "artificial time" variable τ serving as the "ordering parameter". Neri assumes $S_k(\boldsymbol{q},\bar{\boldsymbol{p}};\tau)$, for all $k \geq 2$, is a power-series in τ through order k, and S_2 is initially taken to be the "identity" generating function:

$$S_2(\boldsymbol{q},\bar{\boldsymbol{p}}) := \sum_i \bar{p}_i q^i. \tag{2.28}$$

He substitutes S into (2.27), and integrates with respect to τ, keeping only those terms through order $(k+1)$ in τ:

$$S_{k+1}(\boldsymbol{q},\bar{\boldsymbol{p}};\tau) = S_2(\boldsymbol{q},\bar{\boldsymbol{p}}) - \int_0^\tau h(\boldsymbol{q},\partial_{\boldsymbol{q}} S_k(\boldsymbol{q},\bar{\boldsymbol{p}};\tau'))d\tau'. \tag{2.29}$$

The constant of integration has been set to the identity, S_2, because S_{k+1} must also reduce to the identity for $\tau = 0$.

Essentially, the artificial time τ merely acts as a "bookkeeping" device, similar to Channell and Scoval's parameter ϵ, and in the end, both are set to unity. In Neri's case, the result is a generating function which produces a canonical transformation agreeing with the Lie-algrebraic transfer map $\exp(:h:)$ through order $(n-1)$; however the HOTs will differ more and more as the order increases. Neri has found [Ner86] some evidence that the HOTs resulting from the finite-order polynomial generating function resulting from (2.29) may be uncomfortably large; therefore, other algorithms based on the "Cremona maps" (see footnote 2, this chapter) are currently being investigated.

Warnock and Ruth's Fourier/Hamilton-Jacobi Method

Warnock and Ruth [RRW85, WR87] have also developed a method based on the HJ-equation. They assume the Hamiltonian has been expressed in terms of the "action-angle" variables of the *linearized* system, and is a periodic function (mod 2π) in all variables. Using a combination of Green-function and Fourier techniques, they eliminate the "secular terms" introduced by nonlinear resonances, and solve for the Fourier coefficients to yield a system of simultaneous nonlinear *algebraic* equations, although still involving integrals over all the angle variables. They develop an efficient method of solving these equations using "fast Fourier transforms" and Newton-Raphson iteration.

While in principle applicable to any multiply-periodic system, the need to first transform to action-angle variables is somewhat limiting; it requires that the users already know the behaviour of their linearized system, and expend the labor of placing their Hamiltonian in action-angle form. Although in principle this process can be automated, it will still be somewhat of a nuisance (except to accelerator theorists, who often think in terms of these variables under the aliases of "Courant-Snyder invariants" and "phase advances".)

2.3.4 The Hamilton-Jacobi/Differential-Algebra Method

The Hamilton-Jacobi/Differential-Algebra (HJ/DA) method described in this dissertation bears certain resemblences to several of the above methods. As its name implies, the HJ/DA method uses the HJ-equation to obtain a canonical generating-function representation of the map of a system. I assume that the map about the reference trajectory may be represented as a truncated Taylor series in "mixed" variables, initialized to the identity, the coefficients of which are assumed to be functions of time. By substituing this ansatz into the H-J equation, expanding the Hamiltonian, and equating like terms, the HJ-equation is converted into a system of nonlinear ODEs, which may be solved numerically using standard methods. Since it uses a generating function, the resulting map will be identically symplectic. The generating function may be used to obtain other representations of the map, or to track particles directly using methods similar to Neri's.

The signal advantage of my method over those above is its simplicity; the analyst need not perform any laborious analytical calculations before using it, because the transformation to the RT-centered coordinates and the series-expansions are entirely automated via Berz's remarkable method of *differential algebra* (see [Ber87, Ber88, Ber89a, Ber89b]; I will give a short introduction to DA in the next chapter). With a minimum of knowledge as to how DA works and is implemented, one need only write a FORTRAN subroutine to calculate the Hamiltonian, in whatever coordinates one happens to find convienient, and flag certain statements for processing by the DAFOR precompiler, described in §3.8; many hours of human labor may therefore be saved.

Chapter 3

Basic Concepts of Differential Algebra (DA)

Differential algebra is a remarkable and powerful numerical method recently developed by Martin Berz [Ber87, Ber88, Ber89a, Ber89b]. By systematically exploiting certain properties of the "product" and "chain" rules, DA in effect "teaches" a computer just enough differential calculus to allow it to compute the numerical values of the *analytical* derivatives of functions, to as high an order as one is willing to pay for. These derivatives are *not* finite differences, but true analytical derivatives.

DA allows the numerical analyst to obtain easily, almost effortlessly, derivatives of the results of any complicated algorithm with respect to *any* continuous parameter appearing in it, by making only a few minor alterations to the existing program. Typically, these derivatives may be obtained at a cost of only a few times more than it would have cost to evaluate the function alone. DA allows the numerical analyst to consider for the first time using algorithms which require knowledge of both functions and their analytical derivatives; such algorithms have often been considered impractical, because for complicated functions the derivatives may be too hard to obtain.

Finally, DA opens up the possibilty of completely new algorithms having no counterpart in "traditional" numerical methods.

3.1 Brief History of Differential Algebra and its Precursors

The term "differential algebra" was introduced by Ritt [Rit50] in 1950, in a treatise on algebraic aspects of systems of differential equations. A formal differential algebra in v variables over a field $\mathcal{F}$ is a graded commutative ring defined by the set $D_v(\mathcal{F}) := \{\mathcal{F}, +, \cdot, \partial_i, i = 1, \ldots, v\}$. Here $\mathcal{F}$ is an algebraic field of characteristic zero, which is itself embedded in $D_v(\mathcal{F})$ as a subalgebra. The addition and multiplication operations $+$ and $\cdot$ are commutative and associative, with multiplication distributive over addition; they reduce to ordinary addition and multiplication on the subalgebra isomorphic to $\mathcal{F}$. The ∂_i are a set of v unary operators taking $\mathcal{F}$ into itself, satisfying the Liebniz rule: $\partial_i(ab) = \partial_i ab + a\partial_i b$, for all $a, b \in D_v$ and $i = 1, \ldots, v$. Of necessity, every nontrivial representation of $D_v(R)$ must be infinite dimensional, so Ritt's differential algebras are not suitable for numerical implementation. The important concept here, though, is that in D_v, differentiation is a purely *algebraic* operation.

"Differentiation as algebra" has been considered before; for example, there are data-structures and operations which implement the "sum", "product", and "chain" rules [Ral84, Jer89], usually to at most first or second order in a fixed number of variables; in the literature, this approach is usually refered to under the name "automatic differentiation". Also, algorithms for recursively computing Taylor series coefficients for algebraic functions of a single variable to high order have been repeatedly rediscovered [Ste56, Gib60, Moo66], and even implemented into a FORTRAN prepocessor [KW69]. However, Berz appears to be the first to recognize that the *quotient differential algebras* formed by "moding out" elements of order higher than n generalizes these methods to arbitrary n and v, allowing one to extract the numerical values of derivatives for arbitrarily complicated functions to machine precision.

3.2 DA as a Subset of Nonstandard Analysis

DA may be viewed as a subset of "nonstandard analysis" (NSA). NSA is a consistent generalization of the field of real numbers, R, to include infinitely small quantities, or "infinitesimals", and also infinitely large quantities [Rob61]; the resulting field of "nonstandard reals" is denoted by $^*R$. A differential algebra of order n is equivalent to the subset of $^*R$ containing only the reals and infinitesimals through some fixed order n. Berz denotes a differential algebra of order n in v variables by $_nD_v$. I will find it convienient to refer also to the subsets of $_nD_v$ which contain only those elements which are of order j and higher; I shall denote them by $^j_nD_v$.

To intoduce the concepts behind DA, I will start with the simplest example: the differential algebra of a single variable to first order, or $_1D_1$.

Consider the real vector space of ordered pairs, $(f_1, f_2) \in R^2$. Vector addition and scalar multiplication are defined in the usual way:

$$(f_0, f_1) + (g_0, g_1) := (f_0+g_0, f_1+g_1) \tag{3.1}$$

$$\alpha(f_0, f_1) := (\alpha f_0, \alpha f_1) \tag{3.2}$$

for all f_0, f_1, g_0, g_1, and α in R. Now define a "vector multiplication" as follows:

$$(f_0, f_1) \cdot (g_0, g_1) := (f_0 g_0, f_1 g_0 + f_0 g_1) \tag{3.3}$$

With the definition of a product, the vector space becomes an ***algebra***, the differential algebra $_1D_1$. One can easily show this product to be commutative, associative, and distributive with respect to addition.

An ordering[1] may be defined on $_1D_1$ in the following way: Given any two vectors (a, b) and (c, d), define:

$$(a,b) < (c,d) \qquad \text{if} \quad \{\, (a < c) \text{ or } (a = c \text{ and } b < d) \,\}$$

[1]This ordering is just the *lexicographic* or "dictionary" ordering.

$$
\begin{aligned}
(a,b) > (c,d) \quad & \text{if} \quad \{\, (a > c) \text{ or } (a = c \text{ and } b > d) \,\} \\
(a,b) = (c,d) \quad & \text{iff} \quad \{\, (a = c) \text{ and } (b = d) \,\}
\end{aligned}
\tag{3.4}
$$

Clearly, for every pair of vectors (a,b) and $(c,d) \in {}_1D_1$, exactly one of the above cases must be true to the exclusion of the others; furthermore, for $(a,b) < (c,d)$ and an arbitrary (e,f), $(a,b) + (e,f) < (c,d) + (e,f)$, and for all $e > 0$, $(a,b)\cdot(e,f) \leq (c,d)\cdot(e,f)$.

One can show that the subset ${}_1R_1 := \{ r \in {}_1D_1 : r = (\rho,0),\, \rho \in R\}$ has exactly the same properties as the real numbers. So R may be imbedded in ${}_1D_1$ in much the same way it is imbedded in C. Another interesting subset is the set of elements with vanishing real part, but non-vanishing first order part, ${}^1_1D_1 := \{d \in {}_1D_1 : d = (0,\delta),\, \delta \in R\}$. Let R_+ denote the *positive reals*, $R_+ := \{r \in R : r > 0\}$. One can show that the *positive* elements of ${}^1_1D_1$ (*i.e.* elements such that $d > (0,0)$) have the following interesting property:

$$(\rho,0) > (0,\delta) > (0,0), \tag{3.5}$$

for all $\delta, \rho \in R_+$; therefore $(0,\delta)$ lies "in between" zero and *every positive real number, no matter how small*: it is an "infinitesimal".

We shall call $a := \mathcal{R}(a,b)$ the "real part" of (a,b), and $b := \mathcal{D}(a,b)$ the "differential part". We shall call $d := (0,1)$ the "unit differential"; it has the interesting property that $d^2 := d \cdot d \equiv 0$, and might therefore be thought of as a "square root of zero".

It is easy to verify that $\mathbf{1} := (1,0)$ forms the multiplicative "neutral", or "unit" element of ${}_1D_1$:

$$(1,0)\cdot(a,b) = (a,b)\cdot(1,0) \equiv (a,b), \tag{3.6}$$

whereas the powers of d form a complete basis for ${}_1D_1$:

$$d^0 := (1,0) \tag{3.7}$$

$$d^1 = (0,1) \tag{3.8}$$

$$d^n \equiv (0,0), \quad (n \geq 2) \tag{3.9}$$

where (3.7) is needed for consistency, while the "nilpotent property" (3.9) follows from the multiplication rule (3.3).

Note that (3.3) shows that ${}_1D_1$ is not a field, because

$$(0,a)\cdot(0,b) \equiv (0,0), \quad \forall\, a,b \in R$$

so divisors of zero exist in ${}_1D_1$. Technically, ${}_1D_1$ is a commutative ring with identity; one can easily show that (a,b) has a multiplicative inverse if and only if $a \neq 0$. When the inverse exists, it is given by:

$$(a,b)^{-1} := \left(\frac{1}{a}, -\frac{b}{a^2}\right), \tag{3.10}$$

and one can easily verify that $(a,b)\cdot(a,b)^{-1} \equiv (1,0)$.

Turning now to powers of DA quantities, one notes the following interesting set of examples: if x is any real number, one finds that

$$\begin{aligned}
(x,1)^2 &= (x+d)^2 = x^2 + 2xd + d^2 = x^2 + 2xd = (x^2, 2x) && (3.11)\\
(x,1)^3 &= (x+d)^3 = x^3 + 3x^2d + 3xd^2 + d^3 = x^3 + 3x^2d = (x^3, 3x^2) && (3.12)\\
&\ \vdots\\
(x,1)^n &= (x+d)^n = x^n + nx^{n-1}d + \frac{n(n-1)}{2!}x^{n-2}d^2 + \ldots && (3.13)\\
&= x^n + nx^{n-1}d = (x^n, nx^{n-1})\\
&\ \vdots
\end{aligned}$$

One immediately recognizes, in the above, the "power rule" of elementary differential calculus.

Similarly, for inverse powers:

$$\begin{aligned}
(x,1)^{-1} &= \left(\frac{1}{x}, -\frac{1}{x^2}\right) && (3.14)\\
(x,1)^{-2} &= \left[(x+d)^2\right]^{-1} = (x^2, 2x)^{-1} = \left(\frac{1}{x^2}, -\frac{2x}{x^4}\right) = (x^{-2}, -2x^{-3}) && (3.15)\\
(x,1)^{-3} &= \left[(x+d)^2\right]^{-1} = (x^3, 3x^2)^{-1} = \left(\frac{1}{x^3}, -\frac{3x^2}{x^6}\right) = (x^{-3}, -3x^{-4}) && (3.16)
\end{aligned}$$

$$\begin{aligned}
&\vdots \\
(x,1)^{-n} &= [(x+d)^n]^{-1} = (x^n, nx^{n-1})^{-1} = \left(\frac{1}{x^n}, -\frac{nx^{n-1}}{x^{2n}}\right) \\
&= (x^{-n}, -nx^{-n-1}) \qquad (3.17) \\
&\vdots
\end{aligned}$$

which one again recognizes as the "power rule". One begins to suspect that DA has something to do with derivatives, and that the derivative of any function may be calculated *algebraically*, without the use of limits, by simply evaluating it at $(x+d)$. This is the origin of the name, "differential algebra", for DA is the *algebra of derivatives.*

The connection between DA and derivatives continues to hold for more complicated functions. If one agrees to define analytic functions of DA-valued quantities by their Taylor expansions, one obtain the following results for the exponential, sine, cosine, and natural logarithmic functions.

For the exponential function, begin by splitting $x+d$ into its real and differential parts. Using the algebraic properties of the exponential yields:

$$\begin{aligned}
\exp(x+d) &= e^x e^d = e^x\left[1 + d + \frac{1}{2!}d^2 + \ldots\right] \\
&= e^x[1+d] = (e^x, e^x). \qquad (3.18)
\end{aligned}$$

So the differential part of $\exp(x+d)$ is indeed its derivative.

We may check two other definitions of the exponential to see if they are consistent with (3.18). Using the power-series definition of $\exp(\cdot)$,

$$\begin{aligned}
\exp(x+d) &= \sum_{n=0}^{\infty} \frac{1}{n!}(x+d)^n = \sum_{n=0}^{\infty} \frac{1}{n!}(x^n + nx^{n-1}d) \\
&= \sum_{n=0}^{\infty} \frac{1}{n!}x^n + \sum_{n=1}^{\infty} \frac{1}{(n-1)!}x^{n-1}d \\
&= e^x + e^x d = (e^x, e^x), \qquad (3.19)
\end{aligned}$$

while using the elementary calculus definition of exp(·) in terms of limits,

$$\begin{aligned}\exp(x+d) &= \lim_{n\to\infty}\left[1+\frac{1}{n}(x+d)\right]^n = \lim_{n\to\infty}\left[\left(1+\frac{x}{n}\right)+\frac{d}{n}\right]^n \\ &= \lim_{n\to\infty}\left[\left(1+\frac{x}{n}\right)^n + n\left(1+\frac{x}{n}\right)^{n-1}\frac{d}{n}\right] \\ &= \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n + \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n \lim_{n\to\infty}\left(1+\frac{x}{n}\right)^{-1} d \\ &= e^x + e^x d = (e^x, e^x) \qquad (3.20)\end{aligned}$$

So all three definitions are consistent.

For the trigonomentric functions sin(·) and cos(·), we may perform a similar splitting of the argument into "real" and "differential" parts to yield:

$$\begin{aligned}\sin(x+d) &= \sin(x)\cos(d)+\cos(x)\sin(d) \\ &= \sin(x)[1-\tfrac{1}{2!}d^2+\ldots]+\cos(x)[d-\tfrac{1}{3!}d^3+\ldots] \\ &= \sin(x)+\cos(x)d = (\sin(x),\cos(x)) \qquad (3.21)\end{aligned}$$

and

$$\begin{aligned}\cos(x+d) &= \cos(x)\cos(d)-\sin(x)\sin(d) \\ &= \cos(x)[1-\tfrac{1}{2!}d^2+\ldots]-\sin(x)[d-\tfrac{1}{3!}d^3+\ldots] \\ &= \cos(x)-\sin(x)d = (\cos(x),-\sin(x)) \qquad (3.22)\end{aligned}$$

Again, one finds that the differential part is just the derivative.

For the natural logarithm, a slightly more subtle approach is necessary. Algebraically manipulating $\ln(x+d)$ into a form suitable for Taylor expansion, we obtain:

$$\begin{aligned}\ln(x+d) = \ln\left[x\left(1+\frac{d}{x}\right)\right] &= \ln(x)+\ln\left[1+\frac{d}{x}\right] \\ &= \ln(x)+\left[\frac{d}{x}-\frac{1}{2}\left(\frac{d}{x}\right)^2+\frac{1}{3}\left(\frac{d}{x}\right)^3-\ldots\right] \qquad (3.23) \\ &= \ln(x)+\frac{d}{x} = \left(\ln(x),\frac{1}{x}\right) \qquad (3.24)\end{aligned}$$

This example also illustrates a standard trick which Berz uses to obtain the DA-valued extensions of each of the "elementary" functions: by using their algebraic properties, and the rules of DA, each elementary function may be manipulated into a form which is *polynomial* in the differential part with coefficients given by simple recurrence relations; this form is ideally suited for rapid evaluations on a digital computer, using a generalization of Horner's rule.

The existence of simple and compact algebraic expressions for the elementary functions is, more than any other single feature, what gives DA its tremendous power in numerical applications; there would be little advantage in using DA if the computer still had to evaluate the usual analytic formula for each coefficient in the Taylor expansion to obtain the DA representation of a function. Furthermore, these algebraic representions for the DA-extended elementary functions allow DA to compute derivatives to *arbitrarily high order* (so long as they exist), whereas an algorithm which required the analytic formula of each coefficient would clearly be limited by the maximum order it possessed a formula for.

The second most important feature of DA is that its algebraic operations *automatically* incorporate the "chain rule" of elementary differential calculus into the evaluation of functions. The values of the derivatives of a composite function $f \circ g$ may be obtained by simply evaluating f at $g(x + d)$; this requires essentially the same amount of computational effort as would the evaluation of $f(x + d)$ itself. In this way, one can easily obtain the values of the derivatives of even very complicated functions, by algebraically building them up from the DA representations of the elementary functions.

To motivate this, one should note that while d must be an "infinitesimal", nothing requires it to be the *unit* infinitesimal. It is simply that when the differential part of the argument *is* a unit differential, the differential part of the DA-extended function has a simple interpretation in terms of that function's derivative. However as long as the Taylor coefficients of the function exist through the order of the DA in question (and these coefficients only depend on the *real* part of the function's argument), the differential algebraic extension of that function is well-defined. If the argument instead happens to be the result of evaluating another DA-valued function, a straightforward but tedious calculation shows that the result will be the same as the derivative of the composite function.

To see how this works, I provide the following concrete example. Consider the function $\exp(-ax^2)$; replacing x with $x + d$, we find:

$$\begin{aligned}
\exp\left[-a(x+d)^2\right] &= \exp\left[-ax^2 - 2axd\right] \\
&= \exp(-ax^2)\exp(-2axd) = e^{-ax^2}\left[1 - 2axd\right] \\
&= \left(e^{-ax^2}, -2axe^{-ax^2}\right) \qquad (3.25)
\end{aligned}$$

which is exactly the result one obtains by applying the "chain rule" of differential calculus.

3.3 DA, Functions, and Derivatives

To further understand the relationship between DA and differentiation, I again examine the relationship between DA and nonstandard analysis (NSA).

In NSA, differentiation becomes a purely algebraic procedure; a function f is differentiable if and only if for any arbitrary infinitesimal $\delta \in {}^*R$, the real, or "standard", part of the quotient

$$\frac{f(x+\delta) - f(x)}{\delta} \qquad (3.26)$$

exists and is independent of δ; hence we may compute the derivative of any differentiable function f by simply evaluating (3.26) at some particular value $\delta = \delta_0$ and taking the "standard part":

$$f'(x) = \mathcal{R}\left\{\frac{f(x+\delta_0) - f(x)}{\delta_0}\right\}. \qquad (3.27)$$

The advantage of this procedure is that one need not worry about taking limits.

The quotient (3.26) is undefined when operating within ${}_1D_1$, because the elements ${}^1_1D_1$ of ${}_1D_1$ corresponding to the infinitesimals of ${}^*R$ are not invertible. We can get around this by working with the ***differential*** instead of the derivative.

Let $f : U \to V;\ x \mapsto f(x)$, be a C^1 function from an open interval $U \in R$ to an open interval $V \in R$. We wish to extend f to map ${}_1D_1$ onto itself, for all $x \in U$. Since $(x+d), d = (0,1)$ is only infinitesimally different from x, by continuity we expect the extension $f(x+d)$ also to differ only infinitesimally from $f(x)$:

$$f(x+d) - f(x) = f'(0,1) = f'd \tag{3.28}$$

for some $f' \in R$. But (3.28) is just the usual definition of a "differential"; so the "differential part" of the function $f(x+d)$ is just its derivative. This is confirmed by taking a Taylor expansion of f about the real part of $x+d$ (which is just x):

$$f(x+d) = f(x) + f'(x)d \tag{3.29}$$

where the series terminates at first order in d, because the square and all higher powers of d vanish. We shall see that in a differential algebra of order n, every function taking ${}_nD_v$ onto itself is a ***polynomial of order n*** in the the differentials; following Ritt [Rit50], I shall call such an object a "differential polynomial of order n".

3.4 DA to Higher Orders

The next simplest DA is the second-order algebra in one variable, ${}_2D_1$. This will be a vector space R^3 of ordered ***triples***, (f_0, f_1, f_2), where $f_0, f_1, f_2 \in R$. Vector addition and scalar multiplication are again defined in the usual way; the "vector multiplication" is now defined to be:

$$(f_0, f_1, f_2) \cdot (g_0, g_1, g_2) := (f_0g_0, f_1g_0 + f_0g_1, f_2g_0 + f_1g_1 + f_0g_2) \tag{3.30}$$

The ordering relation is again the "lexicographic ordering" for triples. We will now have not only "infinitesimals" but "superinfinitesimals" as well, because for any $\rho, \delta, \epsilon \in R_+$, we find:

$$(\rho,0,0) > (0,\delta,0) > (0,0,\epsilon) > (0,0,0) \tag{3.31}$$

The multiplicative neutral of ${}_2D_1$ is now $(1,0,0)$, while powers of the unit infinitesimal $d := (0,1,0)$ again form a basis for ${}_2D_1$:

$$\begin{aligned} d^0 &:= (1,0,0) && (3.32)\\ d^1 &= (0,1,0) && (3.33)\\ d^2 &= (0,0,1) && (3.34)\\ d^n &\equiv (0,0,0), \quad (n \geq 3) && (3.35) \end{aligned}$$

Note that in ${}_2D_1$, d is now a "cube root" of zero, rather than a "square root".

The multiplicative inverse is given by:

$$(a,b,c)^{-1} := \left(\frac{1}{a}, -\frac{b}{a^2}, \frac{b^2-ac}{a^3}\right), \quad \forall a \neq 0 \tag{3.36}$$

In addition to the "real" and "differential" parts, I now have a third component to worry about. I introduce the following "order projectors" $\mathcal{P}_k\{\cdot\}$:

$$\begin{aligned} \mathcal{P}_0(a,b,c) &= (a,0,0) = a, && (3.37)\\ \mathcal{P}_1(a,b,c) &= (0,b,0) = b\,d, && (3.38)\\ \mathcal{P}_2(a,b,c) &= (0,0,c) = c\,d^2 && (3.39) \end{aligned}$$

I shall also define projections over ranges of orders, with (hopefully self-explanatory) notations such as $\mathcal{P}_n\{\cdot\}$, $\mathcal{P}_{<n}\{\cdot\}$, and $\mathcal{P}_{>m,<n}\{\cdot\}$. I shall continue to write $\mathcal{R}\{\cdot\}$ for $\mathcal{P}_0\{\cdot\}$; but $\mathcal{D}\{\cdot\}$ shall mean $\mathcal{P}_{>0}\{\cdot\}$, as this projection is particularly important to DA-valued function theory. These definitions of $\mathcal{R}\{\cdot\}$ and $\mathcal{D}\{\cdot\}$ are consistent with my previous use.

Functions on the reals are again extended to ${}_2D_1$ by Taylor expansion about the real part:

$$f(x+d) = f(x) + f'(x)d + \frac{1}{2!}f''(x)d^2 \tag{3.40}$$

where the series again terminates because of the nilpotency of d. From (3.40), and the properties of d, we make the following identifications:

$$f_0 \equiv f(x), \;\; f_1 \equiv f'(x), \;\; f_2 \equiv \frac{1}{2!}f''(x). \tag{3.41}$$

To verify this, consider powers of $(x + d)$ again; we find:

$$\begin{aligned}(x+d)^n &= x^n + nx^{n-1}d + \frac{n(n-1)}{2!}x^{n-2}d^2 + \frac{n(n-1)(n-3)}{3!}x^{n-3}d^3 + \ldots \\ &= \left(x^n,\, nx^{n-1},\, \frac{1}{2!}n(n-1)x^{n-1}\right).\end{aligned} \tag{3.42}$$

Again, the series terminates because of the nilpotency of d, yield the first and second derivatives (up to the purely numerical factor 1/2!). I shall not bother to work through any more examples for ${}_2D_1$, as it would be a straightforward excercise in which nothing new would be learned. The extension to include derivatives through order n, or ${}_nD_1$, is also straightforward.

3.5 DA in Several Variables

We may also also consider algbras with v independent variables, or ${}_nD_v$. In doing so I will also introduce the last of the new notation required in this dissertation. My example will be ${}_2D_2$.

Vectors of the differential algebra ${}_2D_2$ are elements of R^6 (a vector in a general DA is an element of R^N, where N is given by the number of monomials ${}_nN_v$ given in equation (2.7)). Vector addition and scalar multiplication are again defined as usual; the vector product is now:

$$(f_0;\; f_x, f_y;\; f_{xx}, f_{xy}, f_{yy}) \cdot (g_0;\; g_x, g_y;\; g_{xx}, g_{xy}, g_{yy}) \tag{3.43}$$

$$:= \; (f_0 g_0;\ f_x g_0 + f_0 g_x,\ f_y g_0 + f_0 g_y;$$
$$f_{xx} g_0 + f_x g_x + f_0 g_{xx},\ f_{xy} g_0 + f_x g_y + f_y g_x + f_0 g_{xy},\ f_{yy} g_0 + f_y g_y + f_0 g_{yy})$$

Here, subscripts label the components, with semicolons separating groups having common order.

We now have *two* "unit differentials": $dx := (0;1,0;0,0,0)$, and $dy := (0;0,1;0,0,0)$ in the x and y "directions", respectively. Using (3.43), one can easily show that the set of DA-valued monomials $dx^i dy^j$, where $0 \leq i+j \leq 2$, form a basis for ${}_2D_2$, and vanish for all $i+j > 2$. It will also be convenient to define $Dx := x \cdot \mathbf{1} + dx$, and $Dy := y \cdot \mathbf{1} + dy$; in subsequent expressions I shall usually omit the factor $\mathbf{1} := (1;\, 0,0;\, 0,0,0)$ from such expressions, as its presence will be clear from context. The prefix "d" and "D" notations should *not* be thought of as operators, but as analogous to the "vector" symbol $\vec{x}$: they denote that the compound symbols dx and Dx are elements of a differential algebra.

I also use the following "multi-index" notation; define:

$$\begin{aligned}
\langle i \rangle &:= \langle i_1, i_2, \ldots, i_v \rangle, \quad 0 \leq i_k \leq n, \\
|\langle i \rangle| &:= i_1 + i_2 + \ldots + i_v\, , |\langle i \rangle| \leq n, \\
\langle i \rangle! &:= i_1! \cdot i_2! \cdot \ldots \cdot i_v!\, , \\
x^{\langle i \rangle} &:= {x_1}^{i_1} \cdot {x_2}^{i_2} \cdot \ldots \cdot {x_v}^{i_v}, \\
\partial_{\langle i \rangle} &:= \frac{\partial^{|\langle i \rangle|}}{\partial {x_1}^{i_1} \partial {x_2}^{i_2} \cdots \partial {x_v}^{i_v}}, \\
\langle 1_k \rangle &:= \langle \underbrace{0, \ldots 0,}_{(k-1)} \underbrace{\ 1\ }_{k}, \underbrace{0, \ldots 0}_{(v-k)} \rangle
\end{aligned}$$

Then the Taylor expansion of a general function of v variables can be written compactly as:

$$f(x + \Delta x) = \sum_{|I|=0}^{\infty} \frac{1}{\langle I \rangle!} \partial_{\langle I \rangle} f(x)\, \Delta x^{\langle I \rangle}. \tag{3.44}$$

By definition, the ${}_nD_v$-valued extension of $f(x)$ is therefore:

$$Df := f(Dx) = f(x + dx) = \sum_{|I|=0}^{n} \frac{1}{\langle I \rangle!} \partial_{\langle I \rangle} f(x)\, dx^{\langle I \rangle}; \tag{3.45}$$

Comparing like components, we find:

$$f_{\langle I \rangle} = \frac{1}{\langle I \rangle !} \partial_{\langle I \rangle} f(\boldsymbol{x}). \tag{3.46}$$

Invoking the "summation convention" over repeated multi-indices, (3.45) takes on the extremely compact form $Df = f_{\langle I \rangle} d\boldsymbol{x}^{\langle I \rangle}$. Note that at no point did I need to assume that $D\boldsymbol{x} = (\boldsymbol{x}, 1, 0, 0, \ldots)$; it could have been any element of any DA. Note also the importance of the projection $d\boldsymbol{x} = \mathcal{D}\{D\boldsymbol{x}\}$.

The rest of the developement of ${}_nD_v$ very closely follows that of ${}_nD_1$.

3.6 Gradings, Filters, Ideals, and Projections

I shall now show that the order projectors introduced in §3.4 induce natural ***gradings***, ***filters***, and ***ideals*** [ON79, ON82] on ${}_nD_1$.

An algebra A is said to be *graded* if, for some set of integers I, it is a direct sum of subsets:

$$A \equiv \bigoplus_{i \in I} A_i, \tag{3.47}$$

$$A_i A_j \subseteq A_{i+j}, \quad \forall\, i, j, (i+j) \in I. \tag{3.48}$$

Since the order projectors obey the completeness relation, $Df = \sum_{i=0}^{n} \mathcal{P}_i\{Df\}, \forall\ Df \in {}_nD_v$, and by the definition of the multiplication law, $\mathcal{P}_k\{Df \cdot Dg\} = \sum_{i=0}^{k} \mathcal{P}_i\{Df\} \cdot \mathcal{P}_{k-i}\{Dg\}$, it follows that every DA is graded over the nonnegative integers by the order projectors, $\mathcal{P}_i\{\cdot\}$.

An algebra A is said to be ***ascending filtered*** if for every non-negative integer i, there is a subset $A_{(i)}$ such that:

$$A_{(i)} \subseteq A_{(j)}, \quad \forall\, i < j; \tag{3.49}$$

$$A = \bigcup_i A_{(i)}; \tag{3.50}$$

$$A_{(i)} A_{(j)} \subseteq A_{(i+j)} \tag{3.51}$$

If A is graded, then $A_{(j)} = \bigoplus_{i<j} A_i$ defines a natural filtering of A. Since ${}_nD_1$ is graded by $\mathcal{P}_i\{\cdot\}$, it is therefore filtered by $\mathcal{P}_{<i}\{\cdot\} \equiv \sum_{j=0}^{i-1} \mathcal{P}_j\{\cdot\}$.

A subalgebra A' is said to be a *right ideal* if $A'a \subseteq A'$, $\forall\, a \in A$, and a *left ideal* if $aA' \subseteq A'$, $\forall\, a \in A$. A subalgebra which is both a right and left ideal is said to be a *bilateral ideal*, or simply, "an ideal". If an algebra is graded, then ${}^iA := \bigoplus_{j \geq i} A_j$ defines a natural set of ideals of A. Since ${}_nD_1$ is graded by $\mathcal{P}_i\{\cdot\}$, it therefore follows that the subset

$$ {}^i_nD_1 := \mathcal{P}_{\geq i}\{{}_nD_1\} = \sum_{j=i}^{n} \mathcal{P}_j\{{}_nD_1\} \tag{3.52}$$

of ${}_nD_v$ is an ideal. Since the DA product is commutative, there is no distinction between left and right ideals. Two important properties of ${}^i_nD_v$ which I shall need later are:

$$ \mathcal{P}_{<i}\{{}^i_nD_v\} = 0, \tag{3.53}$$

$$ {}^i_nD_v\, {}^j_nD_v \supseteq {}^{i+j}_{n}D_v. \tag{3.54}$$

I shall have need of the above grading, filtering, and ideal properties of differential algebras when I treat the bounds on solutions of the HJ/DA equation.

3.7 Norms on Differential Algebras

Since the elements of each DA form a Euclidian vector space, every definition of a norm on that space induces a corresponding norm on the DA. An example would be the family of "p-norms" defined by:

$$ \|Df\|_p := \left[\sum_{\langle I\rangle} |f_{\langle I\rangle}|^p\right]^{\frac{1}{p}} \tag{3.55}$$

The two norms I shall be most concerned with will be "one-norm" norm given in [Ber89b]:

$$ \|Df\|_1 = \sum_{\langle I\rangle} |f_{\langle I\rangle}|, \tag{3.56}$$

and the "infinity", or "max" norm:

$$\|Df\|_\infty = \sup_{\langle I\rangle} |f_{\langle I\rangle}|. \tag{3.57}$$

It is a straightforward excercise to show that both of these are indeed norms, for they satisfy:

$$\begin{aligned} \|Df\| &= 0 && \text{iff } Df = 0, && (3.58)\\ \|aDf\| &= |a|\,\|Df\|, && \forall\, a \in R, && (3.59) \end{aligned}$$

and the triangle inequality:

$$\|Df + Dg\| \le \|Df\| + \|Dg\|; \tag{3.60}$$

An analogous inequality holds for *products*:

$$\|Df \cdot Dg\|_p \le {}_nL_{v,p}\, \|Df\|_p \cdot \|Dg\|_p, \tag{3.61}$$

where ${}_nL_{v,p}$ is a constant depending on the order and number of variables of the DA, n, v, and also on p.[2] For $p = 1$ (the "one-norm"), ${}_nL_{v,1} = 1$. The optimum value of ${}_nL_{v,p}$ has not yet been determined for $p \neq 1$; however an upper bound of ${}_nL_{v,\infty} \le ((n/v)+1)^v$ has been found for $p = \infty$ (the "max-norm").[3] I will make use of both of these norms in the verification and convergence studies of Chapter 6. Since I will also be interested in how errors depend on the order, I shall also make use of the *seminorms* defined by:

$$\|Df\|_{p,k} := \|\mathcal{P}_k\{Df\}\|_p; \tag{3.62}$$

(3.62) is a seminorm, rather than a norm, because of the presence of the projector $\mathcal{P}_k$: when $\|Df\|_{n,k}$ vanishes, it does not necessarily follow that Df does.

3.8 The DAFOR Extension of FORTRAN

Berz has obtained the DA representations of all the elementary functions (plus a few more, such as the error-function and some other special functions useful in charged-particle optics),

[2] I thank R. F. Streater for pointing out the need to include the constant ${}_nL_{v,p}$ for $p \neq 1$.

[3] Paul F. Zweifel (private communication).

and implemented them in a library of FORTRAN subroutines. This Library also contains routines for the basic arithmetic operations, various utility routines, and several powerful analysis routines that allow one to extract derivatives, perform changes of variables (*e.g.* perform composition of maps), and invert or partially invert changes of variables (*e.g.* invert a map, perform a Legendre transformation) by an application of the implicit function theorem. He has also written the DAFOR precompiler, which parses FORTRAN expressions flagged as containing DA variables, and automatically inserts the appropriate subroutine calls into a FORTRAN program.

With these two tools, it is now possible to compute the value of the *analytical* derivatives to as high an order as one is willing to pay for, for any function which can be represented as a FORTRAN algorithm, which is to say, almost any function at all. In most cases, this may be done using "blind" conversion: one simply declares the variables one wants to evaluate in DA to be of type "DA", and flags all expressions containing DA variables for precompilation using special "comment" cards; the precompiler does the rest. The result is a FORTRAN subroutine capable of evaluating the desired derivatives for any value of the subroutine's parameters. This process is almost completely user-transparent, with only a minimal amount of knowledge about the actual details of DA being required for most applications.

Chapter 4

Perturbative Dynamics and Optics

Having armed the reader with the mathematical tools I will be using, I now turn to the physical framework in which the research for this dissertation was first conceived: Hamilton-Jacobi theory of dynamic systems, as applied to perturbative optics.

4.1 Hamiltonian Optics and the Hamilton-Jacobi Equation

Hamilton [ConSyn31] was the first to show that the trajectory of every extremal ray of light passing through a given optical system may be obtained from a single function, which he called a "characteristic function". In so doing, Hamilton for the first time provided a physical justification for Fermat's principle of "least time".

Hamilton did this by showing that the stationarity of the "optical length" under small variations implied that the variation of the optical length must be an *exact differential*. Therefore a function giving the optical length of the extremal ray in terms of its endpoints must exist.[1] Furthermore, the gradient of this function with respect to the initial and final positions gives the initial and final directions of the extremal ray, respectively. He also showed that the roles of either or both positions in the "point" characteristic function could be interchanged with their corresponding directions via Legendre transformation, to yield "mixed" and "angle" characteristic functions.

Hamilton later went on to show that the trajectories of a time-independent mechanical system were also derivable from a single function, which he this time called a "principal function". He did this by showing the variation in the *action integral* along the extremal trajectory beteween two points in configuration space was also an exact diffential, thereby legitimizing Maupertius' principle of "least action" as well. he found that the gradient of the principal function corresponded to what we now call the "generalized" or "canonical" momentum of Langrangian mechanics.

Ten years latter, Jacobi [Jac63], building on Hamilton's work, showed that associated with every variational principle was a scalar field satisfiying a first-order partial differential equation of the form:

$$F(\boldsymbol{q},\, \partial_{\boldsymbol{q}} S;\; \partial_t S) = 0 \tag{4.1}$$

and *vice-versa*; the study of such first-order equations now bears the name "Hamilton-Jacobi theory". Hamilton-Jacobi theory has long been regarded as an elegent tool for proving theorems about variational problems[2] but of limited practical value, since in general (4.1) is a *nonlinear partial differential equation*, and is therefore rather difficult to solve except in special cases.

While Hamilton's work on mechanics became the centerpiece of perturbative celestial mechanics, his work in optics has largely lain fallow until this century. Indeed, Hamilto-

[1]Assuming, of course, that there is only a single ray connecting them. Hamilton's "characteristic functions" become multiple-valued if there is more than one ray connecting two points; this is closely connected to the existence of *caustic surfaces*, which are discussed in §5.3.1.

[2]Indeed, Rund [Run66], for example, considers HJ-theory the *only* rigorous approach to variational calculus.

nian optics was in effect reinvented in denatured form as the "eikonal approximation" to Maxwell's [BrnWlf70] or Schrödinger's [Gol80, pp. 487–492] equations, and is usually still so treated, although the works of J. L. Synge on the geometrical optics of light [Syn37, Syn51] and "De Broglie waves" [Syn53] have shown that Hamilton's methods are quite capable of standing on their own.

4.2 The Hamilton-Jacobi Equation; Hamilton's Principle and Characteristic Functions

In physics, the Hamilton-Jacobi equation (HJ equation) is usually introduced within the framework of *canonical transformation theory.* A change of variables $(\boldsymbol{q},\boldsymbol{p}) \rightarrow (\boldsymbol{Q},\boldsymbol{P})$ is called "canonical" if the new Lagrangian differs from the old by at most an *exact differential*:

$$P_i\dot{Q}^i - K(\boldsymbol{Q},\boldsymbol{P};t) = p_i\dot{q}^i - H(\boldsymbol{q},\boldsymbol{p};t) - \frac{d}{dt}\Lambda(\boldsymbol{q},\boldsymbol{p};t) \tag{4.2}$$

because then the EOMs for the $(\boldsymbol{Q},\boldsymbol{P})$ also have the canonical form (1.3), but with $H(\boldsymbol{q},\boldsymbol{p};t)$ replaced by the new Hamiltonian $K(\boldsymbol{Q},\boldsymbol{P};t)$. A case of particular interest is whether K can be made to vanish identically; for then (1.3) would imply that the new coordinates are all constant of the motion, and integrating them becomes a trivial task.[3] One simple way of ensuring this (although by no means the most general) is to assume that p_i may be written as the gradient with respect to q^i of some function $S(\boldsymbol{q},\boldsymbol{\alpha};t)$, where the $\boldsymbol{\alpha}$ denote a set of n constants parameterizing the p_i. Since the left hand side of (4.2) vanishes by hypothesis, if I also set Λ equal to S, I am left with

$$0 = \frac{\partial S}{\partial q^i}\dot{q}^i - H\left(\boldsymbol{q}, \frac{\partial S}{\partial \boldsymbol{q}}; t\right) - \left[\frac{\partial S}{\partial t} + \frac{\partial S}{\partial q^i}\dot{q}^i\right],$$

or

$$H\left(\boldsymbol{q}, \frac{\partial S}{\partial \boldsymbol{q}}; t\right) + \frac{\partial S}{\partial t} = 0. \tag{4.3}$$

[3]This is actually a more restrictive definition than necessary; Arnol'd [Arn88, p.260], for example, requires only that K be a function of the Q alone, in which case the Q are constants, while the P are linear functions of time. But this is a trivial generalization, as it is *always* possible to eliminate K entirely.

Equation (4.3) is just a special form of (4.1); however I shall refer to (4.3) as "the" Hamilton-Jacobi equation. A function S of the assumed type that satisfies (4.3) is called a "Hamilton's principal function"; Carathèodory [Car65] has shown that solutions to (4.3) of the assumed type always exist and are unique, at least in some neighborhood about any given point $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ in phase-space.

Since the $\boldsymbol{\alpha}$'s are as-yet-unspecified parameters, one possibility would be to take them to be just the n new (constant) momenta; one can then show from canonical transformation theory that the derivatives of S with respect to the $\boldsymbol{P}$'s are just the new (constant) $\boldsymbol{Q}$'s:

$$p_i = \frac{\partial S}{\partial q^i}, \qquad Q^i = \frac{\partial S}{\partial P_i}.$$

In fact, I can even take $(\boldsymbol{Q}, \boldsymbol{P})$ to be the *initial* positions and momenta, $(\boldsymbol{q}_1, \boldsymbol{p}_1)$, (which do determine the trajectory, and certainly are constants!). In finding S I have therefore in principle found a solution to the initial value problem. The rub, however, is that the above equations are *mixed*: they define the $\boldsymbol{q}_1$ and $\boldsymbol{p}_2$ in terms of the $\boldsymbol{q}_2$ and $\boldsymbol{p}_1$ and therefore provide only an *implicit* solution to the initial value problem.

One can show that $S(\boldsymbol{q}, \boldsymbol{\alpha}; t)$ is the change in *action* along each *extremal* trajectory; that is, formally one can show that, for example:

$$\begin{aligned} S(\boldsymbol{q}, \boldsymbol{\alpha}; t_2) &= \int_{t_1}^{t_2} \frac{dS}{dt}\, dt \\ &= \int_{t_1}^{t_2} \left[\frac{\partial S}{\partial q^k}\frac{dq^k}{dt} + \frac{\partial S}{\partial t}\right] dt \\ &= \int_{t_1}^{t_2} \left[p_k \dot{q}^k - H\right] dt = \int_{t_1}^{t_2} L\, dt \end{aligned} \tag{4.4}$$

(up to a constant of integration) where the integral is taken along the extremal trajectory connecting $\boldsymbol{q}_1$ with $\boldsymbol{q}_2$. However (4.4) is not especially useful for calculating S, since it is precisely this extremal we desire to find. An exception to this is when we desire a *perturbative* expression for S about an RT, in which case one may develop an expansion of S in $\delta\boldsymbol{q}_1$ and $\delta\boldsymbol{q}_2$ by integrating along the RT. The "eikonal method" described in §4.3 below is one approach to this; the method of this dissertation is another.

If the Hamiltonian is independent of time, one can remove the time dependence by "separation of variables"; writing $S(\boldsymbol{q},\boldsymbol{\alpha};t) = W(\boldsymbol{q},\boldsymbol{\alpha};E) - Et$, where E is an arbitrary constant, one reduces (4.3) to the "time-independent HJ (TI-HJ) equation":

$$H(\boldsymbol{q},\partial_{\boldsymbol{q}}W) - E = 0. \tag{4.5}$$

The above equation is still of the form (4.1); a function W which satisfies it is called a "Hamilton's characteristic function".

Note that while the terms "Hamilton's principal function" and "Hamilton's characteristic function" are still in use, they are no longer what Hamilton meant when *he* used them. In his works on both optics and dynamics, Hamilton worked with *pairs* of sets of equations having the forms:

$$H(\boldsymbol{q}_1, -\frac{\partial W}{\partial \boldsymbol{q}_1}) = E, \qquad -\frac{\partial W(\boldsymbol{q}_1,\boldsymbol{q}_2)}{\partial q_1^i} = p_{1i} \tag{4.6a, b}$$

and

$$H(\boldsymbol{q}_2, \frac{\partial W}{\partial \boldsymbol{q}_2}) = E, \qquad \frac{\partial W(\boldsymbol{q}_1,\boldsymbol{q}_2)}{\partial q_2^i} = p_{2i} \tag{4.7a, b}$$

These equations are overdetermined, since they provide $2n+2$ relations between only $2n$ unknowns; if one selects $n-1$ equations each from (4.6) and (4.7), the remaining pair of equations are satisfied identically. Jacobi showed that equations (4.6) and (4.7) are redundant: if, for example, one solves (4.6a), taking (4.7b) to define n constants of integration, then (4.6b) and (4.7a) are satisfied identically. What Hamilton referred to as a *characteristic* function is now referred to as an "eikonal", while the "Hamilton's principal function" $S(\boldsymbol{q},\boldsymbol{\alpha};t)$, is actually *Jacobi's* "complete integral" to the Hamiltonian problem.

4.3 Hamilton's Characteristic Function and the "Eikonal Method"

The "eikonal method" was developed in papers by Glaser, also Sturrock, *circa* 1930–1950 [Gla33, Stu52]. It appears to have been primarily developed for "hand calculations", perhaps aided by some sort of analog computer. These papers are in turn based on traditional optics methods using Hamilton's characteristic functions developed in Bruns's paper of 1892 [Bru95] (for a recent review, see Rose [Ros87], and references therein).

The eikonal method is applicable to time-independent systems only; it is essentially equivalent to a perturbative treatment of the TI-HJ equation. A reference trajectory is chosen, and the n coordinates $\boldsymbol{q}$ are reparameterized in terms of a longitudinal variable, z (which is usually taken to be the arc-length along the RT), and a set of $(n-1)$ variables $\boldsymbol{w}$ parameterizing a set of nonintersecting surfaces transverse to the RT (when $n = 3$, the two $\boldsymbol{w}$'s are often represented by a single complex number). When parameterized in terms of z and $\boldsymbol{w}$, Bruns [Ros87, Bru95] called the characteristic function W an "eikonal".[4] In more recent treatments, rather than starting from the TI-HJ equation most authors follow a laborious and roundabout derivation beginning with the variational form of Hamilton's principle, which they justify by appealing to quantum mechanics. The perturbation expansion of the variation is developed by introducing an artificial "ordering parameter", and expressing deviations from the RT in terms of "Lagrange invariants" and "paraxial rays". The rays may be chosen as the "principal rays" of the *linearized* system, closely analogous to the method of "variation of parameters" in the theory of ODEs, which is useful in enforcing the desired boundary conditions on the action. Applying a method of "successive approximations" directly to the variational principle, these authors obtain integral expressions for the perturbative action of the system, from which one could in principle extract the aberration coefficients or other quantities of interest.

The advantage of the "eikonal method" is that it provides an iterative approach for obtaining *integral* expressions for the perturbative characteristic function, using solutions to the *linearized* system. For many numerical purposes, integral expressions have better stability properties, especially for iterative solutions and "two-point" boundary-value problems (of which the eikonal method is an example).

[4]From the Greek word $\epsilon\iota\kappa\omega\nu$ meaning "image"; this is also the root for the English word "icon", which refers to, among other things, a type of bas-relief religious symbol used by the Eastern Orthodox Faith.

The disadvantage is that, besides being limited to time-independent systems, the eikonal method leads to implicit relations between the initial and final variables in terms of cumbersome integrals with obscure physical meaning. Such implicit relations will result whenever generating functions are used, as mentioned in the previous section; It is difficult to get a "feel" for the system from such implicit relations.[5] This may account for the fact that, despite a claim by Rose that the eikonal method leads to equations well suited for numerical methods [Ros87], this method appears to have largely fallen into disuse.

4.4 Perturbation Theory, Jets, and DA

I stated in §1.7 that the natural geometric framework for perturbation methods was the theory of *jet bundles and prolongations*. Here, I shall briefly describe the concept of a "jet", and show how it relates to perturbation methods, and also DA.

In the modern coordinate-free approach to differential geometry, a *tangent vector* at point P is thought of as an *equivalence class of parameterized curves* all of which have the same derivative at P; in other words, every curve in the equivalence class is tangent to a given line through P, and varying at the same rate with respect to its parameter. This modern definition provides a precise statement of the concept of a vector as "a direction and magnitude".

The theory of jets generalizes the concept of tangent vectors to higher orders of contact ([Bur85, p.107]; [SW86, chap. 6]; see also [Olv86] and is important in studying the symmetries of systems of differential equations. A "1-jet" is just the usual "line-element contact bundle", while a "2-jet" represents an equivalence class of curves whose first and *second* derivatives all agree at a given point; one can go on to consider higher and higher "de-

[5]In principle, this objection applies to my method as well. However, generating functions are only one method of representing maps, and the "partial inversion" routine in the DA package allows one to convert to alternative, more physically useful or transparent representations.

grees of tangency", pinning down more and more coefficients in the Taylor expansion of the parameterized curves.

The connection between perturbation methods and jets comes in when one treats the perturbation parameter ϵ as an "additional coordinate" having trivial dynamics. Consider a system of the form:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{f}(\boldsymbol{\xi}, t; \epsilon), \qquad \dot{\epsilon} = 0, \tag{4.8}$$

with $\boldsymbol{\xi}, \boldsymbol{f} \in M$, $\epsilon \in I$, where I is an interval of R containing 0. Now consider at $t = t_0$ a set of ϵ-dependent initial conditions $\boldsymbol{\xi}_0 : I \to M$; $\epsilon \mapsto \xi_0(\epsilon)$. *i.e.* we let ϵ parameterize a "deformation" of the initial conditions, as well as the dynamics; a "perturbation" results when ϵ is considered to be infinitesimal.

While one would usually view $\boldsymbol{\xi}_0(\epsilon)$ as a parameterized curve on M, one can also view it as a parameterized curve $(\epsilon, \boldsymbol{\xi}_0)$ in the extended manifold $I{\times}M$. Omohundro [Omo86, pp.96–107] calls a such curve a ***path***, and the set of all paths, ***path space***; the flow $\mathcal{F}_{t,t_0}$ of (4.8) will take the path $(\epsilon, \boldsymbol{\xi}_0(\epsilon))$ into the parameterized set of paths $\mathcal{F}_{t,t_0} : I \times M \to I \times M$; $(\epsilon, \boldsymbol{\xi}_0(\epsilon)) \mapsto (\epsilon, \boldsymbol{\xi}(\epsilon, t))$, the perturbed path of the system. Omohundro considers all points on a given path to be members of an equivalence class; a natural projection is defined by associating each point on the path with the unperturbed state at $\epsilon = 0$. The arbitrariness of the initial path $(\epsilon, \xi_0(\epsilon))$ corresponds to the "gauge freedom" discussed in §1.7. One should not interpret this, however, to mean that "all paths are physically equivalent", any more than one would interpret the general coordinate covariance of general relativity to mean that "all metrics are physically equivalent". Each path represents a distinct "point" in path space, even though many paths project to the same point of M. The choice of the path determines, in part, the nature of the perturbation, by specifying how the initial conditions are to be deformed.

Omohundro now looks at the derivatives $\partial^n \boldsymbol{\xi}(t, \epsilon)/\partial \epsilon^n|_{\epsilon=0}$, and finds that these derivatives provide a particular representation for a jet. Furthermore, these derivatives are just the coefficients in the perturbation expansion (1.11). *Therefore, the natural geometric framework* *for describing perturbation "theory" is the theory of jets.* The fact that ϵ is set to zero after taking the derivative supports Omohundro's contention that perturbation "theory" is really telling us something about the response of the *unperturbed* system to the perturbation; alternatively, the perturbation should always be thought of as *infinitesimally* small.

At this point, on recalling that derivatives are just the sort of thing that DA calculates, one should immediatly realize that by letting $\epsilon \to d\epsilon$, and evaluating (4.8) in DA, *one automatically obtains a set of ODEs for the numerical values of the perturbation coefficients.* Furthermore, even if (4.8) cannot be integrated analytically, I can still find approximate values of the perturbation coefficients by simply performing a *numerical* integration in DA. So using DA, I can obtain the numerical values of the perturbation coefficients with only a little more (human) labor than would be required to numerically integrate the equations of motion themselves.

Chapter 5

DA Methods for Approximate Solution of the Hamilton-Jacobi Equation

In this chapter, I show how DA my be used to convert the Hamilton-Jacobi equation, which is a nonlinear, partial differential equation, into a system of ODE's for the perturbative expansion of Hamilton's principle function about some reference trajectory.

In this chapter, the following index conventions are invoked: lower-case latin indices run from 1 to d, where d is the number of *pairs* of canonically conjugate variables; lower-case greek indices run from 1 to $2d$; "multi-indices" run over the set of all monomials of order less than or equal to n, over the natural index range associated with the kernel letter. All DA-valued quantities shall be assumed to be elements of ${}_nD_v$, $v \geq 2d$ (I am allowing for the possibility that there might be more variables than just the "p's and q's", since one might possibly wish to extract the dependence of the map on some set of non-dynamical parameters $\boldsymbol{\delta}$). The natural basis for expanding DA-valued quantities shall be taken to be the $d\zeta$ (plus the $d\delta$, if present). All partial derivatives shall be taken with respect to the ζ's, unless otherwise specified.

5.1 Equivalence of the DA-Valued HJ Equation and a System of ODEs

The proof is almost trivial. Let $\xi^\mu := (q^i, p_i)$, $p_i := \partial S(q,\alpha;t)/\partial q^i$, and $\zeta^\mu := (q^i, \alpha_i)$. I promote q and α to DA-valued variables; the HJ equation then becomes the "HJ/DA equation":

$$\frac{\partial}{\partial t} S(D\zeta;t) = -H(\xi(D\zeta);t) \tag{5.1}$$

Since the powers of $d\zeta^\mu$ form a basis for the DA, by definition I have:

$$\begin{aligned} S(D\zeta;t) &= S(\zeta+d\zeta;t) = \sum_{|I|=0}^{n} \frac{1}{\langle i\rangle!} \partial_{\langle i\rangle} S(\zeta;t)\, d\zeta^{\langle i\rangle} \\ &= S_{\langle i\rangle}(t)\, d\zeta^{\langle i\rangle} \end{aligned} \tag{5.2}$$

and

$$\begin{aligned} H(\xi(D\zeta);t) &= H(\xi(\zeta+d\zeta);t) = \sum_{|I|=0}^{n} \frac{1}{\langle i\rangle!} \partial_{\langle i\rangle} H(\xi(\zeta);t)\, d\zeta^{\langle i\rangle} \\ &= H_{\langle i\rangle}(t)\, d\zeta^{\langle i\rangle} \end{aligned} \tag{5.3}$$

where the summation convention has been invoked on the last line of (5.2) and (5.3). I have written the coefficients $S_{\langle i\rangle}$ and $H_{\langle i\rangle}$ as functions of t alone, because they depend by definition on at most the *real* parts of the $D\xi$, which for the moment I shall just consider to be prescribed functions of time. In the next section I will show that the $\xi_\circ(t) := \mathcal{R}\{D\xi(t)\}$ are determined by requiring that certain *closure conditions* be satisfied; the closure conditions are just that the $\xi_\circ(t)$ must solve Hamilton's equations. Since the $\xi_\circ(t)$ define the RT for the HJ/DA equation, this should come as no great surprise.

Since the ζ and t are independent variables, the $\partial d\zeta/\partial t$ vanish. Therefore, $S(D\xi;t)$ depends on time only via the $S_{\langle i\rangle}$, and I may equate like coefficients in (5.1) to reduce it to the following system of ODE's:

$$\dot{S}_{\langle i\rangle}(t) = -H_{\langle i\rangle}(t). \tag{5.4}$$

The above system may be solved by a number of standard methods, once the ***boundary conditions*** (BC's) have been specified; since I shall need several types of solutions, I shall defer the discussion of BC's for each type until the section in which it is developed.

5.2 Closure Conditions

Before I can solve the HJ/DA equation, I must first address a subtle problem: the existence of *closure conditions*. In order to evaluate the right hand side of the HJ/DA equation, I must first evaluate $\partial DS/\partial q^k$. In DA this is essentially just a "shift" or "bookkeeping operation": by the "power rule",

$$\frac{\partial}{\partial q^k} S_{\langle j\rangle} d\boldsymbol{q}^{\langle j\rangle} = j_k S_{\langle j\rangle} d\boldsymbol{q}^{\langle j\rangle - \langle 1_k\rangle}$$

where j_k is the exponent of dx_k, *i.e.*, one multiplies each monomial coefficient by its "k^{th}" exponent, decrements the "k^{th}" exponent by one, then reinserts it in the appropriate "slot" of the DA vector coresponding to its new exponent, discarding any coefficients for which the new i_k is less than zero. Since the DA-library implements DA-vectors as "packed" arrays containing only the non-zero coefficients of DA-vectors in sorted order, plus a pair of arrays of integers encoding the "index vectors" $\langle j\rangle$ of the corresponding coefficients, this can be done "in place" and very rapidly.

However, in an n^{th}-order DA, the value of the $(n+1)^{th}$ order coefficients are unknown. Faced with this absence of information, Berz elected simply to define all the n^{th}-order coefficients of the "derivative" of a DA vector to be zero ("zero-padding"). One sees that the set of operators $\partial/\partial q^i$ act much like "lowering operators" on DA vectors, and that at most $(n+1)$

applications of members of this set to a vector will "annihilate" every vector of an n^{th}-order DA.

Strictly speaking, one should probably view $\partial/\partial q^i$ as being a linear map from ${}_nD_v$ to ${}_{n-1}D_v$. Since we have not defined operations between elements of different DAs, this raises concern as to whether the HJ/DA equation is well defined. Even with the "zero-padding" trick, which formally extends elements of ${}_{n-1}D_v$ to ${}_nD_v$, there is still a problem: I have potentially lost the "closure" propety of DA under truncation, because the "lowering" action of the $\partial/\partial q^i$ now allows errors in the higher-order elements of DS to feed "back" as well as feed "forward"— errors in $\mathcal{P}_n\{DS\}$ will influence $\mathcal{P}_{n-1}\{DS\}$, as well as $\mathcal{P}_{>n}\{DS\}$, which in turn will effect $\mathcal{P}_{n-2}\{DS\}$, and so on.

Fortunately, there is a way around this problem. An explicit formula for the monomials of the product of two DA-vectors f, g can be written in the "multi-index" notation as:

$$(fg)_{\langle\rho\rangle} = \sum_{\substack{\langle\mu\rangle,\langle\nu\rangle \\ \langle\mu\rangle+\langle\nu\rangle=\langle\rho\rangle}} f_{\langle\mu\rangle}g_{\langle\nu\rangle}, \quad \forall\, \langle\rho\rangle : 0 \leq |\langle\rho\rangle| \leq n \tag{5.5}$$

— *i.e.,* one sums over all pairs of multi-indices such that $\langle\mu\rangle+\langle\nu\rangle = \langle\rho\rangle$. Now, let me assume that $\mathcal{P}_n\{g\}$ vanishes. For the "index norm", $|\langle\rho\rangle| = |\langle\mu\rangle| + |\langle\nu\rangle|$ holds. In any DA, $0 \leq |\langle\mu\rangle| \leq n$ for all valid $\langle\mu\rangle$. Therefore, the only element of $f_{\langle\mu\rangle}$ appearing with $\mathcal{P}_n\{g\}$ in the product is its real part, f_0. *But all functions of DA-valued quantities are polynomial in the differential part of their argument.* Therefore, the only terms in (5.3) in which $\mathcal{P}_n\{\partial S/\partial q^i\}$ will appear are the *first order terms* of $H(D\boldsymbol{q}, D\boldsymbol{p};\, t)$ involving the $\boldsymbol{p}$. But the first order terms of H can always be eliminated by the following trivial canonical transformation:

$$\begin{aligned} q^i &= q_0^i(t) + \bar{q}^i, \qquad & p_i &= p_{0i}(t) + \bar{p}_i, \\ \dot{q}_0^i &= \frac{\partial}{\partial p_{0i}} H(\boldsymbol{q}_0, \boldsymbol{p}_0; t), \qquad & \dot{p}_{0i} &= -\frac{\partial}{\partial q_0^i} H(\boldsymbol{q}_0, \boldsymbol{p}_0; t), \end{aligned} \tag{5.6}$$
$$K(\bar{\boldsymbol{q}}, \bar{\boldsymbol{p}}) = H(\boldsymbol{q}_0(t)+\bar{\boldsymbol{q}}, \boldsymbol{p}_0(t)+\bar{\boldsymbol{p}}; t) - \dot{q}_0^i(t)\bar{p}_i + \dot{p}_{0i}(t)\bar{q}^i,$$

which is just a simple time-dependent translation causing the new origin of phase-space to travel along the RT. In other words, the origin of the new coordinates $(\bar{\boldsymbol{q}}, \bar{\boldsymbol{p}})$ is a *fixed point* *of the time-evolution map.* One can easily verify that this transformation is canonical, and that the new Hamiltonian K has no linear terms.[1] Since K has no linear terms, the HJ/DA equation obtained from it will be closed.

5.3 Need for More Than One Type of Solution

The Hamilton-Jacobi function S is an example of a so-called "mixed" generating function, because of its dependence on $2d$ variables, half of which are obtained by selecting one variable from each of the d pairs of "old" positions and momenta, and the other half by selecting one variable from each of the d pairs of "new" positions and momenta. There are thus $2^{2d} = 4^d$ possible combinations altogether.

The "mixed" functions map $2d$-dimensional submanifolds of the product of the "old" and "new" phase-spaces onto the reals: $P_{old} \times P_{new}$ onto R. One can show that such a function generates an identically canonical map $P_{old} \to P_{new}$ by equating (up to a sign) the partial derivative of the generating function with respect to each argument with its canonical conjugate. The most general proof, found in Carathéodory [Car65, Vol. 1,§ 97, pp. 87–90], is somewhat awkward and messy, and makes extensive use of the implicit function theorem; I will not reproduce it here. A simplified version may be found in Goldstein [Gol80, § 9.4, pp. 403–405].

While in principle one can consider any such partition of the "old" and "new" variables, most texts (*e.g.* Goldstein [Gol80, chap. 9]) emphasize (perhaps overly much) the following four classes:

$$F_1(q_2, q_1), \qquad F_2(q_2, p_1)$$
$$F_3(p_2, q_1), \qquad F_4(p_2, p_1)$$

[1]Actually, (5.6) is more restrictive than necessary; it would have been sufficient to eliminate only those linear terms involving the p. However it almost as simple, and far more convienient, to eliminate *all* the linear terms from K, in order to obtain an origin-preserving map.

(Confusingly, I must take q_2 and p_2 to be "old" variables, and q_1 and p_1 to be "new" variables, if I am to maintain my identification of α with p_1.)

While each class of mixed generating functions produce identically canonical transformations, it does not follow that a *given* canonical transformation can be generated by functions from *each* of the four classes, as Goldstein states, but does not explain [Gol80, § 9.1, p. 385]. A given class of generating function may not be suitable to generate a given canonical transformation beacause of the existence of *caustics* and *foci.*

5.3.1 Caustics, Foci, and Singular Generating Functions

Caustics [Arn88, p.448] occur when a continuous family of extremals[2] possesses an *envelope*; when this happens p_1 and q_2 no longer uniquely specify a trajectory, and the *Hessian* of the generating function either vanishes or becomes infinite on some set of points, *e.g.*:

$$\begin{vmatrix} \dfrac{\partial^2 F_2}{\partial q_2 \partial q_2} & \dfrac{\partial^2 F_2}{\partial q_2 \partial p_1} \\ \dfrac{\partial^2 F_2}{\partial p_1 \partial q_2} & \dfrac{\partial^2 F_2}{\partial p_1 \partial p_1} \end{vmatrix} = \{ \, 0 \text{ or } \infty \, \}$$

Caustics can be classified algebraically [Arn88, pp.448–452], and are closely related to *catastrophe theory* [DeW76, DV79].

Focal points [Arn88, p.442] occur, when all members of a family of extremals pass through the same point q in configuration-space but with different momenta, p. If a family of rays from a focus form a caustic, the point on a ray which touches the caustic is said to be *conjugate* to the focus along that ray. A caustic is therefore a set of conjugate points. Formal definitions of caustics and conjugate point are best given in terms of *Jacobi fields.* Jacobi fields are solutions to the *Jacobi equations.* The Jacobi equations are essentially just the linearization of the Euler-Lagrange equations about the extremals; however they are usually derived by considering the *second variation* of the action [DeW76]. Jacobi fields are

[2] *i.e.*, a set of extremals labeled by one or more continuous parameters.

therefore closely related to the second-order part of the solution to the HJ/DA equation, the principle difference being that a Jacobi field is a global object, while $DS(t)$ is only an expansion about a particular ray. When a caustic exists, it will only be possible to find $2d - k$ linearly independent Jacobi fields, with $1 \leq k \leq d$. The integer k is the ***multiplicity*** of the conjugate points forming the caustic.

Caustics and foci are only apparent singularities, being artifacts of "badly" chosen coordinates. It is always possible, for example, to find at least one set of v variables out of the set $(\boldsymbol{q}_2, \boldsymbol{p}_2)$, so that, when taken with with the "old" coordinates $\boldsymbol{q}_1$, the Hessian does not vanish [Arn88, pp. 267–269]; however the existence of such singularities is ubiquitous, and therefore a nuisance.

5.3.2 Classifying Generating Functions by the Images They Can't Represent

In order to get a feel for how and when caustics and foci appear, I provide the following example of a two degree-of-freedom (2-DOF) *linear* system; it is sufficient to consider linear systems because the Hessian and Jacobian matrices on the RT are determined by the linearized dynamics. The generalization to fully nonlinear systems should be obvious.

The transfer map of a 2-DOF linear system, in matrix form, is:

$$\begin{bmatrix} q_2 \\ p_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} q_1 \\ p_1 \end{bmatrix} \tag{5.7}$$

For a 2-DOF system, the symplectic condition is equivalent to the unimodular condition, $\det\{\mathrm{M}\} = ad - bc = 1$. It follows from (5.7) and the symplectic condition that either ad or bc may vanish, but not both at the same time, and that the vanishing of a given element leads to the following type of imaging:

$$\begin{aligned} a = 0 &: \text{parallel-to-point}, & b = 0 &: \text{point-to-point} \\ c = 0 &: \text{parallel-to-parallel}, & d = 0 &: \text{point-to-parallel} \end{aligned} \tag{5.8}$$

The generating functions are:

$$\begin{aligned}
F_1(q_2,q_1) &= \frac{1}{2}\frac{d}{b}q_2^2 - \frac{1}{b}q_2q_1 + \frac{1}{2}\frac{a}{b}q_1^2, & p_1 &= -\frac{\partial F_1}{\partial q_1}, & p_2 &= \frac{\partial F_1}{\partial q_2};\\
F_2(q_2,p_1) &= \frac{1}{2}\frac{c}{a}q_2^2 + \frac{1}{a}q_2p_1 - \frac{1}{2}\frac{b}{a}p_1^2, & q_1 &= \frac{\partial F_2}{\partial p_1}, & p_2 &= \frac{\partial F_2}{\partial q_2};\\
F_3(p_2,q_1) &= -\frac{1}{2}\frac{b}{d}p_2^2 - \frac{1}{d}p_2q_1 + \frac{1}{2}\frac{c}{d}q_1^2, & p_1 &= -\frac{\partial F_3}{\partial q_1}, & q_2 &= -\frac{\partial F_3}{\partial p_2};\\
F_4(p_2,p_1) &= -\frac{1}{2}\frac{a}{c}p_2^2 + \frac{1}{c}p_2p_1 - \frac{1}{2}\frac{d}{c}p_1^2, & q_1 &= \frac{\partial F_4}{\partial p_1}, & q_2 &= -\frac{\partial F_4}{\partial p_2}.
\end{aligned} \tag{5.9}$$

Calculating the Hessians, I find:

$$\begin{aligned}
\Delta(F_1) &= \frac{c}{b}, & \Delta(F_2) &= \frac{d}{a},\\
\Delta(F_3) &= -\frac{a}{d}, & \Delta(F_4) &= -\frac{b}{c}.
\end{aligned} \tag{5.10}$$

Consulting (5.8), one can easily see from (5.10) which types of generating functions are unsuitable for which types of imaging; one can also see that all four types of imaging are covered by either an F_1 or F_2, as claimed.

In fact, it is even possible to restrict oneself only to the class of F_2-type generating functions, composed with the trivially canonical "exchange map" [For84, pp. 5–10]. $(q,p) \mapsto (p,-q)$. One can easily repeat the above argument for $2v$-DOF systems, with a, b, c, and d replaced by $v \times v$ block matrices; the regularity condition will then be that the determinant of the appropriate block not vanish or diverge.

Can one convert from one representation to another? Yes, one can, as long as both representations satisfy the regularity condition. Converting between representations may then be done by a Legendre transformation; one must then perform a partial inversion (to obtain the former independent variables in terms of the new independent variables), followed by a composition (to reexpress the Legendre-transformed generating function in terms of the new independent variables). However, since inversion is the most expensive operation in the DA-package, one would rather not do this very often.

Can one prove that either an F_1 or F_2 cover all possibilities? Unfortunately one cannot; it is easy to imagine an astigmatic optical system which produces point-to-parallel imaging in one plane, and point-to-point imaging in the perpendicular plane. A practical example is a particle storage ring, one characterizes it by its radial and vertical "tunes", ν_r and ν_z, the number of radial and vertical betatron oscillations a particle undergoes in making one turn around the ring. In order to avoid resonant instabilities, ν_r and ν_z should not be rationally related, it i.e., there should not be a triple of integers n, m, p, all of the same sign (one may be zero) such that $n\nu_r + m\nu_z = p$ (in practice, only the lower order resonances are really dangerous). But if ν_r and ν_z are not rationally related, the particle must pass arbitrarily close to every accessible state, *including the one where its radial and vertical betatron phases are zero and $\pi/2$, respectively.* A generating function representing this state would have to be F_2-like in the radial plane, and F_1-like in the vertical plane. If one insists on representing the linear part of the motion by a generating function, it will therefore be necessary to be able to represent each of the 2^d possible cases of "mixed-mixed" generating functions, and be able to interconvert between them. The bookkeeping of one's choice of independent variables can be handled by a $2d$-bit binary number; converting between representations may be done much as before, except that one only performs a partial Legendre transformation. However the additional complexity is somewhat daunting, and one would rather avoid it, if one could. Therefore, after discussing how one obtains F_1 and F_2 type solutions to the HJ/DA equation, I shall discuss how one may avoid the problem of caustic and foci altogether, by separately treating the linear and nonlinear dynamics.

5.4 Procedure for F_2 Solutions

Obtaining F_2-type solutions to the HJ/DA equation is quite straightforward; since F_2 (and also F_3) generators are connected to the identity, I simply choose the DA representation of the identity map for my initial generating function:

$$S(D\bar{q}_2, D\bar{p}_1;\, t=0) = d\bar{p}_{1i}\, d\bar{q}_2^i \tag{5.11}$$

I can compute S using the HJ/DA equation, with H replaced by the transformed Hamiltonian K defined by (5.6).

I can obtain the derivatives I need to compute K from H by simple projection, since $d\boldsymbol{q} = d\bar{\boldsymbol{q}}$, $d\boldsymbol{p} = d\bar{\boldsymbol{p}}$. The complete system of equations is:

$$d\bar{q}_{1i} = D\bar{q}_{1i} := \frac{\partial}{\partial \bar{p}_1^i} DS(t), \tag{5.12}$$

$$d\bar{p}_{2i} = D\bar{p}_{2i} := \frac{\partial}{\partial \bar{q}_2^i} DS(t), \tag{5.13}$$

$$Dq_2^i := q_0^i + d\bar{q}_2^i, \tag{5.14}$$

$$Dp_{2i} := p_{0i} + d\bar{p}_{2i}, \tag{5.15}$$

$$DH(t) := H(D\boldsymbol{q}_2, D\boldsymbol{p}_2; t) \tag{5.16}$$

$$\dot{q}_0^i(t) = \mathcal{R}\left\{\frac{\partial}{\partial \bar{p}_{2i}} DH(t)\right\}, \tag{5.17}$$

$$\dot{p}_{0i}(t) = -\mathcal{R}\left\{\frac{\partial}{\partial \bar{q}_2^i} DH(t)\right\}, \tag{5.18}$$

$$DK(t) := DH(t) - \dot{q}_0^i(t) d\bar{p}_{2i} + \dot{p}_{0i}(t) d\bar{q}_2^i, \tag{5.19}$$

$$D\dot{S}(t) = -DK(t). \tag{5.20}$$

5.5 Procedure for F_1 Solutions

Obtaining F_1-type solutions is unfortunately a bit trickier. While the HJ/DA equation results in a system of ODEs of exactly the same form as in the F_2 case,[3] the initial value problem for an F_1 solution is ill-defined, because an F_1-type function representing the identity does not, strictly speaking, exist. A "weak", or *generalized* F_1-type solution may be defined, which for short times approaches the *free* action. In Cartesian coordinates, for non-relativistic (*i.e.* "T+V") systems and short times, such "weak" solutions are of the

[3]The difference between F_1 and F_2 generating functions, as we are using them, *implicitly* appears in the identification of the *initial* independent variables as the q_1 rather than the p_1, but does not *explicitly* appear in the HJ equation itself.

form:

$$S(\vec{x}_1, \vec{x}_2; \Delta t) \simeq \frac{m}{2\Delta t}(\vec{x}_2 - \vec{x}_1)^2 + \mathcal{O}(\Delta t);$$

I showed an action of this type in §2.3.2, eqn. (2.18), while discussing Channel and Scoval's Hamilton-Jacobi-based symplectic integrator. Since $(\vec{x}_2 - \vec{x}_1) \simeq \vec{v}\Delta t$, for finite initial velocities such "weak" solutions are actually non-singular; nevertheless, I expect difficulties when seeking F_1-solutions to the HJ/DA equation, when treated as an initial value problem.

Where F_1 solutions can be expected to come into their own is for integrating through caustics and foci. Since an F_1 solution is well behaved in regions where an F_2 solution breaks down, one could continue integrating if one could find an F_1 which represents the same map as the F_2. The implicit function theorem guarantees that it is indeed possible to convert between F_2 and F_1 generating functions, so long as their respective Hessians are nonsingular; but this is true by hypothesis. Since the DA package was created with this possibility in mind, Berz provided the *recursive partial-inversion* subroutine `DAPIN` for interchanging the roles of a subset of the dependent and independent variables. Therefore one can continue integrating through caustics and foci by carrying out a partial inversion when the value of the Hessian becomes too large or too small.

5.6 Procedure for "Hybrid" Solutions

While F_1/F_2 conversion provides a mechanism for integrating through caustics or foci, the partial-inversion process is the most computationally expensive operation in the DA package. Since one "hits" a caustic or a focus roughly four times per oscillation in a periodic system, this is an unpleasant prospect, to say the least.

The problem of caustics and foci arises because the chosen *anstaz* for S (*i.e.*, the generating function is either of type F_1 or F_2) imposes artificial restrictions on the accessible domain of $P_1 \times P_2$ which the dynamics does not respect; in geometric terms, the *chart* induced by each particular type of generating function does not cover the entire solution manifold. However, it is important to note that these artificial constraints may be expressed ***entirely in terms of the Hessian***, which in turn is determined by the ***linear part of the map***. Generating functions are simply the wrong representation for a linear map; the appropriate representation is a *matrix*. By extracting out the linear part of the dynamics and explicitly representing it as a matrix, one might hope to eliminate the problem of caustics and foci. Another viewpoint is to observe that, by hypothesis, the *nonlinear* part of perturbative dynamics, by definition being of at least one order higher than the *linear* part, is "small" compared to the linear dynamics. Therefore the nonlinear part of the map is always "near" the identity in some suitable sense, so I can always represent it by an F_2.

I begin with Hamilton's equations in the tensor form (1.8):

$$\dot{\xi}^\mu = J^{\mu\nu}\partial_\nu H(\xi;t)$$

Let me perform a time-dependent affine transformation of the phase-space variables:

$$\xi^\mu = \xi_0^\mu(t) + M^\mu_\alpha(t)\,\tilde{\xi}^\alpha, \qquad \tilde{\xi}^\alpha = M^{-1}{}^\alpha_\mu(t)\,(\xi^\mu - \xi_0^\mu(t)) \tag{5.21}$$

(since the local chart maps $P \to R^d \times R^d \simeq R^{2d}$, this is always possible). I will go somewhat against current mathematical practice by introducing a new symbol $\tilde{H}$ to denote the "old" Hamiltonian expressed in terms of the "new" variables:

$$\tilde{H}(\tilde{\boldsymbol{\xi}};t) := H(\boldsymbol{\xi}_o(t) + \mathbf{M}(t)\cdot\tilde{\boldsymbol{\xi}};\ t). \tag{5.22}$$

Explicitly expand $\tilde{H}$ through second order:

$$\tilde{H}(\tilde{\boldsymbol{\xi}};\ t) = \tilde{h}_0(t) + \tilde{h}_\alpha(t)\,\tilde{\xi}^\alpha + \frac{1}{2}\tilde{h}_{\alpha\beta}(t)\,\tilde{\xi}^\alpha\tilde{\xi}^\beta + K(\tilde{\boldsymbol{\xi}};\ t) \tag{5.23}$$

I shall show below that $\tilde{h}_\alpha(t)$ determines the EOM of the RT, and is solved by $\boldsymbol{\xi}_o(t)$; $\tilde{h}_{\alpha\beta}(t)$ determines the *linearization* of the EOMs about th RT, ans is solved by the "matrix part" of the evolution map, which is $\mathbf{M}(t)$; finally, the "remainder" term $K(\tilde{\xi};t)$ contains all the nonlinearities of the dynamics. From this point onward I shall usually suppress explicit reference to the time dependence of $\boldsymbol{\xi}_o(t)$, $\mathbf{M}(t)$, $\tilde{h}_\alpha(t)$, and $\tilde{h}_{\alpha\beta}(t)$.

Using the "chain rule", I re-express the factor $\partial_\nu H(\xi;t)$ in Hamilton's equations in terms of the "tilde-ed" variables:

$$\begin{aligned}\partial_\nu H(\boldsymbol{\xi};\,t) &= \frac{\partial\tilde{\xi}^\alpha}{\partial\xi^\nu}\frac{\partial}{\partial\tilde{\xi}^\alpha}H(\boldsymbol{\xi}_o+\mathbf{M}\cdot\tilde{\xi};t) = M^{-1}{}^\alpha_\nu\tilde{\partial}_\alpha\tilde{H}(\tilde{\boldsymbol{\xi}};\,t)\\ &= M^{-1}{}^\alpha_\nu\left\{\tilde{h}_\alpha+\tilde{h}_{\alpha\beta}\tilde{\xi}^\beta+\tilde{\partial}_\alpha K(\tilde{\boldsymbol{\xi}};\,t)\right\},\end{aligned} \tag{5.24}$$

where $\tilde{\partial}_\alpha$ is a shorthand notation for $\partial/\partial\tilde{\xi}^\alpha$.

Differentiating (5.21a), substituting (5.24) into (1.8), and equating the two, I find:

$$\dot{\xi}_0^\mu + \dot{M}^\mu_\alpha\tilde{\xi}^\alpha + M^\mu_\alpha\dot{\tilde{\xi}}^\alpha = J^{\mu\nu}M^{-1}{}^\alpha_\nu\left\{\tilde{h}_\alpha + \tilde{h}_{\alpha\beta}\tilde{\xi}^\beta + \tilde{\partial}_\alpha K(\tilde{\boldsymbol{\xi}};t)\right\}. \tag{5.25}$$

Identifying like terms, I get:

$$\dot{\xi}_0^\mu = J^{\mu\nu}M^{-1}{}^\alpha_\nu\tilde{h}_\alpha \tag{5.26}$$

$$\dot{M}^\mu_\beta = J^{\mu\nu}M^{-1}{}^\alpha_\nu\tilde{h}_{\alpha\beta} \tag{5.27}$$

and

$$M^\mu_\alpha\dot{\tilde{\xi}}^\alpha = J^{\mu\nu}M^{-1}{}^\alpha_\nu\tilde{\partial}_\alpha K(\tilde{\boldsymbol{\xi}};\,t) \tag{5.28}$$

Rewriting (5.28) as:

$$\dot{\tilde{\xi}}^\alpha = M^{-1}{}^\alpha_\mu\, J^{\mu\nu}\, M^{-1}{}^\beta_\nu\, \tilde{\partial}_\beta K(\tilde{\boldsymbol{\xi}};\,t) \tag{5.29}$$

I now demand that (5.29) still have the canonical form; then $\mathbf{M}$ must satisfy:

$$M^{-1}{}^\alpha_\mu\, J^{\mu\nu}\, M^{-1}{}^\beta_\nu = J^{\alpha\beta}, \qquad \Longrightarrow \qquad M^\mu_\alpha\, J^{\alpha\beta}\, M^\nu_\beta = J^{\mu\nu}. \tag{5.30}$$

The above is just the tensor form of the ***symplectic condition***, (1.10); furthermore, it follows that:

$$M^{-1}{}^\lambda_\rho = J^{\alpha\beta}M^\lambda_\beta J^{-1}_{\lambda\rho}, \tag{5.31}$$

Putting it all together, I find:

$$\dot{\xi}_0^\mu = M^\mu_\alpha J^{\alpha\beta}\tilde{h}_\beta \tag{5.32}$$

$$\dot{M}^\mu_\gamma = M^\mu_\alpha J^{\alpha\beta}\tilde{h}_{\beta\gamma} \tag{5.33}$$

and, finally,

$$\dot{\tilde{\xi}}^{\alpha} = J^{\alpha\beta}\tilde{\partial}_{\beta}K(\tilde{\boldsymbol{\xi}};\, t) \tag{5.34}$$

The above equations have the following interpretation:

- Equation (5.32) is just the EOM of the RT in "tilde-ed" variables: $\tilde{h}_{\beta} := \tilde{\partial}_{\beta}\tilde{H}|_{\tilde{\xi}=0} = M^{\nu}_{\beta}\partial_{\nu}H|_{\xi=\xi_0}\ , \implies \dot{\xi}^{\mu}_{0} = M^{\mu}_{\alpha}J^{\alpha\beta}M^{\nu}_{\beta}\partial_{\nu}H|_{\xi=\xi_0} = J^{\mu\nu}\partial_{\nu}H|_{\xi=\xi_0}$
- Equation (5.33) is the EOM for the *linear transfer matrix*, describing deviations from the RT due to the *linearized* EOMs: $\tilde{h}_{\beta\gamma} := \tilde{\partial}^2_{\beta\gamma}\tilde{H}|_{\tilde{\xi}=0} = M^{\nu}_{\beta}M^{\lambda}_{\gamma}\partial^2_{\nu\lambda}H|_{\xi=\xi_0}\ ,$ $\implies \dot{M}^{\mu}_{\gamma} = M^{\mu}_{\alpha}J^{\alpha\beta}M^{\nu}_{\beta}\partial^2_{\nu\lambda}H|_{\xi=\xi_0}M^{\lambda}_{\gamma} = J^{\mu\nu}\partial^2_{\nu\lambda}H|_{\xi=\xi_0}M^{\lambda}_{\gamma}\ ;$
- Equation (5.34) is the canonical EOM for the deviations from the linearized EOMs produced by the "nonlinear effective Hamiltonian", K.

Since K has no linear (or quadratic) part by construction, the HJ/DA equation arising from K will be closed; our system of ODEs will be (5.32), (5.33), and:

$$D\dot{S} = -DK(t) \tag{5.35}$$

where:

$$DK(t) := H(\boldsymbol{\xi}_{\text{o}}(t)+\mathbf{M}(t)\cdot d\tilde{\boldsymbol{\xi}};t) - \tilde{h}_{\alpha}(t)d\tilde{\xi}^{\alpha} - \frac{1}{2}\tilde{h}_{\alpha\beta}(t)\, d\tilde{\xi}^{\alpha}d\tilde{\xi}^{\beta}, \tag{5.36}$$

and

$$d\tilde{\xi}^{\mu} = (d\tilde{q}^{i}_{2}, d\tilde{p}_{2i}), \qquad d\tilde{p}_{2i} = \frac{\partial}{\partial\tilde{q}^{i}_{2}}DS. \tag{5.37}$$

The boundary conditions for DS are again taken as the identity:

$$DS(t_1) = DS_0 := d\tilde{p}_{1i}\, d\tilde{q}^{i}_{2} \tag{5.38}$$

Since DK consists of only third- and higher-order terms by construction, by (5.35) one sees that $\mathcal{P}_{<3}\{DS(t)\} = d\tilde{p}_{1i}\, d\tilde{q}^{i}_{2}$ is a constant of the motion; therefore the map $DS(t)$ produces is indeed a near-identity map, since it differs from DS_0 only by higher order infinitesimals.

Once a solution of the hybrid HJ/DA equation has been obtained, the tranfer matrix is implicitly given by:

$$D\xi_1^\mu = \xi_{01}^\mu + \delta_\alpha^\mu d\tilde{\xi}_1^\alpha, \qquad d\tilde{\xi}_1^\mu = (d\tilde{q}_1^i, d\tilde{p}_{1i}), \qquad d\tilde{q}_1^i = \frac{\partial}{\partial \tilde{p}_{1i}} DS_{12}, \tag{5.39}$$

and

$$D\xi_2^\mu = \xi_{02}^\mu + M_\alpha^\mu d\tilde{\xi}_2^\alpha, \qquad d\tilde{\xi}_2^\mu = (d\tilde{q}_2^i, d\tilde{p}_{2i}), \qquad d\tilde{p}_{2i} = \frac{\partial}{\partial \tilde{q}_2^i} DS_{12}, \tag{5.40}$$

with $DS_{12} := DS(t_2)$. Equations (5.32), (5.33), (5.35), and (5.36), are the central results of this Dissertation; (5.39) and (5.40) provide the means whereby these results may be used.

Suppose now that instead of the identity, one wished to determine the result of composing some previously obtained map with the map produced by the current system. Some thought should convince one that this is equivalent to breaking up the domain of integration into two pieces, $T_{12} := \{t_1 \le t \le t_2\}$ and $T_{23} := \{t_2 \le t \le t_3\}$, and using a Hamiltonian corresponding to system "A" during the first part, and "B" during the second part; the initial conditions for the second part are then $\mathbf{M}(t_2) = \mathbf{M}_2$, $DS(t_2) = DS_{12}$, where $\mathbf{M}_2$ and DS_{12} are the results of the previous integration. Then the map for the composite system will again be of the form (5.39) and (5.40), save that 3 replaces 2 everywhere! Therefore, there is no need to start both maps from the identity, and then actually perform a composition; this is a great advantage, since composition is a very computationally expensive operation.

5.7 Now that I've got it, what do I do with it ... ?

I have presented a method for extracting a mixed canonical generating-function representation of the evolution map from an arbitrary Hamiltonian system. I expect it to be fast, efficient, and easy to operate. Since it is a canonical generating function, the map it represents is guaranteed to be symplectic, even in the presence of round-off and truncation errors. There is only one question: what is a generating function good for?

The answer is: in and of itself, not much.

The generating function is an unlovely object. It has no nice symmetries or group properties. It contains its information encoded in an implicit, and not especially transparent, way. Its importance lies in the results which may be *derived* from it. by use of differentiation and elimination.

If one's purpose is symplectic tracking, the expressions (5.39) and (5.40) are already sufficient; all that remains is to use them in a Newton-Raphson type algorithm to determine the final conditions of a specified trajectory, given its initial conditions. Neither I, nor anyone else to my knowledge, has yet written such an algorithm; however it should be quite straightforward, given the existing routines in the DA-library.

If ones desires instead the explicit Taylor-series representation (*i.e.* a "transfer matrix"), one need only call the DA partial inversion routine `DAPIN` to find $(d\tilde{q}_2^i, d\tilde{p}_{1i})$ in terms of $d\tilde{\xi}_1^\alpha$, then back-substitute into (5.40) using the DA composition routine `DACCT`, to obtain $d\tilde{\boldsymbol{\xi}}_2$ in terms of $d\tilde{\boldsymbol{\xi}}_1$. However in doing so one must be willing to pay the price of losing an identically symplectic representation of the map. This is not a severe handicap, if one is not interested in performing long-term tracking studies; for these, the symplectic tracking method is clearly superior. (Recall that the Taylor-series and generating-function representations of a given symplectic map contain exactly the same amount of *information*; it is only that the Taylor-series contains it in a redundant fashion, because many of its coefficients are related by the symplectic identities. It is only the process of *evaluating* the (truncated) Taylor-series to obtain the final state that introduces non-symplecticity into the result.)

To convert to the Lie-algebraic representation, the best path seems to be to descend to the Taylor-series representation, as discussed above, then ascend back to the Lie-algebraic representation using the Dragt-Finn-Forest algorithm. The properties of generating function and Lie transforms seem to be sufficiently different that no simple *direct* path from the former to the latter exists (the converse *does* exist, however; it is the Neri algorithm discussed in §2.3.3).

Rather than tracking single particles, it is also possible to track the *moments* of a particle distribution function directly [DNR88b].

Let $\mathcal{M}$ be the system evolution map from t_1 to t_2. Let $f(\xi_1;t_1)$ be the initial phase-space distribution function. Let $P^A(\boldsymbol{\xi})$ be a complete set of functions on phase-space. Define the moments with respect to P^A by

$$\langle P^A\rangle(t_1) := \int P^A(\boldsymbol{\xi}_1) f(\boldsymbol{\xi}_1;t_1)\, d^{2d}\xi_1, \tag{5.41}$$

$$\langle P^A\rangle(t_2) := \int P^A(\boldsymbol{\xi}_2) f(\boldsymbol{\xi}_2;t_2)\, d^{2d}\xi_2, \tag{5.42}$$

By Liouville's theorem, $f(\boldsymbol{\xi}_2;t_2) = f(\boldsymbol{\xi}_1;t_1)$. Since $\boldsymbol{\xi}_2 = \mathcal{M}\boldsymbol{\xi}_1$, with $\mathcal{M}$ a symplectic map, the Jacobian $\partial\boldsymbol{\xi}_2/\partial\boldsymbol{\xi}_1$ is unimodular, and $d^{2d}\xi_2 = d^{2d}\xi_1$. Therefore,

$$\langle P^A\rangle(t_2) := \int P^A(\mathcal{M}\boldsymbol{\xi}_1) f(\boldsymbol{\xi}_1;t_1)\, d^{2d}\xi_1. \tag{5.43}$$

Since the set P^A is complete by hypothesis, there must be some set of coefficients $\mathcal{L}^A_B(\mathcal{M})$ such that

$$P^A(\mathcal{M}\boldsymbol{\xi}) = \mathcal{L}^A_B(\mathcal{M}) P^B(\boldsymbol{\xi}). \tag{5.44}$$

it immediately follows that

$$\langle P^A\rangle(t_2) = \mathcal{L}^A_B(\mathcal{M})\langle P^B\rangle(t_1) \tag{5.45}$$

i.e., the moments transform like *vectors* under the infinite-dimensional group of symplectic diffeomorphisms.

Things become especially simple if I choose the monomials $\boldsymbol{\xi}^{\langle\mu\rangle}$ as my set of basis functions. Let $m := |\langle\mu\rangle|$, be the order of the monomial I wish to calculate the moment of. From (5.43),

$$\langle\boldsymbol{\xi}^{\langle\mu\rangle}\rangle(t_2) := \int \prod_{k=1}^{m} (\mathcal{M}\xi^{\mu_k}) f(\boldsymbol{\xi};t_1)\, d^{2d}\xi. \tag{5.46}$$

Expanding $\mathcal{M}\xi^\mu$ in a Taylor series, $\mathcal{M}\xi^\mu = M^\mu_{\langle\lambda\rangle}\boldsymbol{\xi}^{\langle\lambda\rangle}$, distributing the products over the the implied sums on the $\langle\lambda_k\rangle$, and collecting like terms, one finds that that the result is a linear combination of moments:

$$\begin{aligned}
\langle\boldsymbol{\xi}^{\langle\mu\rangle}\rangle(t_2) &= \int \left[\prod_{k=1}^{m} M^{\mu_k}_{\langle\lambda_k\rangle}\boldsymbol{\xi}^{\langle\lambda_k\rangle}\right] f(\boldsymbol{\xi};t_1)\,d^{2d}\boldsymbol{\xi} \\
&= \int \left[\sum_{\langle\lambda\rangle} \sum_{\substack{\langle\lambda_1\rangle,\ldots,\langle\lambda_m\rangle \\ \langle\lambda_1\rangle,\ldots,\langle\lambda_m\rangle=\langle\lambda\rangle}} C^{\langle\lambda_1\rangle,\ldots,\langle\lambda_k\rangle}_{\langle\lambda\rangle} \prod_{k=1}^{m} M^{\mu_k}_{\langle\lambda_k\rangle}\boldsymbol{\xi}^{\langle\lambda\rangle}\right] f(\boldsymbol{\xi};t_1)\,d^{2d}\boldsymbol{\xi} \\
&= \sum_{\langle\lambda\rangle}\left[\sum_{\substack{\langle\lambda_1\rangle,\ldots,\langle\lambda_m\rangle \\ \langle\lambda_1\rangle,\ldots,\langle\lambda_m\rangle=\langle\lambda\rangle}} C^{\langle\lambda_1\rangle,\ldots,\langle\lambda_k\rangle}_{\langle\lambda\rangle} \prod_{k=1}^{m} M^{\mu_k}_{\langle\lambda_k\rangle}\right] \int \boldsymbol{\xi}^{\langle\lambda\rangle} f(\boldsymbol{\xi};t_1)\,d^{2d}\boldsymbol{\xi} \\
&= \sum_{\langle\lambda\rangle}\left[\sum_{\substack{\langle\lambda_1\rangle,\ldots,\langle\lambda_m\rangle \\ \langle\lambda_1\rangle,\ldots,\langle\lambda_m\rangle=\langle\lambda\rangle}} C^{\langle\lambda_1\rangle,\ldots,\langle\lambda_k\rangle}_{\langle\lambda\rangle} \prod_{k=1}^{m} M^{\mu_k}_{\langle\lambda_k\rangle}\right] \langle\boldsymbol{\xi}^{\langle\lambda\rangle}\rangle \qquad (5.47)
\end{aligned}$$

where the quantity $C^{\langle\lambda_1\rangle,\ldots,\langle\lambda_k\rangle}_{\langle\lambda\rangle}$ is a combinatoric factor, the exact form of which I will not need to know. The important thing to note is that (5.47) just a contraction, or "dot product", over a multi-index $\langle\lambda\rangle$ labeling the initial moments.

Note that from (5.47), $\langle\boldsymbol{\xi}^{\langle\mu\rangle}\rangle(t_2)$ is determined by a sum over all initial moments of order $\geq |\langle\mu\rangle|$. This shows why one must face a "feedback" problem in attempting to truncate and solve the moment evolution equations directly. But equation (5.46) provides the complete solution to the moment evolution equations, as a functional of the map! Since the map's evolution equations are closed, one does not in principle face any additional closure problems in truncating and evaluating (5.47) beyond those already faced in truncating the map.[4]

The sums and product in (5.47) certainly look terrifying; however in practice they are quite easy to handle, since they simply means "collect all the like terms on the right-hand-side". Because DA multiplication is isomorphic to polynomial multiplication through order n, one can simply proceed as follows:

[4]Other than the usual convergence questions involved in truncating an infinite summation, of couse. This statement also assumes that the map does not depend explicitly on the beam, *i.e.*, only *external* forces act on the particles. If one includes space charge effects, for example, then the map and beam must be determined self-consistenly, and the "feedback" problem will reappear.

- as before, use DAPIN and DACCT to obtain $D\boldsymbol{\xi}_2 := \boldsymbol{\xi}_2(D\boldsymbol{\xi}_1)$, which is equivalent to the Taylor series representation of the map.

- evaluate the product:

$$D\boldsymbol{\xi}_2^{\langle\mu\rangle} = \prod_{k=1}^{m} D\xi_2^{\mu_k}.$$

  The DA product will keep track of the exponents and collect the like terms, automatically.

- Form a DA-vector Df whose components are equal to the moments of the initial distribution:

$$f_1^{\langle\mu\rangle} := \int \boldsymbol{\xi}^{\langle\mu\rangle}\, f(\boldsymbol{\xi}, t_1)\, d^{2d}\boldsymbol{\xi}$$

- Evaluate the "dot product":

$$\langle \boldsymbol{\xi}_2^{\langle\mu\rangle} \rangle(t_2) \doteq \sum_{\langle\lambda\rangle} \left[D\boldsymbol{\xi}_2^{\langle\mu\rangle} \right]_{\langle\lambda\rangle} f_1^{\langle\lambda\rangle}. \tag{5.48}$$

  where $\left[D\boldsymbol{\xi}_2^{\langle\mu\rangle} \right]_{\langle\lambda\rangle}$ refers to the component $\langle\lambda\rangle$ of the product DA-vector $D\boldsymbol{\xi}_2^{\langle\mu\rangle}$. I have used the symbol "$\doteq$", because (5.48) is the result of truncating the infinite summation in (5.47) to the order of the DA.

One sees that transporting moments is almost as easy as transporting particles.

Finally, note that the HJ/DA method is not limited to the computation of evolution maps alone; *any* function on phase-space may be used to generate a one-parameter Hamiltonian flow via the Poisson bracket; in particlar, one may used any conserved quantity to generate a symmetry transformation. One aspect of this will be apparent in the next chapter; in three of the test-problems examined, the independent variable is z or θ, rather than t, and the "Hamiltonian" becomes the (negative of) the corresponding canonically conjugate variable.

5.8 Bounds on Solution Norm

I now show that for any norm $||\cdot||$, the seminorms $||\mathcal{P}_{\leq j}\{DS(t)\}||$ are bounded from above on every interval $T := \{t : t_1 \leq t \leq t_2\}$ over which $||\mathcal{P}_j\{K(d\boldsymbol{\xi};t)\}||$ is bounded from above, for all $d\boldsymbol{\xi}$ in some bounded domain in ${}^1_nD_v$, and grows no more rapidly than $(t_2 - t_1)$. For the remainder of this section, the following compressed notation shall hold: I shall simply write $d\boldsymbol{q}$ and $d\boldsymbol{p}$ instead of $d\tilde{\boldsymbol{q}}_2$ and $d\tilde{\boldsymbol{p}}_1$, $d\sigma := DS - d\boldsymbol{p}\cdot d\boldsymbol{q}$, $\xi^\mu(d\sigma) := (dq^i,\ dp_i + \partial d\sigma/\partial q^i)$, and $K[d\sigma;\ t] := K(\boldsymbol{\xi}(d\sigma);\ t)$.

In integral form, the HJ/DA equation is:

$$d\sigma(t_2) = -\int_{t_1}^{t_2} K[d\sigma(t);t]\,dt, \tag{5.49}$$

where I have made use of the fact that $\mathcal{P}_2\{DS\}$ is a constant of the motion, if K has no terms lower than third order, and have chosen $DS(t_1) = d\boldsymbol{p}\cdot d\boldsymbol{q}$ as my initial condition.

By construction, $K(\boldsymbol{\xi};t)$, the "nonlinear part" of the Hamiltonian defined by (5.23), contains only third order terms and higher:

$$K(d\boldsymbol{\xi};t) = K_{\mu\nu\lambda}d\xi^\mu d\xi^\nu d\xi^\lambda + K_{\mu\nu\lambda\rho}d\xi^\mu d\xi^\nu d\xi^\lambda d\xi^\rho + \dots\ , \tag{5.50}$$

where the $K_{\mu\nu\lambda}$, *etc.*, are totally symmetric phase-space tensors, and the expansion terminates after at most the n^{th} term. Therefore, K maps ${}^1_nD_v \to {}^3_nD_v$. However, I can make a stronger statement: let $d\boldsymbol{\xi}_1$ and $d\boldsymbol{\xi}_2$ be two vectors in ${}^1_nD_v$ which agree up to order j; *i.e.*, let $\Delta d\boldsymbol{\xi} := (d\boldsymbol{\xi}_2 - d\boldsymbol{\xi}_1) \in {}^j_nD_v$, so that $\mathcal{P}_{<j}\{\Delta d\boldsymbol{\xi}\} = 0$. By examining the following identity,

$$\begin{aligned} K(d\boldsymbol{\xi}_2;\ t) &= K(d\boldsymbol{\xi}_1 + \Delta d\boldsymbol{\xi};\ t) \\ &= K_{\mu\nu\lambda}d\xi_1^\mu d\xi_1^\nu d\xi_1^\lambda + 3K_{\mu\nu\lambda}d\xi_1^\mu d\xi_1^\nu \Delta d\xi^\lambda \\ &\quad + 3K_{\mu\nu\lambda}d\xi_1^\mu \Delta d\xi^\nu \Delta d\xi^\lambda + K_{\mu\nu\lambda}\Delta d\xi^\mu \Delta d\xi^\nu \Delta d\xi^\lambda \\ &\quad + K_{\mu\nu\lambda\rho}d\xi_1^\mu d\xi_1^\nu d\xi_1^\lambda d\xi_1^\rho + 4K_{\mu\nu\lambda\rho}d\xi_1^\mu d\xi_1^\nu d\xi_1^\lambda \Delta d\xi^\rho + \dots \end{aligned} \tag{5.51}$$

one concludes that, since that the $\Delta d\boldsymbol{\xi}$ always appear multiplied by at least two of the $d\boldsymbol{\xi}_1 \in {}^1_n D_j$, $K(d\boldsymbol{\xi}_2;t) = K(d\boldsymbol{\xi}_1;t) + 3K_{\mu\nu\lambda} d\xi_1^\mu d\xi_1^\nu \Delta d\xi^\lambda + \{HOTs\}$. Therefore,

$$[K(d\boldsymbol{\xi}_2;t) - K(d\boldsymbol{\xi}_1;t)] \in {}^{j+2}_n D_v, \qquad \forall \quad d\boldsymbol{\xi}_1, d\boldsymbol{\xi}_2 : (d\boldsymbol{\xi}_2 - d\boldsymbol{\xi}_1) \in {}^j_n D_v. \tag{5.52}$$

So K acts something like a "raising operator" on the difference between two DA-vectors: if $d\boldsymbol{\xi}_1$ and $d\boldsymbol{\xi}_2$ differ by terms of order j and higher, their images under K will differ by terms of order $(j+2)$ and higher.

Suppose now that $[d\sigma_2 - d\sigma_1] \in {}^j_n D_v$ Because of the presence of the ∂_i operator in the definition of $d\boldsymbol{\xi}$, $[d\boldsymbol{\xi}(\sigma_2) - d\boldsymbol{\xi}(\sigma_1)] \in {}^{j-1}_n D_v$ by (5.52), then it also it follows that:

$$[K[d\sigma_2;\, t] - K[d\sigma_1;\, t]] \in {}^{j+1}_n D_v. \tag{5.53}$$

Now let $D_T := \{(d\sigma, t) : t \in T,\, d\sigma \in \Sigma\}$ where Σ is some bounded compact domain in ${}^3_n D_v$. Since $K(d\boldsymbol{\xi};\, t)$ is a differential polynomial in $d\boldsymbol{\xi}$, it follows that the norm of $K(d\boldsymbol{\xi};\, t)$ is bounded over D_T, and therefore the seminorms $\|\mathcal{P}_j\{K(d\boldsymbol{\xi};\, t)\}\|$ are also bounded. Let the bounds on the norms be called

$$\Sigma_{max} := \sup_{D_T} \|d\sigma\|, \tag{5.54}$$

and

$$K_j^* := \sup_{D_T} \|\mathcal{P}_j\{K(d\boldsymbol{\xi}(d\sigma);t)\}\|. \tag{5.55}$$

(Note that "$\sup_{D_t} \|\cdot\|$" denotes the supremum of the *value* of the norm over the domain D_T; it is the "maximum value of the norm", not the "max-norm"). I shall show that Σ_{max} is bounded by $\sum_{j=3}^n K_j^*\,(t_2 - t_1)$, for all t such that $d\sigma(t)$ remains inside D_T.

In the spirit of Picard iteration, define the following sequence of approximate solutions to (5.49):

$$d\sigma_j(t) := -\int_{t_1}^t \mathcal{P}_{\le j}\{K[d\sigma_{j-1}(t');t']\}\, dt', \tag{5.56}$$

with the initial approximation $d\sigma_2(t) \equiv 0\ \forall\, t \in T$, so that:

$$\mathcal{P}_{>k}\{d\sigma_j\}(t) = 0, \quad \forall\, k \ge j. \tag{5.57}$$

will hold identically. From (5.50), one sees that $d\sigma_3 \in {}^3_n D_v$, Carrying out the next step in the iteration and subtracting,

$$\begin{aligned} d\sigma_4 - d\sigma_3 &= -\int_{t_1}^{t_2} [\mathcal{P}_{\leq 4}\{K[d\sigma_3(t');\, t']\} - \mathcal{P}_{\leq 3}\{K[d\sigma_2(t');\, t']\}]\, dt' \\ &= -\int_{t_1}^{t_2} [\mathcal{P}_4\{K[d\sigma_3(t');\, t']\} + \mathcal{P}_{\leq 3}\{K[d\sigma_3(t');\, t'] - K[d\sigma_2(t');\, t']\}]\, dt'. \end{aligned} \tag{5.58}$$

By (5.53), the second projector in (5.58) vanishes, since $\mathcal{P}_{\leq 3}\{{}^4_n D_v\} \equiv 0$. Therefore,

$$d\sigma_4 - d\sigma_3 = -\int_{t_1}^{t_2} \mathcal{P}_4\{K[d\sigma_3(t');\, t']\} \in {}^4_n D_v. \tag{5.59}$$

A consequence of (5.59) is that $d\sigma_3 = \mathcal{P}_{<4}\{d\sigma_4\}$, since $\mathcal{P}_4\{d\sigma_3\} \equiv 0$.

Continuing on in this fashion, I find the general term:

$$\begin{aligned} d\sigma_j - d\sigma_{j-1} &= -\int_{t_1}^{t} \mathcal{P}_j\{K[d\sigma_j(t');\, t']\}\, dt' \in {}^j_n D_v, && (5.60) \\ \Longrightarrow \mathcal{P}_{<j}\{d\sigma_j\} &= d\sigma_{j-1}, && (5.61) \\ \mathcal{P}_j\{d\sigma_j\} &= d\sigma_j - d\sigma_{j-1} && (5.62) \end{aligned}$$

for all $j \geq 2$. By (5.61) and (5.62), (5.60) becomes:

$$\mathcal{P}_j\{d\sigma_j(t)\} = -\int_{t_1}^{t} \mathcal{P}_j\{K[d\sigma_{j-1}(t'); t']\}\, dt', \tag{5.63}$$

showing that (5.56) is equivalent to:

$$d\sigma_{j+1}(t) = d\sigma_j(t) - \int_{t_1}^{t} \mathcal{P}_{j+1}\{K[d\sigma_j(t'); t']\}\, dt'. \tag{5.64}$$

Since no differential polynomial of order greater than n exists in ${}_n D_v$, the Picard iteration process is guaranteed to converge in exactly $(n-2)$ steps to the value of $d\sigma(t)$; it follows that $\mathcal{P}_{\leq j}\{d\sigma\}(t) \equiv d\sigma_j(t)$.

I now use the triangle inequality to bound each member of the sequence $\|d\sigma_j(t)\|$:

$$\begin{aligned} d\sigma_2(t) &= 0, \qquad \forall\, t \in T; && (5.65) \\ \|d\sigma_3(t)\| &\leq \left\| \int_{t_1}^{t} \mathcal{P}_3\{K[d\sigma_2(t'); t']\}\, dt' \right\| \end{aligned}$$

$$
\begin{aligned}
&\le \int_{t_1}^{t} \left\|\mathcal{P}_3\{K[d\sigma_2(t');t']\}\right\| dt' \\
&\le \int_{t_1}^{t} K_3^* \, dt' = K_3^* \, (t - t_1); && (5.66)\\
\|d\sigma_4(t)\| &\le \left\|d\sigma_3(t) \;+ \int_{t_1}^{t} \mathcal{P}_3\{K[d\sigma_2(t');t']\} \, dt'\right\| \\
&\le \|d\sigma_3(t)\| + \left\|\int_{t_1}^{t} \mathcal{P}_3\{K[d\sigma_2(t');t']\} \, dt'\right\| \\
&\le K_3^* \, (t - t_1) + K_4^* \, (t - t_1); && (5.67)\\
&\;\vdots \\
\|d\sigma_j(t)\| &\le \sum_{i=3}^{j} K_i^* \, (t - t_1). && (5.68)
\end{aligned}
$$

A slight variation on the preceding proof bounds each of the the seminorms $\mathcal{P}_j\{d\sigma\}(t)$ by $K_j^* \, (t - t_1)$.

Since there are no "feedback" terms, oscillatory and "stiff" behavior are impossible; and since the magnitudes of the coefficients of $d\sigma(t)$ are bounded by ***linear*** growth, given that nearly ***all*** numerical integrators assume the solution may be well approximated by a polynomial over each time step, I expect the method to be quite stable, even with relatively large time steps.

5.9 Estimate of the K_j^*

Finally, I place crude bounds on the K_j^* themselves.

From repeated use of the triangle inequality, for any norm, and any DA-vector DK,

$$
\|DK\|_p \le \sum_{j=0}^{n} \|\mathcal{P}_j\{DK\}\|_p \tag{5.69}
$$

(for the special case of the "1-norm", equality holds).

For the particular case of $DK = K(\boldsymbol{\xi}(d\sigma(t)); t)$, I may futher decompose each of the above order projections into "p's and q's":

$$\|\mathcal{P}_k\{DK\}\| = \Big\| \sum_{\substack{\langle i\rangle,\langle j\rangle \\ |\langle i\rangle|+|\langle j\rangle|=k}} K_{\langle i\rangle,\langle j\rangle}(t)\ \boldsymbol{dq}_2^{\langle i\rangle}\, \boldsymbol{dp}_1^{\langle j\rangle} \Big\|_p \tag{5.70}$$

$$\leq \sum_{\substack{\langle i\rangle,\langle j\rangle \\ |\langle i\rangle|+|\langle j\rangle|=k}} |K_{\langle i\rangle,\langle j\rangle}(t)|\ \|\boldsymbol{dq}_2^{\langle i\rangle}\boldsymbol{dp}_1^{\langle j\rangle}\|_p \tag{5.71}$$

$$\leq \sum_{\substack{\langle i\rangle,\langle j\rangle \\ |\langle i\rangle|+|\langle j\rangle|=k}} {}_nL_{v,p}\, |K_{\langle i\rangle,\langle j\rangle}(t)|\ \|\boldsymbol{dq}_2^{\langle i\rangle}\|_p\, \|\boldsymbol{dp}_1^{\langle j\rangle}\|_p. \tag{5.72}$$

$\|\boldsymbol{dq}_2^{\langle i\rangle}\|_p = 1$, since it is a basis element of the algebra; whereas, by the product inequality,

$$\|\boldsymbol{dp}_1^{\langle j\rangle}\| = \Big\| \prod_{k=1}^{d} (\partial_{q^k} DS(t))^{j_k} \Big\| \tag{5.73}$$

$$\leq {}_nL_{v,p}^{|\langle j\rangle|-1} \prod_{k=1}^{d} \Big\| \partial_{q^k} DS(t) \Big\|^{j_k}. \tag{5.74}$$

(Since at most $|\langle j\rangle|$ of the j_k will be non-zero, there will be at most $|\langle j\rangle| - 1$ factors of ${}_nL_{v,p}$.) Since $DS(t) = dq_2^i\, dp_{1i} + d\sigma(t)$,

$$\|\boldsymbol{dp}_1^{\langle j\rangle}\| \leq {}_nL_{v,p}^{|\langle j\rangle|-1} \prod_{k=1}^{d} \Big\| dp_{1k} + \partial_{q^k} d\sigma(t) \Big\|^{j_k} \tag{5.75}$$

$$\leq {}_nL_{v,p}^{|\langle j\rangle|-1} \prod_{k=1}^{d} \Big[\|dp_{1k}\| + \|\partial_{q^k} d\sigma(t)\| \Big]^{j_k} \tag{5.76}$$

$\|\boldsymbol{dp}_{1k}\| = 1$, since it is a basis element of the algebra. $\|\partial_{q^k} d\sigma(t)\| \leq n\|d\sigma(t)\|$,. since ∂_{q^k} simply throws some of the coefficients away, shifts the rest of them downward, and multiplies them by at most the order of the algebra, n. Therefore:

$$\|\boldsymbol{dp}_1^{\langle j\rangle}\| \leq {}_nL_{v,p}^{|\langle j\rangle|-1} \prod_{k=1}^{d} [1 + n\,\|d\sigma(t)\|]^{j_k} = [1 + n\,\|d\sigma(t)\|]^{|\langle j\rangle|}, \tag{5.77}$$

yielding the final result:

$$\|\mathcal{P}_k\{K[d\sigma(t);\, t]\}\| \leq \sum_{\substack{\langle i\rangle,\langle j\rangle \\ |\langle i\rangle|+|\langle j\rangle|=k}} {}_nL_{v,p}^{|\langle j\rangle|}\ |K_{\langle i\rangle,\langle j\rangle}(t)|\ [1 + n\|d\sigma(t)\|]^{|\langle j\rangle|}. \tag{5.78}$$

All that now remains is to choose the domain D_T, and determine the suprema of the $\mathcal{P}_k\{K[d\sigma(t);t]\}$ over it. A particularly simple choice would be to simply set the bound on $d\sigma(t)$ to some positive constant Σ_{max}, in which case:

$$K_k^* \le \sum_{\substack{\langle i\rangle,\langle j\rangle \\ |\langle i\rangle|+|\langle j\rangle|=k}} {}_nL_{v,p}^{|\langle j\rangle|} \sup_{t_1\le t\le t_2} |K_{\langle i\rangle,\langle j\rangle}(t)|\,[1+n\Sigma_{max}]^{|\langle j\rangle|}. \tag{5.79}$$

This will in turn set an upper bound on the size of (t_2-t_1) I may reach, since I must have $\sum_{j=3}^{n} K_j^*\,(t_2-t_1) \le \Sigma_{max}$ if $d\sigma$ is to remain inside D_T. I may always push up the bound on (t_2-t_1) by increasing Σ_{max}, so long as K and its derivatives do not becomes singular somewhere on the RT. However, this does not rule out the possibility that the sequence of t_2's thus generated might not accumulate to some upper limit, regardless of *how* large one makes Σ_{max}. It will therefore be necessary to determine t_2-t_1 and the appropriate Σ_{max} in a self-consistent fashion in order to find the K_j^*; however such problems often occur when determining bounds on the solution to an ODE.

Chapter 6

Verification of the HJ/DA Equation

I now describe the implementation of the "hybrid" HJ/DA method as an algorithm; the problems it was tested on; and the results of those tests. I close with an "practical example": minimizing the transverse momentum-spread emerging from a Lithium lenses.

6.1 Implementation of the Hybrid HJ/DA Equation

I chose to implement and test only the "hybrid method"of section §5.6, because I expect it to be most useful in practice. While I have not actually tested the F_2 and F_1 methods described in sections §5.4 and §5.5 on a fully nonlinear problem, it is a simple exercise to show that the *linear* part of the HJ/DA equation is satisfied by the generating functions given in (5.9) for the case of the time independent harmonic oscillator[1]. Since the *nonlinear*

[1]Any stable time-independent $2n$-degree of freedom linear Hamiltonian system can be reduced to a collection of independent harmonic oscillators. The solution of a general $2n$-DOF time-*dependent* system is in principle solvable using matrix techniques, but generally not in closed form; I have not bothered to pursue it.

parts of both the "pure" and "hybrid" methods are essentially identical, I do not expect any surprises; nevertheless, the verification of the F_1 and F_2 methods represent "loose ends" of this investigation in need of tying off.

Because I am primarily interested in a "proof of principle" demonstration, I have endevored to keep things simple; the code presented here is not really at the stage of being a practical calculation tool, ready to be integrated into a "general purpose" dynamics package, such as COSY or MARYLIE; it operates in a "stand-alone", and must be "hardwired" for the problem at hand. In particular, it will need at a minimum a more sophisticated numerical integration routine, incorporating automatic error estimation, and adaptive stepsize control. The modification of Bulirsch and Stoer's "Richardson extrapolation" method [BS66] discussed by Deuflhard [Deu83], would seem ideal for this purpose; this method automatically selects both the effective order[2] and stepsize of the integrator so as to minimize the computational effort needed to acheive a desired accuracy,

Nevertheless, I have attempted to maintain a "general purpose" design philosophy; to write the code *as if* it were to be used in a G.P. package. I have therefore attempted to follow the principles of "structured programing" (insomuch as FORTRAN allows). I avoid the use of `COMMON` blocks, passing all variables as arguments to the integrator, including the name of the Hamiltonian routine. I have broken program functions up into "modules", for ease of understanding and maintainance. Application-dependent features appear in only two modules: the "master" module, and the Hamiltonian itself; this is only one step from the irreducible minimum, which is to imbed the routines into a general-purpose dynamics package, obviating the driver, and relegating the Hamiltonian to a mere "input file".

I describe the functions of the various major blocks of code below; the actual FORTRAN listings of the various subroutines involved may be found in appendix A.

The program consists of the following parts:

[2]The order of the error in the time-step for the integrator must not be confused with the order of the DA; the two are independent parameters.

- **The Master Module:** A specialized "master" module controls each problem. The master is composed of two routines: Main, and HJDAdrive.
  - ▹ **Main:** The "main program" is essentially a dummy; its only function is to call the routine which initializes the DA package[3], followed by the "driver" routine which actually controls the integration process.
  - ▹ `HJDAdrive`: The "driver" subroutine `HJDAdrive` initializes the parameters and variables used, calls the HJ/DA integrator routine, performs those calculations required to check the results in the analytically solvable cases, and writes out the final results. The driver routine, and the Hamiltonian routine below, are the only two routines which are application-dependent.
- **The Integration Module:**

  The name of the current integration module is `HJDAbsint`. It requires the following arguements:
  - ▹ `t1`: The initial value of the independent variable.
  - ▹ `x1,R1,Ds0`: The initial values of the reference state-variables, linear transfer matrix (**M**, in 5), and DA-valued generating function, respectively. Normally, `R1` and `Ds1` are both set to the identity by `HJDXident` (see below). However by using instead the hybrid representation of a map (for example, the result of a previous integration) the final result will automatically be the composition of the current map with the input map. this is an important feature, as composition is the most expensive operation in the DA-library.
  - ▹ `t2`: The final value of the independent variable.
  - ▹ `x2,r2,Ds12`: The resultant values of the reference state-variables, linear transfer matrix, and DA-valued generating function after integration, respectively. When

[3]A quirk of the current implementation of DA is that the routine `DAINI` must be called before any other DA routine, including the routines that allocate DA variables. Since the subroutine calls to the DA library (including the allocating routine `DAALL`) are normally inserted into a program by the DAFOR preprocessor (which currently cannot distinguish between a subroutine and the main program), no DA variables may appear in the main program, unless they are allocated "by hand".

R1 and Ds0 are initiallized to the identity, R2 is the linear part, and Ds12 the generating function for the nonlinear part, of the map from t1 to t2.

- ▷ Dz,Dx: A pair of DA-valued arrays of "scratch-variables", declared to be of dimension $2 \times$ nd, order no, and number of variables nv (*i.e.,* Dx(i) $\in {}_{\mathtt{no}}D_{\mathtt{nv}}$). Here nd is the number of *pairs* of canonically conjugate variables; I allow for the possiblity that nv $\geq 2 \times$ nd so that one may include derivatives with respect to non-dynamical system parameters in the calculation.
- ▷ no,nv,nd: (See previous item.)
- ▷ **nstep,istep**: the number of sub-steps the integration interval is to be broken up into, and the number of "extrapolation stages" the integrator should use, respectively. (The effective order of the integrator is $2\times$ **istep**.)
- ▷ **hmltn**: The name of the Hamiltonian, passed in as an EXTERNAL routine.

Initially, I chose a fixed stepsize fourth-order Runge-Kutta ("RK4") method for my integration algorithm. My motivations were simplicity, a desire for a self-starting method, and a desire for ease in checking the dependence of errors on stepsize. I based my integration module on the routines RKDUMB and RK4 of Press, *et. al.'s* "Numerical Recipes" [PFTV86, pp. 553–554], modifying it to integrate equations (5.32), (5.33), and (5.34), simultaneously. Because of the low order of RK4, excessive computation effort was required, and high-accuracy calculations could not be made. This was intimately connected with the "feedforward" phenomenon; because high-order coefficients depend on lower ones, and because of the rapid increase in the number of coefficients as the order is increased, round-off error soon began to dominate the result at large order. Therefore, I replaced the RK4 integration module With a fixed-stepsize version of the "Numerical Recipes" Bulirsch-Stoer integrator [PFTV86, pp. 563–568]. Since the Bulirsch-Stoer method is effectively a variable-order method (effective order equal to twice the number of extrapolation stages), this provided the added bonus of allowing the error as a function of integrator order to be studied. I have found that for every problem investigated, four to six extrapolation stages (8th- to 12th-order integrator) was quite sufficient.

- **The Derivative Module:**

  This is the heart of the HJ/DA program. It consists of five subroutines: `HJDX1`, `HJDX2`, `HamSplit`, and `HJDAderiv`.

  ▹ `HJDXident`: Sets $\boldsymbol{\xi}_o = 0$, $\mathbf{M} = \mathbf{I}$, and DS to the identity.

  ▹ `HJDX1`: Computes $D\boldsymbol{\xi}_1$, at time `t1`, from $\boldsymbol{\xi}_{01} \Leftrightarrow$ `x1`, $\mathbf{M}_1 \Leftrightarrow$ `R1`, and $DS \Leftrightarrow$ `Ds12`, using (5.21).

  ▹ `HJDX2`: Computes $D\boldsymbol{\xi}_2$, at time `t2`, from $\boldsymbol{\xi}_{02} \Leftrightarrow$ `x2`, $\mathbf{M}_2 \Leftrightarrow$ `R2`, and $DS \Leftrightarrow$ `Ds12`, using (5.21).

  ▹ `HamSplit`: Separates DH into h_α, $h_{\alpha\beta}$, and DK, using (5.23).

  ▹ `HJDAderiv`: Evaluates the right hand side of (5.32), (5.33), and (5.34).

  All five of these routines are perfectly general, and could be used in a production code without modification.

- **The Matrix Module:**

  For clarity, the vector and matrix multiplications involved in evaluating the EOMs for $\boldsymbol{\xi}$ and $\mathbf{M}$, ((5.32), (5.33), and (5.34)), have been split off into the following subroutines; also included is a special-purpose inverter for symplectic matrices:

  ▹ `MMmul`: multiplication of two matrices,

  ▹ `MVmul`: multiplication of a vector by a matrix,

  ▹ `RJmul`: multiplication of a matrix by $\boldsymbol{J}$ from the right, and

  ▹ `SympInv`: Inversion of a symplectic matrix, using the identity $\mathbf{M}^{-1} \equiv \mathbf{J}\mathbf{M}^{\mathrm{T}}\mathbf{J}^{-1}$.

  These routines have been written for simplicity and generality, rather than efficiency; while modest improvements in speed could probably be made by "hardwiring" these routines into HJDAderiv, the fraction of computational overhead thus eliminated would probably not be worth the effort.

- **The Hamiltonian Module:**

  Finally, there is the module computing the Hamiltonian itself. It is the only other user-supplied module in the program. Its name is passed to the program as an argument using FORTRAN's "EXTERNAL" mechanism, so there are no restrictions on the routine's name. Only a minimal knowledge of DA is required by the user; essentially, one just writes a FORTRAN routine to compute the Hamiltonian, and keep in mind a few simple rules:

  1. The argument list is of the form: "(t,Dx, no,nv,nd Dh)". Here t is the independent variable, Dx is one-dimensional array of length $2\times$nd containing the DA-valued phase-space variables, no, nv, and nd are defined as before, and Dh is the value of the DA-valued Hamiltonian returned.
  2. Dx and Dh must be declared as DA by the statement:

     "*DAEXT(no,nv) Dx(nd*2), Dh";

     the * must occupy the first column. (All statements flagged for preprocessing by DAFOR start with a "*DA" in the first column; therefore the FORTRAN compiler considers them to be "comments".)
  3. Any local DA-valued variables must be declared using a "*DAINT(no,nv)" statement analogous to the "*DAEXT(no,nv)" statement above.
  4. Any statements containing a DA-valued variable must be flagged by placing a "*DA" in the first column.

  And that is all one really needs to know; for examples, please see the routines drift, UBfield, and polrsho in appendix ??.

- **The Tracking Module:** This module contains two functions used for tracking purposes: DFeval and DFdot.

- ▷ `DFeval`: Double precision function `DFeval( x0,Df, no,nv, Xn,Xnn )`; evaluates to the value of the DA variable `Df` interpreted as a Taylor series, at argument `x0(nv)`. returns auxilliary 1-D arrays `Xn` containing the partial sum through each order up to no, and `Xnn` containing the contribution each order makes to `DFeval`.
- ▷ `DFdot`: Double precision function `DFdot( Df,Dg, DotProd,DotByOrd )`; evaluates to the value of the "dot product" of `Df` and `Dg`; returns auxilliary 1-D arrays `DotProd` containing the partial sum through each order up to no, and `DotByOrd` containing the contribution each order makes to `DFdot`.

- **Miscellaneous Subroutines:**

  - ▷ `norms`: Calculates the sum of the 1-norms and the max of the max-norms for a 1-D array of DA-valued quantities. Also returns the contribution made by each order to the above measures.
  - ▷ `mypri`: Special routine printing a sorted comparison of two DA-valued quantities, including the absolute and fractional difference of each component when both are non-zero.
  - ▷ **The `RayGen` Module:** Special truncated Gaussian random beam generator. The particle sets it generates are guaranteed to have zero mean, specified standard deviations along each axis, and to be "upright beam ellipses", *i.e.*, zero correlation coefficients in the x–p_x and y–p_y phase-planes. See the listing in app. ?? for details. Includes a subroutine for computing the first and second moments of a particle set.

6.2 Testing the HJ/DA Equation

Finding a good set of test problems for the HJ/DA equation was somewhat difficult. On the one hand, I would like test problems which can be solved in closed form5, so I may be confident of my results. On the other hand, I demand test problems which are fully nonlinear, in order to provide a true test of the method; however nonlinear systems solvable in closed form are hard to find. Fortunately, I can meet both preceding criteria if the test problem can be solved using *geometry alone*. Three such systems are the *uniform relativistic drift*, *two-dimensional simple harmonic oscillator* (2D-SHO) *in polar coordinates*, and a *particle in a uniform magnetic field*.

6.2.1 Case 1: The Uniform Relativistic Drift

It is customary in both kinds of "optics" to take the arc-length along the RT, s, as the evolution parameter, and time as a coordinate. This is because, in perturbative optics, it is usually easier to determine *when* a particle arrives at a given optical element, than it is to determine *which element* a particle has arrived at at a given time. It is simple to show that with this parameterization, t and $p_t := -E$ are conjugate variables, while the "Hamiltonian" becomes $-p_s$ [Dra82]. For example, if I choose $s = z$,

$$H = -\left[\frac{1}{c^2}(p_t + q\phi)^2 - \left(p_x - \frac{q}{c}A_x\right)^2 - \left(p_y - \frac{q}{c}A_y\right)^2 - m^2c^2\right]^{\frac{1}{2}} - \frac{q}{c}A_z \qquad (6.1)$$

which is exactly what I would get by solving algebraically for $-p_z$. The other p's do not change under this transformation; however Hamilton's equations now take the form of:

$$\frac{dx}{dz} = \frac{\partial H}{\partial p_x}, \qquad \frac{dp_x}{dz} = -\frac{\partial H}{\partial x},$$

and similarly for y and t.

The other property I shall need is the change of form of Hamilton's equations under rescaling. For numerical purposes, it is best to scale the physical variables by putting them into "dimensionless" form, choosing the scales so that all quantities are of order unity. Suppose that t is a cyclic variable (this will be the only case I shall need); then p_t can be treated as a fixed parameter. Under the scaling transformation:

$$x = \ell X, \qquad y = \ell Y, \qquad p_x = p_\perp P_X, \qquad p_y = p_\perp P_Y, \qquad z = LZ,$$

the canonical equations become:

$$\frac{dX}{dZ} = \frac{\partial \bar{H}}{\partial P_X}, \quad \frac{dP_X}{dZ} = -\frac{\partial \bar{H}}{\partial X},$$
$$\frac{dY}{dZ} = \frac{\partial \bar{H}}{\partial P_Y}, \quad \frac{dP_Y}{dZ} = -\frac{\partial \bar{H}}{\partial Y}, \tag{6.2}$$

with:

$$\bar{H}(X,Y,P_X,P_Y;Z) := \frac{L}{\ell p_\perp} H(\ell X, \ell Y, p_\perp P_X, p_\perp P_Y; LZ) \tag{6.3}$$

Rescaling is *not* a symplectic transformation, because the determinant is not unity; nevertheless, the *structure* of Hamilton's equations is preserved, so it is conventional to admit rescalings to the family of "generalized" canonical transformations [Gol80, chap. 9,p. 381].

For a free relativistic drift (*i.e.*, a field-free region of space), s is again simply z. For simplicity, I limit myself to (2+1) dimensions, (t,x,z). From the "mass-shell constraint", $E^2 - p_x^2c^2 - p_z^2c^2 = m_0c^4$, the Hamiltonian is:

$$H := -p_z = -\sqrt{\frac{{p_t}^2}{c^2} - {p_x}^2 - m^2c^2}.$$

The sign of H has been chosen to make t increase with increasing z. Choosing units such that $m = c = 1$,

$$H := -\sqrt{{p_t}^2 - {p_x}^2 - 1}. \tag{6.4}$$

Using a prime to denote d/dz, Hamilton's equations for a drift are:

$${p_t}' = -\frac{\partial H}{\partial t} = 0, \quad t' = \frac{\partial H}{\partial p_t} = \frac{p_t}{H} = \frac{(-E/c)}{(-p_z)} = \frac{1}{\beta_z}, \tag{6.5}$$

$${p_x}' = -\frac{\partial H}{\partial x} = 0, \quad x' = \frac{\partial H}{\partial p_x} = -\frac{p_x}{H} = \frac{p_x}{p_z} = \frac{\beta_x}{\beta_z}, \tag{6.6}$$

the solution of which can be found in closed form:

$$t_2 = t_1 + z\frac{p_t}{H}, \quad p_{t2} = p_{t1}, \qquad x_2 = x_1 - z\frac{p_x}{H}, \quad p_{x2} = p_{x1} \tag{6.7}$$

Since the Hamiltonian depends only on the p_i, the HJ/DA equation can also be solved in closed form. For simplicity I treat the energy as a "fixed parameter", since $p_t = -\gamma_0$ is a constant of motion.[4] Initial conditions are: $x_{01} = 0$, $p_{x01} = 0$ (on-axis beam), $\mathbf{M}_1 = \mathbf{I}$, $DS_0 = d\tilde{p}_{x1} d\tilde{x}_2$. Using

$$h_\alpha = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \qquad h_{\alpha\beta} = \begin{bmatrix} 0 & 0 \\ 0 & \dfrac{1}{\gamma_0\beta_0} \end{bmatrix}, \tag{6.8}$$

the solution may easily be verified to be:

$$M^\mu_\alpha(z) = \begin{bmatrix} 1 & \dfrac{z}{\gamma_0\beta_0} \\ 0 & 1 \end{bmatrix} \tag{6.9}$$

$$DK = -[\gamma_0^2\beta_0^2 - d\tilde{p}_x^2]^{\frac{1}{2}} - \frac{1}{2}\frac{d\tilde{p}_x^2}{\gamma_0\beta_0} \tag{6.10}$$

$$DS_{12}(z) = d\tilde{p}_{x1}\, d\tilde{x}_2 - zK(\tilde{p}_{x1}) \tag{6.11}$$

(Note: equation (6.11) is actual valid for *any* "purely nonlinear" Hamiltonian K which depends only on the $\boldsymbol{p}$'s.)

I ran this test problem in ${}_{12}D_2$ for $z = 1$ (a unit length drift), in twenty steps ($\Delta z = .02$), with six extrapolation stages (effective order 12), and the "zero-tolerance cutoff" parameter of the DA-library (minimum size below which a coefficient is set to zero) set to 1.0×10^{-30}. Since the results of the run are several rather large tables of numbers, I have relegated them to appendix A.

Table A-3 shows the theoretical matrix $\mathbf{M}_{theo}$; Table A-2 shows the corresponding numerical result $\mathbf{M}_{num}$. Table A-1 shows the difference between $\mathbf{M}_{theo}$ and $\mathbf{M}_{num}$; there are only two non-zero elements, having absolute errors of 4×10^{-17} and 8×10^{-17} or a few parts in 10^{-16} fractional error.

I have not reported a comparison between the theoretical and numerical solutions to the RT equations, because this difference is identically zero, by virtue of the ICs chosen (initial momentum parallel to z-axis).

[4]So is p_x, for that matter, but I need at least *one* dynamical variable, or the problem becomes trivial!

I also do not report a check made of the symplectic identity for the matrix part. The measure of symplecticity error used was: $\mathbf{M}_{num}\mathbf{M}^{-1}{}_{num} - \mathbf{I}$, $\mathbf{M}^{-1}_{num} := \mathbf{J}\mathbf{M}^{\mathrm{T}}{}_{num}\mathbf{J}^{-1}$; this difference is also identically zero, by virtue of the structure of the EOMs for the matrix.

Table A-4 shows the theoretical and numerical HJ functions, DS_{theo} and DS_{num}. Note that the difference is again small, about 10^{-18} - 10^{-16} . The fractional difference is also shown; note that even this is very small, about 10^{-17} - 10^{-15} . Had the zero-tolerance parameter been set to something more reasonable, like 1.0×10^{-15} these differences would have "vanished".

Note that the average magnitude of the coefficients in *DS* are roughly constant, or at best weakly decreasing with order. This should not be cause for alarm, however, since as coefficients of a "perturbation expansion", one expects them to behave at best like an asymptotic expansion. Since, even when veiwed at a Taylor-series, the coefficients appear multiplied by "small" quantities, *DS* actually appears to be *converging*, and converging approximately like a geometric series.

A additional test is provided by rewriting (6.7) as:

$$t_1 = t_2 - z\frac{p_t}{H}, \tag{6.12}$$

and

$$x_1 = x_2 + z\frac{p_x}{H}; \tag{6.13}$$

Tables A-5 and A-6, app. A, compare the right- and left-hand sides of A-5 and A-6. Note that all terms are again small.

6.2.2 Case 2: The 2-D Harmonic Oscillator in Polar Coordinates

In Cartesian coordinates, the 2D-SHO Hamiltonian is a quadratic form, and its equations of motion are linear. However by making a "bad" choice of coordinates (polar coordinates), the Hamiltonian for this system becomes nonlinear:

$$H = \frac{1}{2}p_r^2 + \frac{1}{2}\frac{p_\theta^2}{r^2} + \frac{1}{2}\omega r^2 \tag{6.14}$$

(I have set $m = 1$ in H, since it is a "scale" parameter, not a dynamical variable.)

In this case, I have not been able to find a closed-form solution to the hybrid HJ/DA equation, because of the complicated time-dependence induced is K by the RT.[5] However, I don't really *need* an explicit solution, since the HJ/DA equation yields *implicit* relations, anyway. If I can find any set of four equations between $(r_1, \theta_1, p_{r1}, p_{\theta 1})$ and $(r_2, \theta_2, p_{r2}, p_{\theta 2})$ which are identically satisfied by every solution to the EOMs of our harmonic oscillator, they must also be satisfied identically by the canonical transformation generated by a solution to the HJ equation, since the two are equivalent. The DA equivalent of this statement is that an n^{th}-order solution DS_{12} to the HJ/DA equation must identically satisfy these same four relations to $(n-1)^{th}$-order (since I must differentiate once to obtain the $\tilde{p}_2$ and $\tilde{q}_1$).

Now, I know that the general solution to a 2D-SHO in *Cartesian* coordinates is given by:

$$x_2 = x_1 \cos(\omega t) + \frac{1}{\omega} p_{x1} \sin(\omega t) \tag{6.15}$$

$$y_2 = y_1 \cos(\omega t) + \frac{1}{\omega} p_{y1} \sin(\omega t) \tag{6.16}$$

$$p_{x2} = -\omega x_1 \sin(\omega t) + p_{x1} \cos(\omega t) \tag{6.17}$$

$$p_{y2} = -\omega y_1 \sin(\omega t) + p_{y1} \cos(\omega t), \tag{6.18}$$

and these, together with the polar-rectangular conversion formulas:

$$p_x = p_r \cos(\theta) - \frac{1}{r} p_\theta \sin(\theta) \tag{6.19}$$

$$p_y = p_r \sin(\theta) + \frac{1}{r} p_\theta \cos(\theta) \tag{6.20}$$

and

$$p_r = p_x \cos(\theta) + p_y \sin(\theta) \tag{6.21}$$

$$p_\theta = r\left[-p_x \sin(\theta) + p_y \sin(\theta)\right] \tag{6.22}$$

[5]For the special RT leading to "uniform circular motion", r, p_r, and p_θ are all constant, and a solution can in principle be found by separation of variables; however it in addition to being rather complicated, it satisfies the wrong BC's.

provide four relations of exactly the type I need. So I can still test the HJ/DA method on this problem, even though I do not have an *explicit* closed-form solution for DS.

I ran this test problem for $\omega t = 3\pi/4$ (passing through the F_2 method's caustic at $\omega t = \pi/2$, just to show it is not a problem for the hybrid method), with a stepsize of $\omega\Delta t = 2\pi/120$ (30 steps), number of extrapolation stages **istep** $= 6$, and zero-tolerance parameter set to 1.0×10^{-30} . The initial conditions on the RT were: $r_{01} = r_0 = 1.0$, $p_{r01} = (0.1)r_0\omega$, $\theta_0 = 0.0$, $p_{\theta 0} = (1.1)r_0^2\omega$. Tables B-3–B-6, appendix B, show the differences and fractional differences between the right- and left-hand-sides of (6.15–6.18); as in the previous case the differences are all small, about 10^{-16} – 10^{-14} . The fractional differences are also small, for the most part; occasional larger values occur, but only between coefficients which are themselves very small, and probably would have vanished if the zero-tolerance parameter been set to something more reasonable.

6.2.3 Case 3: Uniform Magnetic Field

A commonly used charged-particle optics element is the "normal-faced uniform horizontal bend"; this is simply a magnet with a uniform vertical field, designed so a particle moving on the reference trajectory with the design energy enters and exits perpendicular to the faces of the magnet. Neglecting edge-effects, and restricting motion to the x-y plane, one knows that inside the magnet, every charged particle must follow an arc of a circle.[6] If I know the energy and entry angle of a particle, it is a matter of simple geometry to work out the relations between the phase-space variables at entry and exit:

$$\sin(\alpha_1) = \frac{p_{r1}}{p}, \qquad \sin(\alpha_2) = \frac{p_{r2}}{p}, \tag{6.23}$$

$$pc = qBR = \sqrt{E_1^2 - m^2c^4} = \sqrt{E_2^2 - m^2c^4} \tag{6.24}$$

[6]If I include the z-component as well, the particle move along a helix, instead; however since the Hamiltonian is independent of z, p_z is conserved, and has no effect other than to alter the gyration radius.

$$x_{*1} = R\sin(\alpha_1) \tag{6.25}$$

$$x_{*2} = R\sin(\alpha_2)\cos(\theta) + [r_2 - R\cos(\alpha_2)]\sin(\theta) \tag{6.26}$$

$$y_{*1} = r_1 - R\cos(\alpha_1) \tag{6.27}$$

$$y_{*2} = -R\sin(\alpha_2)\sin(\theta) + [r_2 - R\cos(\alpha_2)]\cos(\theta) \tag{6.28}$$

where the meanings of the various quantities above are illustrated in figure F-1, appendix F.

In terms of time, the Lagrangian of a charged particle in a uniform magnetic field B_0 is:

$$L = -mc\sqrt{c^2 - \dot{r}^2 - r^2\dot{\theta}^2} - \frac{qB_0}{2c}r^2\dot{\theta} \tag{6.29}$$

Following custom, I shall use θ as my independent variable; the Lagrangian becomes:

$$L\,dt = \Lambda\,d\theta = \left\{-mc\sqrt{c^2t'^2 - r'^2 - r^2} - \frac{qB}{2c}r^2\right\}d\theta \tag{6.30}$$

Introducing dimensionless coordinates $\rho := r/R_0$, $\tau := ct/R_0$, Λ becomes:

$$\Lambda = -mcR_0\left\{\sqrt{\tau'^2 - \rho'^2 - \rho^2} + \frac{qB_0R_0}{2mc^2}\rho^2\right\} \tag{6.31}$$

Choose $R_0 = p_0c/qB_0$, the radius of gyration of a particle having the reference momentum $p_0 = mc\gamma_0\beta_0$. Then Λ becomes:

$$\Lambda = -mcR_0\left\{\sqrt{\tau'^2 - \rho'^2 - \rho^2} + \frac{1}{2}\gamma_0\beta_0\rho^2\right\} =: mcR_0\hat{\Lambda} \tag{6.32}$$

Choosing units such that $m = c = R_0 = 1$, the Hamiltonian associated with $\hat{\Lambda}$; is:

$$H := p_\tau\tau' + p_\rho\rho' - \hat{\Lambda} = \rho\left\{\frac{1}{2}\gamma_0\beta_0\rho - \sqrt{p_\tau^2 - p_\rho^2 - 1}\right\} \tag{6.33}$$

Despite its odd-looking form, this is actually just equal to $-p_\theta$ (in non-dimensional form, of course!). This is the Hamiltonian I shall use in the HJ/DA equation

I ran this test problem for $\theta = 3\pi/4$, with thirty steps; I chose the IC's for the reference trajectory as: $\rho_{01} = 1.0$, $p_{\rho 01} = 0.0$, so that it is a circle of radius R_0, centered on the origin. Since $p_\tau = -\gamma$ is again a constant of motion, I again treat it as a parameter, and

ρ and p_ρ as my only variables. The results are presented in tables UB-mat through UB-tst2, appendix C; again, note that the differences between (6.25) and (6.26), (Table C-3, appendix ??), and also (6.27) and (6.28) (Table C-4, appendix ??), are quite small.

As a final amusing example, I integrate with the same initial conditions around a full 2π radians; The results are presented in tables C-5 through C-8, Appendix C. Note that the RT returns very nearly to its initial conditions, and the matrix and generating-function parts to the identity; in particular, the coefficients of the generating function differ from the identity by only about $10^{-16} - 10^{-14}$. Occasional large fractional differences do occur, however this should not be too surprising: since ideally all the generating function coefficients should have *vanished*, one expects that the small (but non-zero) values observed are entirely due to truncation and round-off error in the numerical integrator, and therefore quite uncertain.

6.2.4 Convergence Study

The errors shown in appendices A–C are small, but they are not zero. How are these errors affected by step size and extrapolation order? To determine the answer, I ran `UBfield` with the same IC's as before, but with `nstep` varying from 8 to 128, and `istep` varying from 2 to 6. I shall use the matrix max-norm on the symplecticity error matrix, [7] and the sum of the 1-norms (3.56) or max-norms (3.57) of the error-vectors (differences of (6.25) and (6.26), and (6.27) and (6.28)) as a measure of the "overall error". Since I am also interested in the error depends on order, I shall also look at the "1-seminorms" and "max-seminorms" defined by (3.62) Because of the closure property of DA, I am guaranteed that the results at k^{th} order are not influenced by $(k+1)^{th}$ and higher orders, so it is sufficient to do this once and for all[8]; I do not need to run the problem for varying values of k. In order to be able to compare more readily errors at different orders, I shall look at the "average error per coefficient" by rescaling the norm and seminorms by ${}_nN_v$ and ${}_nN_{v-1} = {}_nN_v - {}_{n-1}N_v$,

[7]For 2×2 matrices, the matrix 1-norm and max-norm only differ by a factor of 2, because one can show that $\mathbf{MJM^TJ} = \det\{\mathbf{M}\}bfI$.

[8]Note that I could not have made this claim, if the HJ/DA method were not free of "feedback" problems.

respectively. I also look at the "max-norm" (3.57. Tables D-1, through D-14, app. D, show how these error-measures vary with **nstep** and **istep**. One sees that, like the DA-coefficients themselves, they vary only slowly with order; again, this should not be cause for alarm, unless the *fractional* error increases with order. Since *DS* is equivalent to a "perturbation expansion", one expects that it is at best an asymptotic expansion anyway, and therefore one should expect its coefficients to begin to diverge for sufficiently high order.

Figures F-4–F-17, app. F, show log-log plots of the sum of the 1-norms and max-norms *vs* **nstep**; these curves are qualitatively very similar, since inspection of tabels A-5, A-6, B-3–B-6, C-3, C-4, and C-7, C-8 show that there are usually one or two coeffieciemts at each order which dominate the error. The slope of the early portions of these curves are consistent with the $\mathcal{O}(h^{2\mathtt{istep}})$ behavior one expects from the integrator. The tendency of these curves to "bottom out" at about **nstep** $\simeq$ 24 and begin to slowly rise again is also quite typical behavior for a numerical integrator [How74, HNW87]: for large step size (small **nstep**), the "local truncation error" (discretization error; "order" of method) dominates, and the error goes like $\mathcal{O}((h/\mathtt{nstep})^{2\mathtt{istep}})$; while for small step size (small **nstep**), the "global round-off error" dominates, so the error goes like $\mathcal{O}((h/\mathtt{nstep})^{-1})$.

For comparison, I also hold the stepsize fixed and vary the number of extrapolation stages **istep**. Figures F-18–F-31 show log-linear plots of the errors *vs* **istep**; since the error due to finite step-size is proportional to $h^{2\mathtt{istep}}$, these should be straight lines (the coefficient of the error term actually also varies somewhat with **istep**, as well as the effective order, so one does not expect them to be *exactly* straight). Ones sees that the lines on figs. F-18–F-31 are indeed remarkably straight; however they also tend to "bottom out" for sufficiently high **nstep** and **istep**.

The "bottoming out" tendency shows that there is little point in increasing **nstep** and **istep** beyond a certain point. This points out the need for effective methods of order and step size control in a "practical" implementation of the HJ/DA method. Fortunately, the Bulirsh-Stoer method also provides an internal estimate of the solution error, and algorithms exist which automatically optimizing the order and step size to achieve a perscribed estimated error with minimum computational effort [BS66], [HNW87]. As stated earlier, I chose not to implement order and step size control in `HJDAbsint`, because it would have made the integrator much more complex, and greatly complicate the convgerence behavior analysis; however I consider it a "must" for a production version of this code.

6.3 Example Application: Optimization of a Lithium Lens

To show an application of the HJ/DA method to a "practical" calculation, I give the following example: optimization of a combined target/lithium-lens system for minimum final transverse momentum spread. Before I discuss what a lithium lens is, and why a minimum final transverse momentum spread is desirable, I must first give a brief discussion of the "emittance" and "beam ellipse" concepts.

6.3.1 The "Emittance" and "Beam Ellipse" Concepts

The "emittance", ϵ, is a common figure of merit for beam quality. Emittance is a measure of the "phase-space volume" a distribution of particles occupies. One often desires the emittance of a beam to be as small as possible; for example, the event rate in a colliding-beam experiment naively scales like $1/\epsilon_x\epsilon_y$.

Now by Liouville's theorem, the phase-volume of a distribution is conserved by Hamiltonian flows; ideally, emittance should share this property. Finding a good measure of the phase-volume is difficult, however, since in practice some form of "course graining" must be performed. The problem is exacerbated by the fact that, for many-particle systems, Liouville's theorem is valid only for the $2dN$-dimensional N-particle distribution function, whereas the experimenter usually only has access to averages over the 1-particle distribution function at best.[9] "Emittance" is therefore a somewhat ambiguously defined quantity. There are many definitions of "emittance" in common use; unfortunately, the principle thing that they all have in common is that they are unsatisfactory in some way [LLG73].

One popular version is the "rms-emittance", defined in terms of *second moments of the distribution function*. For *linear* systems, with no cross-plane coupling, the quantity [10]

$$\bar{\epsilon}_x := [4]\Big\{\langle x^2\rangle\langle p_x^2\rangle - \langle x p_x\rangle^2\Big\}^{\frac{1}{2}} \tag{6.34}$$

is a constant of the motion, as is the analogously defined ϵ_y. (The factor of four in square brackets is frequently dropped.)

For nonlinear systems, (6.34) is no longer conserved, and is usually observed to increase. This is usually attributed to "phase-space filamentation and mixing", discussed further below.

To get a physical interpretation of (6.34), assume the beam has a Gaussian distribution in the transverse coordinates and momenta, [11] and again, that there is no coupling between planes; then ϵ_x enjoys simple physical interpretation: it is equal to the area of the elliptical "one-sigma" contour in the x–p_x plane, divided by π. This picture is so common and useful that "phase space ellipses" are used almost exclusively to describe the shape of a beam. The mathematical description of beams is either in terms of the "Twiss parameters" α, β, and γ, [12] which are related to the one-sigma contour by:

$$\gamma x^2 + 2\alpha x p_x + \beta p_x^2 = \frac{\epsilon}{\pi}, \qquad \beta\gamma - \alpha^2 = 1.$$

[9] Even more likely, one only has access to averages of *projections* of the distribution onto the spatial coordinates. Relatively non-invasive methods of measuring the spatial distribution of the beam exist, however momentum measurements are much harder. Usually one must insert a slit or pinhole into the beam, then measure the spatial distribution of the emerging beam after a short drift.

[10] It is actually far more common to use the "slopes" x' and y', $x' := dx/dz$ to describe particle trajectories, rather than p_x, p_y. However $x' \simeq p_x/p_z$, $p_z \simeq \{const\}$ to second order in transverse coordinates and momenta, so my description differs from the conventional one only by a scale factor.

[11] This is actually fairly close to what is usually seen, so long as space charge effect are negligable.

[12] One of the more confusing aspects of accelerator physics is the extrordinary number of unrelated things the letters α, β, and γ are used for.

Alternatively, one uses the "sigma matrix" which is related to the Gaussian beam distribution by:

$$f(\boldsymbol{\xi};t) \propto \exp[-\tfrac{1}{2}\boldsymbol{\xi}\cdot\boldsymbol{\sigma}^{-1}\cdot\boldsymbol{\xi}],$$

in which case the one-sigma contour is given by $\boldsymbol{\xi}\cdot\boldsymbol{\sigma}^{-1}\cdot\boldsymbol{\xi} = 1$.

While the "one-sigma" emittance provides an intuitive picture for Gaussian beams, closely related to the only physical scales one has in this case, it is somewhat unsatisfactory in that one can show that even in only two dimensions, only 14.7% of a Gaussian beam is contained within the one-sigma contour; it therefore represents only the innermost core of the beam, and the fraction get rapidly smaller as the dimensionality increases. It is also not clear how one should apply the "one sigma" criterion to *non*-Gaussian beams; for example, a space-charge dominated beam tends to evolve until its spatial distribution function is approximately uniform (a "flat" beam); what does one mean by the "one sigma contour" then? One sometimes defines the emittance to be the area of the smallest ellipse containing some large specified fraction of the beam — 90% and 95% are popular choices.

The definition (6.34) reduces to the one-sigma ellipse for Gaussian beams, and has the advantage of being well-defined even for "point" distributions, such as one obtains from a Monte-Carlo simulation. Unfortunately, it is not "robust"— the second moment tends to weight "outliers" too highly, and so may *over*estimate the size of the beam, rather than underestimate it. Furthermore, there are fairly well-behaved distributions for which the second moment diverges — a Lorentzian, for example. Again, one may compromise by choosing the ellipse having the same aspect ratio and orientation as the ellipse determined via second moments (assuming they exist; for a finite size particle set this is always the case), but rescaled so that 90% or 95% of the beam is included.

Lawson, Lapostolle, and Gluckstern have shown that the entropy of the distribution is proportional to the log of the rms emittance [LLG73]. One might therfore suggest that "emittance " be *defined* in terms of beam entropy for non-Gaussian beams. Unfortunately, the entropy of the beam is impossible to measure, in practice, so this definition is of only conceptual utility, as they themselves recognize. However it might be a useful definition of emittance for numerical simulations using non-Gaussian beams.

Despite its potential pitfalls, for the remainder of this section one may assume that I mean (6.34) when I say "emittance", as it is simple to calculate in a Monte-Carlo simulation, and I shall be using Gaussian beams in my example.

A "dual" concept to emittance is the *acceptance*. This is the area of the largest phase-area in each phase-plane that a beam transport line can accept without losses. It, too, is usually parameterized by assuming it is an ellipse. A characteristic of the most common beam transport lines is that if the beam ellipse and acceptance ellipse are *geometrically similar* on injection, they will remain similar as the beam is transported down the beamline. A beam satisfying this condition is said to be "matched". If a beam is not matched, then it "rotates" relative to the acceptance ellipse. When nonlinear effects are present, the resulting dependence of frequency on amplitude cause the beam to deform as it rotates, making it "wrap up" until the initially elliptical beam looks more like a spiral nebula. The beam therefore tends to "fill up" the machine ellipse which initially contained it, causing most measures of its emittance to grow. This process is called "filamentation and mixing", and the result is refered to as "emittance dilution".

6.3.2 The Lithium Lens

A "lithium lens" is an axisymmetric focusing device used for highly relativistic particles. It is essentially a solid cylinder of lithium (chosen for its low particle scattering and absorption cross-sections) through which one drives a high-density axial current ($\sim$ MA/cm^2). Assuming a uniform current density J_z, [13] by Ampere's law the resulting circumferential

[13]Uniform current density is physically reasonable for time-independent currents. However a lithium lens is almost always pulsed, in order to cope with the heating caused by the immense current-density used. Therefore the current-density will be non-uniform, because the magnetic field must diffuse into (out of) the body of the metal cylinder during the leading (trailing) edge of the pulse. The pulse is usually timed so that the particles to be focused pass through the lens at the moment of maximum current uniformity.

magnetic induction is given by:

$$B_\theta = \frac{2\pi r}{c} J_z.$$

One sees that B_θ rises linearly with distance from the axis. The field is essentially confined to within the body of the cylinder, because one usually arranges for the return current to flow back along a coaxial shell surrounding the lens; this helps to minimize the inductance of the system.

The vector potential producing a linearly rising B_θ is:

$$A_z = -\frac{\pi}{c} J_z r^2 = -\frac{I_0}{c}\left(\frac{r}{R_0}\right)^2,$$

where I_0 is the total current through the lens, and R_0 its radius. Since $-A_z$ plays the role of the "potential" when the evolution parameter is z, one therefore expects harmonic oscillator-like behavior for sufficiently small transverse momenta.

The Lagrangian is given by:

$$\begin{aligned} \mathcal{L}\,dt &= \left\{ -mc\sqrt{c^2 - \dot{x}^2 - \dot{y}^2 - \dot{z}^2} - \frac{q}{c}\frac{I_0}{c}\left(\frac{x^2+y^2}{R_0^2}\right)\dot{z}\right\} dt \\ &= \left\{ -mc\sqrt{c^2 dt^2 - dx^2 - dy^2 - dz^2} - \frac{qI_0}{c^2}\left(\frac{x^2+y^2}{R_0^2}\right) dz\right\} \\ &= \left\{ -mc\sqrt{c^2 t'^2 - x'^2 - y'^2 - 1} - \frac{qI_0}{c^2}\left(\frac{x^2+y^2}{R_0^2}\right)\right\} dz \qquad (6.35) \end{aligned}$$

(6.36)

The corresponding Hamiltonian is:

$$\begin{aligned} H &= -\sqrt{\frac{1}{c^2}p_t^2 - p_x^2 - p_y^2 - m^2c^2} + \frac{qI_0}{c^2}\left(\frac{x^2+y^2}{R_0^2}\right) && (6.37) \\ &= -\sqrt{p_0^2 - p_x^2 - p_y^2} + \frac{qI_0}{c^2}\left(\frac{x^2+y^2}{R_0^2}\right) && (6.38) \\ &\simeq -p_0 + \frac{1}{2}\frac{p_x^2 + p_y^2}{p_0} + \frac{qI_0}{c^2}\left(\frac{x^2+y^2}{R_0^2}\right), && (6.39) \end{aligned}$$

where I have eliminated p_t in favor of the total momentum,

$$p_0 := \sqrt{\frac{1}{c^2}p_t^2 - m^2c^2},$$

since $p_t = -E$ is a constant of the motion. For small transverse momenta, $p_0 \simeq p_z$.

For $|p_x|$ and $|p_y|$ small compared to $|p_0|$ one sees that the particle indeed executes simple harmonic motion in x and y as a function of z, with wave-number

$$k^2 = 2\frac{qI_0}{p_0 c^2 R_0^2},$$

The phase-plane orbits are therefore concentric ellipses in this approximation; particles progress "clockwise" about the ellipse as z increases.

Now consider a beam striking a production target. The distribution of particles produced will have the same transverse dimensions as the primary beam, but the transverse momentum spread will increase. Therefore, at a production target, one desires that the momentum spread therefore as large as possible, (and therefore, by (6.34), that the beam be as small as possible), so the increase in emittance will be minimized. This occurs at what is evocatively known as a "waist". At a waist, the beam also satisfies the condition $\langle x p_x \rangle = 0$, in which case the ellipse is said to be "upright". When the ellipse is upright, its major and minor axes are aligned with the coordinate axes. One also has $\langle x p_x \rangle = 0$ when the momentum spread is minimized (and the beam size therfore maximized) this might be called a "momentum waist", or an "antiwaist".

On the other hand, the maximum transverse momentum a beam transport line can accept is limited — generally, it is smaller than the transverse momentum spread emerging from the target. It is therefore necessary to transform, or "match", the emittance of the beam to the acceptance of the beamline. This is the function of the lithium lens.

First of all, one ideally desires the lens to be as close to the production target as possible ("minimum drift space"), so the spray of particles produced in the target will not have time to spread out. From (6.7), one sees that the effect of a drift is to "shear" the ellipse; for each particle, p_x is constant in a drift, while x translates at a rate proportional to p_x. Therefore, while a somewhat weaker and/or shorter lens can be used, it must have a larger diameter or particles will be lost. Furthermore, since the orbits within the lens are ellipses, and the area of a phase-plane ellipse is related to the emittance, a "stand off" distance puts the most extreme particles on orbits of larger "effective emittance" (emittance of a matched ellipse). Nonlinear effects will lead to filamentation and mixing, causing the beam to more nearly fill its effective beam ellipse; this is to be avoided.

Finally, for a target of non-zero length, the differences in distance to the lens for particles produced at various points along the target results in their respective ellipses being sheared by varying amounts. The result is a characteristic "bow tie" or "butterfly" pattern for the phase-plane distribution. The longer the particles drift, the larger the phase ellipse required to enclose this odd shape.

The ultimate limit, of course, would be to actually have the lens surround the target like a blanket. Alternatively, one can run current through the target itself [Aut83].

If the current is adjusted so the lens is approximately a half-wavelength long, the particles produced in the target will then more or less uniformly fill a phase-space ellipse in both the x–p_x and y–p_y planes. The ellipse so produced still has the same large maximum transverse momentum, however the maximum displacement is now on the order of a centimeter, rather than a millimeter or less. The aspect ratio of the ellipse is therefore much closer to that of the acceptance ellipse of a typical beam transport line, and hence easier to match.

Unfortunately, practical limitations on the current-density exist, due to heating, thermomechanical, or magnetomechanical effects. Practical current-densities may make such an ideal "half-wave transformer" impractical, because the target will have to be too long, resulting in too much antiproton absorbtion (targets are typically made of high-Z metals, such as copper, tungsten, or tungsten-rhenium alloy). This is particularly true if one is using the target itself as the conducting medium.

For short targets, the ideal length is therfore approximately a quarter wavelength, since the momentum spread will then be a minimum; if the lens is close to linear (*i.e.*, $p_x \ll p_z$), and surrounds the target along its full length, the ideal length works out to be:

$$L_{lens} = \frac{\lambda}{4} + \frac{L_{target}}{2}.$$

Nonlinear effects will modify this, since the orbits will now be neither precisely elliptical, nor the wavenumber amplitude independent.

To obtain the nondimensional form of the Hamiltonian, I choose the rescaling:

$$x = \ell X, \qquad y = \ell Y, \qquad p_x = p_\perp P_X, \qquad p_y = p_\perp P_Y, \qquad z = LZ,$$

where ℓ is a transverse scale length, L is a longitudinal scale length, and $p_\perp$ is a transverse scale momentum. The Hamiltonian then becomes:

$$H = -\left(\frac{L}{\ell}\frac{p_0}{p_\perp}\right)\left[1 - \frac{p_\perp^2}{p_0^2}\left(P_X^2 + P_Y^2\right)\right]^{\frac{1}{2}} + \frac{1}{2}\frac{L}{\ell}\frac{p_0}{p_\perp}\frac{B_0 R_0}{B\rho}\frac{\ell^2}{R_0^2}\left(X^2 + Y^2\right). \tag{6.40}$$

An alternative parameterization is

$$H = -\left(\frac{L}{\ell}\frac{p_0}{p_\perp}\right)\left[1 - \frac{p_\perp^2}{p_0^2}\left(P_X^2 + P_Y^2\right)\right]^{\frac{1}{2}} + \frac{L}{\ell}\frac{p_0}{p_\perp}\frac{q}{e}\frac{m_e c}{p_0}\frac{I_0}{I_A}\frac{\ell^2}{R_0^2}\left(X^2 + Y^2\right). \tag{6.41}$$

Here I have introduced the quantities $B_0 := B_\theta(R_0)$, the magnetic induction at $r = R_0$, the "magnetic rigidity",

$$B\rho := \frac{p_0 c}{q} = (3335.6)\,\frac{p_0}{1\,\text{GeV}}\,[\text{kGauss}\cdot\text{cm}].$$

and the "Alfvén current",

$$I_A := \frac{ec}{r_{cl}} = \frac{m_e c^3}{e} \simeq 17.05\,\text{kA}$$

(m_e is the electron mass, and e the electron charge). Please note that $B\rho$ is a compound symbol, and should be read as a single unit; note also that it is *not* an independent scale, but is simply proportional to the momentum. The advantage of the parameterizations (6.40) and (6.41) is that they are manifestly dimensionless, and allow one to easily study the cases $B_0 = \{const\}$ and $I_0 = \{const\}$.

In my case-study I chose to use the following parameters, inspired by the FermiLab antiproton source [Peo89]:

$$\begin{aligned} \ell = L &= 1\,\mathrm{cm} \\ p_\perp &= 1.0\,\mathrm{GeV} \\ p_0 &= 8.89\,\mathrm{GeV} \\ R_0 &= 1.0\,\mathrm{cm} \\ B_0 &= 100\,\mathrm{kGauss} \Longrightarrow I_0 = 500\,\mathrm{kA} \\ L_{target} &= 5\,\mathrm{cm} \end{aligned}$$

With the above parameters, $\lambda/4 \simeq 27.05\,\mathrm{cm}$.

This problem is complicated by a distributed source of particles. I initially attempted to optimize the lens by dividing the target up into "slices", integrating from each slice through the lens, and then tracking a swarm of particles through the lens for each slice. Since the Hamiltonian is independent of z in this case, I do not need to do the integrals separately, but can acculmulate them by "automatic composition", *i.e.*, restart the integral with the x, R, and DS obtained from the previous slice. This method proved to have much too large statistical fluctuations even for large numbers of particles.

Therefore, I used "moment transport" instead. In the case of a distributed source, the solution to the moment transport equation becomes

$$\langle P^A \rangle(t_2) := \int_{t_1}^{t_2} \int P^A(\mathcal{M}(t)\boldsymbol{\xi})\dot{f}(\boldsymbol{\xi};t)\, d^{2d}\xi\, dt. \tag{6.42}$$

where $\dot{f}$ is the source term in the transport equation.

I solved for the map as before, transporting moments through tenth order from each of ten "slices" to the end of the lens. I did the integral over the source via Simpson's rule. Since the current HJ/DA code was intended for "proof of principle" only, I did not make provision for including an optimizing package. I therefore employed a "brute force" approach of simply stepping the length of the lens through a near $\lambda/4$, observing which subinterval had the minimum beam divergence (*i.e.*, minimum $\langle p_x^2 \rangle$), then stepping through that subinterval around the minimum, *etc.*, until the minimum was located to four significant figures.

The minimum beam divergence was found to be at $L_{lens} \simeq (1.0826)\times\lambda/4$; for comparison, the theoretical value for the minimum is

$$L_{lens} = \frac{\lambda}{4} + \frac{1}{2}L_{target} \simeq (1.0924)\times\lambda/4 = 29.549\,\mathrm{cm}.$$

Therefore the effect of including nonlinearities is to make the lens about 1% shorter; the magnitude of this effect is consistent with the fact that the assumed transverse momentum is about 10% of the longitudinal momentum, and that only even powers of $p_\perp/p_{||}$ enter into the Taylor expansion of the Hamiltonian. The effects of nonlinearity are therefore fairly small, perhaps obviating the entire calculation. Numerical results are contained in app. E. Fig. F-32 gives a scatter-plot of the final x–p_x phase-space distribution, together with the 50% and 90% contours (the 14.7% contour is too small to be shown). Note that the ellipse has no visible "tilt" (the fitting routine gave the angle between the major axis and the X axis as 0.03°). Therefore the desired "upright" beam distribution has been very nearly achieved. The "uprightness" of the beam is confirmed by the value of $\langle XP_X \rangle$ which, according to table E-1, appears to cross through zero somewhere quite close to the minimum value found for $\langle P_X^2 \rangle$; hence the beam does seem to be passing through an "antiwaist". Despite this, there would appear to be a small but significant asymmetry to the scatter plot — the upper left and lower right "wingtips" of the butterfly seem to be slightly longer and more pointed than the other two. Perhaps this asymmetry represents the begining of the filamentation process.

Finally, it is perhaps interesting to note that a near cancellation occurs between the contributions made by the fourth- and sixth-order moments to $\langle P_X^2 \rangle$ over the whole range investigated; the difference between them is comparable to the eighth order contributions. The tenth order effects are an order of magnitude smaller than the eighth order effects.

Chapter 7

Conclusion

I have presented a new method, the HJ/DA equation, for calculating the perturbative transfer map of an arbitrary Hamiltonian system, using the new method of Differential Algebra. The results of Chapter 6 demonstrate that it indeed functions as advertised: one needs only write a FORTRAN routine for calculating the Hamiltonian, run it through the DAFOR preprocessor, integrate it, and one has an implicit representation of the map though arbitrarily high order, which will be accurate, and *exactly symplectic*, except for the linear part (which is still quite accurate). I also presented methods based on F_1 and F_2 generating functions which will be exactly symplectic for the linear part as well. Using the tools of the DA-library, one can then convert them into whatever representation one desires, for use in analysis or tracking programs. I have not yet compared computational efficiency to competitive methods, however my experience in working with the DA-package *vs* other packages suggests that it will be comparable, and quite possibly superior. With the inclusion of automatic step-size and order control into the integrator, and the use of a symplectic integration algorithm to obtain the reference trajectory and linear transfer matrix, the potential for a fast, general, fully symplectic analysis program for arbitrary nonlinear systems, to arbitrarily high order, now appears to exist.

Bibliography

[AMR88] R. Abraham, J. E. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications.* Springer-Verlag, New York, second edition, 1988.

[Arn88] V. I. Arnol'd. *Mathematical Methods of Classical Mechanics.* Springer-Verlag, New York, 1988.

[Art57] E. Artin. *Geometric Algebra.* Interscience, New York, 1957.

[Aut83] B. Autin. Technical developments for an antiproton collector at cern. In Francis T. Cole and Rene Donaldson, editors, *Proc. 12th International Conf. on High-Energy Particle Accelerators*, pages 393–396. Fermi National Accelerator Laboratory, 1983. Held at Fermilab, Aug 11-16.

[Ber87] M. Berz. The method of power series tracking for the mathematical description of beam dynamics. *Nucl. Inst. and Methods*, A258:431–436, 1987. Presented at the 2nd International Conference on Charged Particle Optics.

[Ber88] M. Berz. Differential algebraic treatment of beam dynamics to very high orders, including applications to spacecharge. In Charles R. Eminhizer, editor, *Linear Accelerator and Beam Optics Codes: Proccedings of the 1988 La Jolla Institute*, pages 275–300. American Institute of Physics, 1988. AIP Conf. Proc. no. 177.

[Ber89a] M. Berz. Description of particle accelerators using high-order perturbation theory on maps. In Melvin Month and Margaret Dienes, editors, *Physics of*

Particle Accelerators: Fermilab Summer School 1987, Cornell Summer School 1988, pages 961–994. American Institute of Physics, 1989. AIP Conference Proceedings no. 184.

[Ber89b] M. Berz. Differential algebraic description of beam dynamics to very high orders. *Particle Accelerators*, 24:108–125, 1989.

[BHW87] M. Berz, H. C. Hofmann, and H. Wollnik. COSY 5.0, the fifth order code for corpuscular optical systems. *Nucl. Inst. and Methods*, A258:402, 1987. Presented at the 2nd International Conference on Charged Particle Optics.

[BrzWln87] M. Berz and H. Wollnik. The program HAMILTON for the analytic solution of the equations of motion in particle optical systems through fifth order. *Nuclear Instruments and Methods*, A258:364, 1987. Presented at the 2nd International Conference on Charged Particle Optics.

[BrnWlf70] M. Born and E. Wolf. *Principles of Optics*. Pergamon Press, Ltd., Oxford, fourth edition, 1970.

[Bru95] H. Bruns. . *Lepz. Sitzungsber.*, 21:321, 1895.

[BS66] R. Bulirsch and J. Stoer. Numerical treatment of ordinary differential equations by extrapolation methods. *Num. Math.*, 8:1–13, 1966.

[Bur85] William L. Burke. *Applied Differential Geometry*. Cambridge University Press, Cambridge, New York, Melborne, 1985.

[Bro77] K. Brown *et al.* Transport, a computer program for designing charged particle beam transport systems. Technical Report SLAC-91, Stanford Linear Accelerator Center, 1977.

[Car65] C. Carathéodory. *Calculus of Variations and Partial Differential Equations of the First Order*, volume 1–2. Holden-Day, San Francisco, London, Amsterdam, 1965.

[Car81] John R. Cary. Lie transform perturbation theory for hamiltonian systems. *Physics Reports*, 79:129–159, 1981.

[Cha83] Paul J. Channell. Symplectic integration algorithms. AT-Division Technote AT6:ATN-83-9, Los Alamos National Laboratory, April 1983.

[CHMM78] A. Chorin, T. J. R. Hughes, M. F. McCracken, and J. Marsden. Product formulas and numerical algorithms. *Commun. Pure Appl. Math.*, 31:205–256, 1978.

[ChSc88] Paul J. Channell and J. C. Scoval. Symplectic integration of hamiltonian systems. Technical Report LA-UR-88-1828, Los Alamos National Laboratory, 1988.

[ConSyn31] A. W. Conway and J. L. Synge, editors. *The Mathematical Papers of Sir W. R. Hamilton.* Cambridge University Press, Cambridge, 1931.

[Deu83] P. Deuflhard. Order and stepsize control in extrapolation methods. *Num. Math.*, 41:399–422, 1983.

[DeV56] R. De Vogelaére. Methods of integration which preserve the contact transformation property of the hamiltonian equations. Technical Report Report 4, Dept. of Mathematics, University of Notre Dame, 1956.

[DeW76] Cecile DeWitt-Morette. The semiclassical expansion. *Ann. Phys.*, 97:367–399, 1976.

[DV79] G. Dangelmayr and W. Veit. Semiclassical approxiations of path integrals on and near caustics in terms of catastrophes. *Ann. Phys.*, 118:108–138, 1979.

[DF76] Alex J. Dragt and John M. Finn. Lie series and invariant functions for analytic symplectic maps. *J. Math. Phys.*, 17:2215–2227, 1976.

[DF83] Alex J. Dragt and Etienne Forest. Computation of nonlinear behavior of hamiltonian systems using lie algebraic methods. *J. Math. Phys.*, 24:2734–2744, 1983.

[DNR88a] Alex J. Dragt, F. Neri, and G. Rangarajan. Lie algebraic treatment of moments and moment invariants. Technical report, Dept. of Physics and Astronomy, University of Maryland, 1988.

[DNR88b] Alex J. Dragt, Filippo Neri, and Govindan Rangarajan. Lie algebraic treatment of linear and nonlinear beam dynamics. *Ann. Rev. Nucl. Part. Sci.*, 38:455–96, 1988.

[Dra82] Alex J. Dragt. Lectures on nonlinear orbit dynamics. In *et al* R. A. Carrigan, editor, *1981 Fermilab Summer School on High Energy Particle Accelerators*, pages 147–227. American Institute of Physics, 1982. AIP Conference Proceedings no. 87.

[Dra87] Alex J. Dragt. Elementary and advanced lie algebraic methods with applications to accelerator design, electron microscopes, and light optics. *Nucl. Inst. and Methods*, A258:339–354, 1987. Presented at the 2^{nd} International Conference on Charged Particle Optics.

[FDL87] Etienne Forest, David Douglas, and Beat Leemann. Study of the aberrations of a periodic arc using lie algebraic techniques. *Nucl. Inst. and Methods*, A258:355–363, 1987. Presented at the 2^{nd} International Conference on Charged Particle Optics.

[For84] Etienne Forest. *Lie Algebraic Methods for Charged Particle Beams and Light Optics.* PhD thesis, Dept. of Physics and Astronomy, University of Maryland, 1984.

[For87] Etienne Forest. Lie algebraic maps and invariants produced by tracking codes. *Particle Accelerators*, 22:15–34, 1987.

[For89] Etienne Forest. Normal form methods for complicated periodic systems: A complete solution using differential algebra and lie operators. *Particle Accelerators*, 24:91–108, 1989.

[Gib60] A. Gibbons. A program for the automatic integration of differential equations using the method of ttaylor series. *Computer J.*, 3:108–111, 1960.

[Gla33] W. Glaser. *Z. Phys.*, 80:451, 1933.

[Gol80] Herbert Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, Massachusetts, second edition, 1950, 1980.

[Hea86] Liam Michael Healy. *Lie Algebraic Methods for Treating Lattice Parameter Errors in Particle Accelerators.* PhD thesis, Dept. of Physics and Astronomy, University of Maryland, 1986.

[HH66] M. Hénon and C. Heiles. The applicability of the third integral of motion: Some numerical experiments. *Astron. J.*, 71:670, 1966.

[HNW87] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I: Nonstiff problems.* Springer-Verlag, Berlin, Hiedelberg, New York, 1987.

[How74] Bernard E. Howard. Phase space analysis in numerical integration of ordinary differential equations. In Dale G. Bettis, editor, *Proc. of the Conf. on the Numerical Solution of Ordinary Differential Equations*, volume 362 of *Lecture Notes in Mathematics*, pages 109–127. Springer-Verlag, 1974. Conf. held at U.T. Austin, Oct. 1972.

[Jac63] C. G. J. Jacobi. Jur theorie der variations-rechnung und der differentialgleichungen. *Akad d. Wiss. zu Berlin*, 5:41–55, Nov 1863.

[Jer89] Max E. Jerrel. Differentiation in PASCAL-SC: Type gradient. *ACM Transactions on Mathematical Software*, 24:2–7, 1989.

[Kir70] Donald E. Kirk. *Optimal Control Theory: an Introduction.* Prentice-Hall, Englewood Cliffs, New Jersey, 1970.

[KW69] H. Knapp and G. Wanner. LIESE II, aprogram for ordinary differential equations using lie series. MRC Report 1008, Math. Research Center, Univ. Wisconsin, Madison, Wisc 53706, 1969.

[LLa] L. D. Landau and E. M. Lifshitz. *Classical Mechanics*, volume 1 of *Course of Theoretical Physics*. Pergamon Press, Ltd., New York.

[LLb] L. D. Landau and E. M. Lifshitz. *Physical Kinetics*, volume 10 of *Course of Theoretical Physics*. Pergamon Press, Ltd., New York.

[LLG73] J. D. Lawson, P. M. Lapostolle, and R. L. Gluckstern. Emittance entropy and information. *Particle Accelerators*, 5:61–65, 1973.

[LN88] C.A. Lucey and E.T. Newman. On the construction of hamiltonians. *J. Math. Phys.*, 29:2430–2433, 1988.

[LO88] W. P. Lysenko and M. S. Overley. Moment invariants for particle beams. In Charles R. Eminhizer, editor, *Linear Accelerator and Beam Optics Codes: Proccedings of the 1988 La Jolla Institute*, pages 323–335. American Institute of Physics, 1988. AIP Conf. Proc. no. 177).

[Moo66] R. E. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1966.

[Mos66] Ronald W. Moses. Extension of schertzer's theorem. *Rev. Sci. Instr.*, 37:1370–1372, 1966.

[Ner86] F. Neri. Notes on symplectification through fifth order. Technical report, Dept. of Physics and Astronomy, University of Maryland, 1986.

[Ner87] F. Neri. Lie algebras and canonical integration. Technical report, Dept. of Physics and Astronomy, University of Maryland, 1987.

[Olv86] Peter J. Olver. *Applications of Lie Groups to Differential Equations.* Springer-Verlag, New York, 1986.

[Omo86] Stephen M. Omohundro. *Geometrical Perturbation Theory in Physics.* World Scientific Publishing, Singapore, 1986.

[ON79] Van F. Oystaeyen and C. Nastasescu. *Graded and Filtered Rings and Modules.* Number 758 in Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1979.

[ON82] Van F. Oystaeyen and C. Nastasescu. *Graded Ring Theory.* Library of Mathematics. North-Holland, Amsterdam, 1982.

[Peo89] J. Peoples. The tevatron: Antiproton source. In Melvin Month and Margaret Dienes, editors, *Physics of Particle Accelerators: Fermilab Summer School 1987, Cornell Summer School 1988*, pages 1845–1877. American Institute of Physics, 1989. AIP Conference Proceedings no. 184.

[PFTV86] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, New York, Melborne, 1986.

[Ral84] L. B. Rall. Automatic differentiation using almost any language. *Signum Newsletter*, 10:161–184, 1984.

[Ran] G. Rangarajan. Dissertation research (in progress).

[Rit50] Joseph Fels Ritt. *Differential Algebra*, volume 33 of *AMS Colloquium Publications.* American Mathematical Society, New York, 1950.

[Rob61] A. Robinson. Non-standard analysis. *Proc. Royal Acad. Amsterdam*, A:432, 1961.

[Ros87] H. Rose. Hamiltonian magnetic optics. *Nucl. Inst. and Methods*, A258:374–401, 1987. Presented at the 2nd International Conference on Charged Particle Optics.

[RRW85] R. D. Ruth, T. Raubenheimer, and R. L. Warnock. Superconvergent tracking and invariant surfaces in phase space. *IEEE Trans. Nucl. Sci.*, NS-32:2206–2208, Oct 1985.

[Run66] Hano Rund. *Hamilton-Jacobi Theory and the Calulus of Variations.* Van Nostrand, New York, 1966.

[Rut83] Ronald D. Ruth. A canonical integration technique. *IEEE Trans. Nucl. Sci.*, NS-30:2669–2671, Aug 1983.

[Ryn87] Robert Douglas Ryne. *Lie Algebraic Treatment of Space Charge.* PhD thesis, Dept. of Physics and Astronomy, University of Maryland, 1987.

[Sag67] Andrew P. Sage. *Optimum Systems Control.* Prentice-Hall, Englewood Cliffs, New Jersey, 1967.

[Sch36] O. Schertzer. *Z. Phys.*, 101:593, 1936.

[Ser85] Roger V. Servranckx. Improved tracking codes: Present and future. *IEEE Trans. Nucl. Sci.*, NS-32:2186–2190, Oct 1985.

[SM74] E. C. G. Sudarshan and N. Mukunda. *Classical Dynamics: A Modern Perspective.* Wiley-Interscience, New York, 1974.

[Ste56] J. F. Steffenson. On the restricted problem of three bodies. *K. danske Vidensk. Selsk., Mat-fys. Medd. 30 Nr. 18*, 1956.

[Stu52] P. A. Sturrock. Perturbation characteristic functions and their applications to electron optics. *Proc. Roy Soc.*, A210:269–289, 1952.

[SW86] D H. Sattinger and O. L. Weaver. *Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics.* Springer-Verlag, New York, 1986.

[Syn37] John L. Synge. *Geometrical Optics*, volume 37 of *Cambridge Tracts in Mathematics and Mathematical Physics.* Cambridge University Press, Cambrige, 1937.

[Syn51] John L. Synge. *Hamilton's Method in Geometrical Optics*, volume 9 of *University of Maryland Lecture Series*. University of Maryland, 1951.

[Syn53] John L. Synge. *Geometrical Mechanics of De Broglie Waves*. Cambridge University Press, Cambridge, 1953.

[Syn60] John L. Synge. *Relativity: The General Theory*. Cambridge University Press, Cambridge, 1960.

[WR87] R. L. Warnock and R. D. Ruth. Invariant tori through direct solution of the hamilton-jacobi equation. *Physica*, 26D:1–36, 1987.

Appendix A

Numerical Results — Drift to 12^{th} Order

The Hamiltonian used for this run was the "relativistic drift". Initial conditions were as follows:

```
%
% HJDAdrift: Particle in a Uniform Relativistic Drift.
%
%         no =    6
%      nstep =   20
%      istep =    5
%        eps =  1.000000E-30
%
%        t_i =  0.000000E+00
%        x_i =  0.000000E+00
%       Pt_i = -2.000000E+00 ( = - Gamma0 )
%       Px_i =  0.000000E-01
%
%        z_i =  0.000000E+00
%        z_i =  1.000000E+00
%
```

Table A-1: Difference between $R_{2,numer}$ and $R_{2,theo}$

```
 r2,numer
=============================================================
 1.000000E+00,  0.000000E+00,  1.924501E-01,  0.000000E+00
 0.000000E+00,  1.000000E+00,  0.000000E+00,  5.773503E-01
 0.000000E+00,  0.000000E+00,  1.000000E+00,  0.000000E+00
 0.000000E+00,  0.000000E+00,  0.000000E+00,  1.000000E+00
```

Table A-2: Numerically determined R_2

```
 r2,theo
=============================================================
 1.000000E+00,  0.000000E+00,  1.924501E-01,  0.000000E+00
 0.000000E+00,  1.000000E+00,  0.000000E+00,  5.773503E-01
 0.000000E+00,  0.000000E+00,  1.000000E+00,  0.000000E+00
 0.000000E+00,  0.000000E+00,  0.000000E+00,  1.000000E+00
```

Table A-3: Theoretically Determined R_2

```
 r2,numer - r2,theo
=============================================================
 0.000000E+00,  0.000000E+00,  3.816392E-17,  0.000000E+00
 0.000000E+00,  0.000000E+00,  0.000000E+00,  8.326673E-17
 0.000000E+00,  0.000000E+00,  0.000000E+00,  0.000000E+00
 0.000000E+00,  0.000000E+00,  0.000000E+00,  0.000000E+00
```

Table A-4: Numerical *vs* Theoretical DS_{12}

```
A = DS12         NO =  12, NV =    4, ina = 36
B = DSTHEO       NO =  12, NV =    4, inb = 71
********************************************************************************
  I  A Coefficient B Coefficient  Difference     Frac. Diff.    Order Exponents

  ALL ORDER =    0 COMPONENTS ZERO
  ALL ORDER =    1 COMPONENTS ZERO

  1   1.000000E+00  1.000000E+00   0.000000E+00  0.000000E+00    2    1 0 1 0
  2   1.000000E+00  1.000000E+00   0.000000E+00  0.000000E+00    2    0 1 0 1

  3  -6.415003E-02 -6.415003E-02  -2.081668E-17 -1.622500E-16    3    0 0 3 0
  4  -1.924501E-01 -1.924501E-01  -9.020562E-17 -2.343611E-16    3    0 0 1 2

  5  -4.543960E-02 -4.543960E-02  -2.515349E-17 -2.767794E-16    4    0 0 4 0
  6  -1.443376E-01 -1.443376E-01  -1.040834E-17 -3.605555E-17    4    0 0 2 2
  7  -2.405626E-02 -2.405626E-02  -1.127570E-17 -2.343611E-16    4    0 0 0 4

  8  -3.385696E-02 -3.385696E-02  -2.688821E-17 -3.970855E-16    5    0 0 5 0
  9  -1.176084E-01 -1.176084E-01  -6.938894E-17 -2.950000E-16    5    0 0 3 2
 10  -4.811252E-02 -4.811252E-02  -6.071532E-18 -6.309721E-17    5    0 0 1 4

 11  -2.628369E-02 -2.628369E-02  -1.040834E-17 -1.980000E-16    6    0 0 6 0
 12  -1.011254E-01 -1.011254E-01  -9.367507E-17 -4.631629E-16    6    0 0 4 2
 13  -6.815941E-02 -6.815941E-02   5.204170E-18  3.817647E-17    6    0 0 2 4
 14  -4.009377E-03 -4.009377E-03  -2.168404E-19 -2.704166E-17    6    0 0 0 6

 15  -2.108635E-02 -2.108635E-02   3.469447E-18  8.226759E-17    7    0 0 7 0
 16  -8.998824E-02 -8.998824E-02   6.418477E-17  3.566287E-16    7    0 0 5 2
 17  -8.464240E-02 -8.464240E-02  -7.979728E-17 -4.713789E-16    7    0 0 3 4
 18  -1.336459E-02 -1.336459E-02  -6.071532E-18 -2.271500E-16    7    0 0 1 6

 19  -1.736778E-02 -1.736778E-02  -1.257675E-17 -3.620712E-16    8    0 0 8 0
 20  -8.189523E-02 -8.189523E-02  -5.204170E-17 -3.177334E-16    8    0 0 6 2
 21  -9.856385E-02 -9.856385E-02  -2.949030E-17 -1.496000E-16    8    0 0 4 4
 22  -2.784289E-02 -2.784289E-02   1.170938E-17  2.102760E-16    8    0 0 2 6
 23  -8.352868E-04 -8.352868E-04  -3.794708E-19 -2.271500E-16    8    0 0 0 8

 24  -1.461168E-02 -1.461168E-02   5.637851E-18  1.929228E-16    9    0 0 9 0
 25  -7.568318E-02 -7.568318E-02   6.591949E-17  4.354963E-16    9    0 0 7 2
 26  -1.107034E-01 -1.107034E-01   2.775558E-17  1.253601E-16    9    0 0 5 4
 27  -4.677606E-02 -4.677606E-02  -4.597017E-17 -4.913856E-16    9    0 0 3 6
 28  -3.898005E-03 -3.898005E-03   3.306817E-18  4.241678E-16    9    0 0 1 8

 29  -1.250742E-02 -1.250742E-02  -3.252607E-18 -1.300271E-16   10    0 010 0
 30  -7.071786E-02 -7.071786E-02   1.214306E-17  8.585571E-17   10    0 0 8 2
 31  -1.215745E-01 -1.215745E-01  -4.683753E-17 -1.926290E-16   10    0 0 6 4
 32  -6.951443E-02 -6.951443E-02   0.000000E+00  0.000000E+00   10    0 0 4 6
 33  -1.071951E-02 -1.071951E-02   2.168404E-19  1.011428E-17   10    0 0 2 8
 34  -1.949003E-04 -1.949003E-04   7.115077E-20  1.825312E-16   10    0 0 010

 35  -1.086026E-02 -1.086026E-02  -7.589415E-18 -3.494123E-16   11    0 011 0
 36  -6.662747E-02 -6.662747E-02   3.122502E-17  2.343254E-16   11    0 0 9 2
 37  -1.315051E-01 -1.315051E-01   3.122502E-17  1.187217E-16   11    0 0 7 4
 38  -9.554444E-02 -9.554444E-02  -9.194034E-17 -4.811392E-16   11    0 0 5 6
 39  -2.273836E-02 -2.273836E-02  -7.806256E-18 -1.716539E-16   11    0 0 3 8
 40  -1.169402E-03 -1.169402E-03  -5.421011E-19 -2.317857E-16   11    0 0 110

 41  -9.543445E-03 -9.543445E-03  -7.372575E-18 -3.862638E-16   12    0 012 0
 42  -6.317944E-02 -6.317944E-02   1.387779E-17  1.098283E-16   12    0 010 2
 43  -1.407085E-01 -1.407085E-01   7.632783E-17  2.712269E-16   12    0 0 8 4
 44  -1.244835E-01 -1.244835E-01  -1.734723E-18 -6.967683E-18   12    0 0 6 6
 45  -4.122682E-02 -4.122682E-02   3.469447E-18  4.207755E-17   12    0 0 4 8
 46  -3.995455E-03 -3.995455E-03  -3.577867E-18 -4.477421E-16   12    0 0 210
 47  -4.872507E-05 -4.872507E-05   1.778769E-20  1.825312E-16   12    0 0 012
```

Table A-5: Numerical *vs* Theoretical Dt_1

```
A = DTC1        NO =  12, NV =   4, ina = 54
B = TTC1        NO =  12, NV =   4, inb = 62
********************************************************************************
  I  A Coefficient B Coefficient  Difference     Frac. Diff.    Order Exponents

  1                1.665335E-16                                   0    0 0 0 0

  2   1.000000E+00  1.000000E+00  0.000000E+00  0.000000E+00      1    1 0 0 0
  3                 2.775558E-17                                  1    0 0 1 0

  4  -1.924501E-01 -1.924501E-01  -6.591949E-17 -1.712639E-16     2    0 0 2 0
  5  -1.924501E-01 -1.924501E-01  -9.367507E-17 -2.433750E-16     2    0 0 0 2

  6  -1.817584E-01 -1.817584E-01  -7.979728E-17 -2.195147E-16     3    0 0 3 0
  7  -2.886751E-01 -2.886751E-01  -2.775558E-17 -4.807407E-17     3    0 0 1 2

  8  -1.692848E-01 -1.692848E-01  -1.526557E-16 -4.508841E-16     4    0 0 4 0
  9  -3.528252E-01 -3.528252E-01  -2.012279E-16 -2.851666E-16     4    0 0 2 2
 10  -4.811252E-02 -4.811252E-02  -7.806256E-18 -8.112499E-17     4    0 0 0 4

 11  -1.577022E-01 -1.577022E-01  -3.469447E-17 -1.100000E-16     5    0 0 5 0
 12  -4.045016E-01 -4.045016E-01  -3.885781E-16 -4.803171E-16     5    0 0 3 2
 13  -1.363188E-01 -1.363188E-01   1.040834E-17  3.817647E-17     5    0 0 1 4

 14  -1.476045E-01 -1.476045E-01   4.510281E-17  1.527827E-16     6    0 0 6 0
 15  -4.499412E-01 -4.499412E-01   3.330669E-16  3.701227E-16     6    0 0 4 2
 16  -2.539272E-01 -2.539272E-01  -2.428613E-16 -4.782105E-16     6    0 0 2 4
 17  -1.336459E-02 -1.336459E-02  -5.854692E-18 -2.190375E-16     6    0 0 0 6

 18  -1.389422E-01 -1.389422E-01  -7.979728E-17 -2.871599E-16     7    0 0 7 0
 19  -4.913714E-01 -4.913714E-01  -3.261280E-16 -3.318549E-16     7    0 0 5 2
 20  -3.942554E-01 -3.942554E-01  -1.040834E-16 -1.320000E-16     7    0 0 3 4
 21  -5.568579E-02 -5.568579E-02   2.341877E-17  2.102760E-16     7    0 0 1 6

 22  -1.315051E-01 -1.315051E-01   3.122502E-17  1.187217E-16     8    0 0 8 0
 23  -5.297822E-01 -5.297822E-01   4.302114E-16  4.060267E-16     8    0 0 6 2
 24  -5.535168E-01 -5.535168E-01   1.526557E-16  1.378962E-16     8    0 0 4 4
 25  -1.403282E-01 -1.403282E-01  -1.387779E-16 -4.944761E-16     8    0 0 2 6
 26  -3.898005E-03 -3.898005E-03   3.252607E-18  4.172142E-16     8    0 0 0 8

 27  -1.250742E-01 -1.250742E-01  -1.561251E-16 -6.241301E-16     9    0 0 9 0
 28  -5.657429E-01 -5.657429E-01   1.249001E-16  1.103859E-16     9    0 0 7 2
 29  -7.294467E-01 -7.294467E-01  -2.914335E-16 -1.997634E-16     9    0 0 5 4
 30  -2.780577E-01 -2.780577E-01   0.000000E+00  0.000000E+00     9    0 0 3 6
 31  -2.143903E-02 -2.143903E-02   1.301043E-18  3.034285E-17     9    0 0 1 8

 32  -1.194628E-01 -1.194628E-01   3.122502E-17  1.306893E-16    10    0 010 0
 33  -5.996473E-01 -5.996473E-01   1.249001E-16  1.041446E-16    10    0 0 8 2
 34  -9.205356E-01 -9.205356E-01   2.220446E-16  1.206062E-16    10    0 0 6 4
 35  -4.777222E-01 -4.777222E-01  -4.718448E-16 -4.938485E-16    10    0 0 4 6
 36  -6.821509E-02 -6.821509E-02  -2.428613E-17 -1.780114E-16    10    0 0 2 8
 37  -1.169402E-03 -1.169402E-03  -5.421011E-19 -2.317857E-16    10    0 0 010

 38  -1.145213E-01 -1.145213E-01  -1.925543E-16 -8.406918E-16    11    0 011 0
 39  -6.317944E-01 -6.317944E-01   9.714451E-17  7.687984E-17    11    0 0 9 2
 40  -1.125668E+00 -1.125668E+00   6.661338E-16  2.958839E-16    11    0 0 7 4
 41  -7.469011E-01 -7.469011E-01   0.000000E+00  0.000000E+00    11    0 0 5 6
 42  -1.649073E-01 -1.649073E-01   1.734723E-17  5.259693E-17    11    0 0 3 8
 43  -7.990911E-03 -7.990911E-03  -7.155734E-18 -4.477421E-16    11    0 0 110
```

Table A-6: Numerical *vs* Theoretical Dx_1

```
A = DXC1        NO =  12, NV =   4, ina = 55
B = TXC1        NO =  12, NV =   4, inb = 63
********************************************************************************
  I  A Coefficient B Coefficient  Difference     Frac. Diff.   Order Exponents

  ALL ORDER =   0 COMPONENTS ZERO

  1    1.000000E+00  1.000000E+00   0.000000E+00  0.000000E+00    1    0 1 0 0
  2                  8.326673E-17                                 1    0 0 0 1

  3   -3.849002E-01 -3.849002E-01  -1.873501E-16 -2.433750E-16    2    0 0 1 1

  4   -2.886751E-01 -2.886751E-01  -2.775558E-17 -4.807407E-17    3    0 0 2 1
  5   -9.622504E-02 -9.622504E-02  -4.683753E-17 -2.433750E-16    3    0 0 0 3

  6   -2.352168E-01 -2.352168E-01  -1.318390E-16 -2.802500E-16    4    0 0 3 1
  7   -1.924501E-01 -1.924501E-01  -3.122502E-17 -8.112499E-17    4    0 0 1 3

  8   -2.022508E-01 -2.022508E-01  -1.942890E-16 -4.803171E-16    5    0 0 4 1
  9   -2.726376E-01 -2.726376E-01   2.081668E-17  3.817647E-17    5    0 0 2 3
 10   -2.405626E-02 -2.405626E-02  -2.168404E-18 -4.506944E-17    5    0 0 0 5

 11   -1.799765E-01 -1.799765E-01   1.318390E-16  3.662673E-16    6    0 0 5 1
 12   -3.385696E-01 -3.385696E-01  -3.261280E-16 -4.816262E-16    6    0 0 3 3
 13   -8.018754E-02 -8.018754E-02  -3.642919E-17 -2.271500E-16    6    0 0 1 5

 14   -1.637905E-01 -1.637905E-01  -1.110223E-16 -3.389156E-16    7    0 0 6 1
 15   -3.942554E-01 -3.942554E-01  -1.110223E-16 -1.408000E-16    7    0 0 4 3
 16   -1.670574E-01 -1.670574E-01   6.938894E-17  2.076800E-16    7    0 0 2 5
 17   -6.682295E-03 -6.682295E-03  -2.927346E-18 -2.190375E-16    7    0 0 0 7

 18   -1.513664E-01 -1.513664E-01   1.249001E-16  4.125755E-16    8    0 0 7 1
 19   -4.428134E-01 -4.428134E-01   1.110223E-16  1.253601E-16    8    0 0 5 3
 20   -2.806564E-01 -2.806564E-01  -2.775558E-16 -4.944761E-16    8    0 0 3 5
 21   -3.118404E-02 -3.118404E-02   2.645453E-17  4.241678E-16    8    0 0 1 7

 22   -1.414357E-01 -1.414357E-01   3.122502E-17  1.103859E-16    9    0 0 8 1
 23   -4.862978E-01 -4.862978E-01  -1.873501E-16 -1.926290E-16    9    0 0 6 3
 24   -4.170866E-01 -4.170866E-01   0.000000E+00  0.000000E+00    9    0 0 4 5
 25   -8.575612E-02 -8.575612E-02   3.469447E-18  2.022857E-17    9    0 0 2 7
 26   -1.949003E-03 -1.949003E-03   7.047314E-19  1.807928E-16    9    0 0 0 9

 27   -1.332549E-01 -1.332549E-01   2.428613E-17  9.112655E-17   10    0 0 9 1
 28   -5.260203E-01 -5.260203E-01   1.249001E-16  1.187217E-16   10    0 0 7 3
 29   -5.732666E-01 -5.732666E-01  -5.689893E-16 -4.962693E-16   10    0 0 5 5
 30   -1.819069E-01 -1.819069E-01  -6.245005E-17 -1.716539E-16   10    0 0 3 7
 31   -1.169402E-02 -1.169402E-02  -5.204170E-18 -2.225143E-16   10    0 0 1 9

 32   -1.263589E-01 -1.263589E-01   2.081668E-17  8.237126E-17   11    0 010 1
 33   -5.628338E-01 -5.628338E-01   2.914335E-16  2.588984E-16   11    0 0 8 3
 34   -7.469011E-01 -7.469011E-01  -4.163336E-17 -2.787073E-17   11    0 0 6 5
 35   -3.298146E-01 -3.298146E-01   2.775558E-17  4.207755E-17   11    0 0 4 7
 36   -3.995455E-02 -3.995455E-02  -3.556183E-17 -4.450285E-16   11    0 0 2 9
 37   -5.847008E-04 -5.847008E-04   2.168404E-19  1.854285E-16   11    0 0 011
```

Appendix B

Numerical Results — 2DSHO

The Hamiltonian used for this run was the "2-D simple harmonic oscillator, in polar coordinates". Initial conditions were as follows:

```
%   HJDApolrsho: Harmonic Oscillator in Polar Coordinates
%
%         no =    6
%      nstep =   30
%      istep =    6
%        eps =  1.000000E-30
%
%        R_i =  1.000000E+00
%    Theta_i =  0.000000E+00
%       Pr_i =  1.000000E-01
%   Ptheta_i =  1.100000E+00
%
%         t1 =  0.000000E+00
%         t2 =  7.500000E-01
```

Table B-1: Reference Trajectory, Matrix, and Symplecticity Error at t_2

```
  x2                r2
================  ==============================================================
 1.104536E+00;     -1.095483E+00,  0.000000E+00,  1.440921E-02,  1.585013E-01
 4.621729E+00;     -9.016393E-02,  1.000000E+00, -1.435004E-01,  1.304549E-02
-5.688529E-01;     -1.133043E+00,  0.000000E+00, -8.979365E-01,  8.163059E-02
 6.911504E+00;      0.000000E+00,  0.000000E+00,  0.000000E+00,  1.000000E+00

                   Symplecticity check:  r2 * ( J r2^T J^-1)  -  I
                  ==============================================================
                    1.000000E+00,  0.000000E+00,  0.000000E+00, -5.690934E-14
                   -3.781697E-16,  1.000000E+00,  5.690934E-14,  0.000000E+00
                    0.000000E+00,  0.000000E+00,  1.000000E+00, -3.799044E-16
                    0.000000E+00,  0.000000E+00,  0.000000E+00,  0.000000E+00
```

Table B-2: DS_{12} for 2D-SHO (continue next page)

```
DS12         NO =    6, NV =    4, INA = 30
********************************************
   I    COEFFICIENT              ORDER    EXPONENTS

    ALL ORDER =    0 COMPONENTS ZERO
    ALL ORDER =    1 COMPONENTS ZERO

   1    1.00000000000000E+00    2      1 0   1 0
   2    1.00000000000000E+00    2      0 1   0 1

   3    6.18925720317969E-01    3      3 0   0 0
   4    9.79675611615971E-01    3      2 0   1 0
   5   -1.78855829339794E-01    3      2 0   0 1
   6   -6.36322471288479E-03    3      1 0   2 0
   7   -1.41167176130881E-01    3      1 0   1 1
   8    1.28859682820735E-02    3      1 0   0 2
   9    3.48832207543786E-03    3      0 0   3 0
  10   -1.53445252596570E-05    3      0 0   2 1
  11   -8.36974104946068E-05    3      0 0   1 2
  12    5.11484175535291E-06    3      0 0   0 3

  13    5.95423188972857E-01    4      4 0   0 0
  14    9.31922275516333E-01    4      3 0   1 0
  15   -2.59434313806014E-01    4      3 0   0 1
  16   -2.48554132891555E-02    4      2 0   2 0
  17   -2.70867838061382E-01    4      2 0   1 1
  18    3.76096764978563E-02    4      2 0   0 2
  19    1.05431804382796E-02    4      1 0   3 0
  20    2.61886741395383E-03    4      1 0   2 1
  21    1.95185979845385E-02    4      1 0   1 2
  22   -1.80398277843570E-03    4      1 0   0 3
  23   -3.43964382932907E-05    4      0 0   4 0
  24   -1.00882927670326E-03    4      0 0   3 1
  25    8.19449197314248E-06    4      0 0   2 2
  26    2.36257586969176E-05    4      0 0   1 3
  27   -1.45522162915632E-06    4      0 0   0 4

  28    5.58851078014409E-01    5      5 0   0 0
  29    8.51380481837993E-01    5      4 0   1 0
  30   -3.26993951494580E-01    5      4 0   0 1
  31   -5.97659072682202E-02    5      3 0   2 0
  32   -3.75769336610911E-01    5      3 0   1 1
  33    7.16417102178937E-02    5      3 0   0 2
  34    2.13337886541097E-02    5      2 0   3 0
  35    1.36580481314620E-02    5      2 0   2 1
  36    5.50370371004263E-02    5      2 0   1 2
  37   -6.95116148760288E-03    5      2 0   0 3
  38   -3.06328027683061E-04    5      1 0   4 0
  39   -4.55646508754974E-03    5      1 0   3 1
  40   -7.03553205538965E-04    5      1 0   2 2
  41   -2.64113025484643E-03    5      1 0   1 3
  42    2.49361502015267E-04    5      1 0   0 4
  43    3.32979821683729E-05    5      0 0   5 0
  44    1.95697068579221E-05    5      0 0   4 1
  45    2.17782513333632E-04    5      0 0   3 2
  46   -2.78626148221566E-06    5      0 0   2 3
  47   -4.96315697585036E-06    5      0 0   1 4
  48    3.09056425899763E-07    5      0 0   0 5
```

Table B-2: (cont.) DS_{12} for 2D-SHO

```
*********************************************
  I    COEFFICIENT           ORDER    EXPONENTS

 49    5.08429549983533E-01    6      6 0  0 0
 50    7.35291098206934E-01    6      5 0  1 0
 51   -3.75397864062992E-01    6      5 0  0 1
 52   -1.12882453662681E-01    6      4 0  2 0
 53   -4.40479004605637E-01    6      4 0  1 1
 54    1.10712142710012E-01    6      4 0  0 2
 55    3.63624262750108E-02    6      3 0  3 0
 56    4.06775421740246E-02    6      3 0  2 1
 57    9.85741636038610E-02    6      3 0  1 2
 58   -1.62843985501847E-02    6      3 0  0 3
 59   -1.31398380982056E-03    6      2 0  4 0
 60   -1.22828726759353E-02    6      2 0  3 1
 61   -4.62210776211041E-03    6      2 0  2 2
 62   -9.72606131936708E-03    6      2 0  1 3
 63    1.19057656214169E-03    6      2 0  0 4
 64    1.72057139692110E-04    6      1 0  5 0
 65    2.17493784793327E-04    6      1 0  4 1
 66    1.30659594521423E-03    6      1 0  3 2
 67    1.53831369667279E-04    6      1 0  2 3
 68    3.49235226301906E-04    6      1 0  1 4
 69   -3.40175789016407E-05    6      1 0  0 5
 70   -1.08642252052135E-06    6      0 0  6 0
 71   -1.93775027887211E-05    6      0 0  5 1
 72   -6.90653635225183E-06    6      0 0  4 2
 73   -4.15803410169200E-05    6      0 0  3 3
 74    7.63644095695723E-07    6      0 0  2 4
 75    9.19487119295613E-07    6      0 0  1 5
 76   -5.80682626677918E-08    6      0 0  0 6
```

Table B-3: Numerical *vs* Theoretical x_2 (continue next page)

```
A = TXC2        NO =   6, NV =   4, ina = 63
B = DXC2        NO =   6, NV =   4, inb = 59
*****************************************************************************************
  I  A Coefficient B Coefficient  Difference      Frac. Diff.   Order Exponents

  1  -1.000000E-01 -1.000000E-01   1.164589E-13  5.822946E-13     0    0 0 0 0

  2   3.834759E-17  2.044528E-12  -2.044489E-12 -9.999625E-01     1    1 0 0 0
  3   1.100000E+00  1.100000E+00   8.043566E-14  3.656166E-14     1    0 1 0 0
  4  -1.591549E-01 -1.591549E-01   2.387673E-14  7.501097E-14     1    0 0 1 0
  5                -1.626546E-13                                  1    0 0 0 1

  6  -1.967414E-01 -1.967414E-01  -1.277766E-12 -3.247324E-12     2    2 0 0 0
  7  -1.100000E+00 -1.100000E+00   5.093703E-13  2.315320E-13     2    1 1 0 0
  8   5.000000E-02  5.000000E-02  -5.822946E-14 -5.822946E-13     2    0 2 0 0
  9  -1.552839E-01 -1.552839E-01   3.381323E-14  1.088755E-13     2    1 0 1 0
 10                -1.530893E-13                                  2    0 1 1 0
 11   2.834913E-02  2.834913E-02   1.040526E-13  1.835199E-12     2    1 0 0 1
 12   1.591549E-01  1.591549E-01  -3.728268E-14 -1.171270E-13     2    0 1 0 1
 13  -1.687898E-05 -1.687898E-05  -1.900974E-14 -5.631188E-10     2    0 0 2 0
 14  -1.841343E-04 -1.841343E-04   5.319801E-15  1.444544E-11     2    0 0 1 1
 15   1.687898E-05  1.687898E-05   1.494377E-15  4.426742E-11     2    0 0 0 2

 16  -8.863633E-02 -8.863633E-02   2.169168E-13  1.223633E-12     3    3 0 0 0
 17   4.471244E-03  4.471244E-03  -4.502156E-13 -5.034567E-11     3    2 1 0 0
 18  -1.917379E-17 -1.022265E-12   1.022246E-12  9.999625E-01     3    1 2 0 0
 19  -1.833333E-01 -1.833333E-01  -1.340941E-14 -3.657112E-14     3    0 3 0 0
 20  -1.426707E-01 -1.426707E-01  -1.863856E-13 -6.532021E-13     3    2 0 1 0
 21  -1.176232E-04 -1.176232E-04  -4.512696E-14 -1.918284E-10     3    1 1 1 0
 22   7.957747E-02  7.957747E-02  -1.193837E-14 -7.501097E-14     3    0 2 1 0
 23   2.592637E-02  2.592637E-02   1.083682E-13  2.089922E-12     3    2 0 0 1
 24  -1.293856E-03 -1.293856E-03   6.141959E-14  2.373510E-11     3    1 1 0 1
 25                 8.132731E-14                                  3    0 2 0 1
 26   2.897633E-03  2.897633E-03  -1.118794E-13 -1.930530E-11     3    1 0 2 0
 27  -1.151300E-02 -1.151300E-02   4.049712E-15  1.758757E-13     3    0 1 2 0
 28   2.065760E-02  2.065760E-02   2.825734E-14  6.839456E-13     3    1 0 1 1
 29   1.701847E-05  1.701847E-05   5.808556E-15  1.706544E-10     3    0 1 1 1
 30  -1.868291E-03 -1.868291E-03  -8.058278E-15 -2.156591E-12     3    1 0 0 2
 31   9.360160E-05  9.360160E-05  -4.654997E-15 -2.486601E-11     3    0 1 0 2
 32  -1.109712E-03 -1.109712E-03   9.771101E-16  4.402538E-13     3    0 0 3 0
 33   1.558573E-05  1.558573E-05   7.255051E-15  2.327467E-10     3    0 0 2 1
 34   5.132329E-05  5.132329E-05  -1.339011E-15 -1.304487E-11     3    0 0 1 2
 35  -3.960818E-06 -3.960818E-06   1.525110E-16  1.925247E-11     3    0 0 0 3

 36  -7.671478E-02 -7.671478E-02  -7.050211E-12 -4.595080E-11     4    4 0 0 0
 37   4.228411E-03  4.228411E-03   6.237058E-13  7.375180E-11     4    3 1 0 0
 38   9.837071E-02  9.837071E-02   6.388813E-13  3.247315E-12     4    2 2 0 0
 39   1.833333E-01  1.833333E-01  -8.489390E-14 -2.315288E-13     4    1 3 0 0
 40  -4.166667E-03 -4.166667E-03   4.852455E-15  5.822946E-13     4    0 4 0 0
 41  -1.185267E-01 -1.185267E-01   1.593232E-12  6.720988E-12     4    3 0 1 0
 42  -2.886885E-02 -2.886885E-02   4.399176E-13  7.619243E-12     4    2 1 1 0
 43   7.764195E-02  7.764195E-02  -1.690661E-14 -1.088755E-13     4    1 2 1 0
 44                 2.551491E-14                                  4    0 3 1 0
 45   3.438781E-02  3.438781E-02   2.061113E-12  2.996865E-11     4    3 0 0 1
 46  -1.856503E-03 -1.856503E-03  -1.335639E-13 -3.597191E-11     4    2 1 0 1
 47  -1.417457E-02 -1.417457E-02  -5.202653E-14 -1.835207E-12     4    1 2 0 1
 48  -2.652582E-02 -2.652582E-02   6.213346E-15  1.171188E-13     4    0 3 0 1
 49   1.322212E-02  1.322212E-02   1.843623E-13  6.971738E-12     4    2 0 2 0
 50  -3.397514E-02 -3.397514E-02  -1.490127E-14 -2.192968E-13     4    1 1 2 0
 51   8.439489E-06  8.439489E-06   9.504843E-15  5.631172E-10     4    0 2 2 0
 52   3.556726E-02  3.556726E-02  -3.584945E-13 -5.039670E-12     4    2 0 1 1
 53   4.201904E-03  4.201904E-03  -8.627571E-14 -1.026626E-11     4    1 1 1 1
 54   9.206715E-05  9.206715E-05  -2.659897E-15 -1.444542E-11     4    0 2 1 1
 55  -5.097111E-03 -5.097111E-03  -2.290608E-13 -2.246967E-11     4    2 0 0 2
 56   2.716670E-04  2.716670E-04   1.678412E-14  3.089099E-11     4    1 1 0 2
 57  -8.439489E-06 -8.439489E-06  -7.471887E-16 -4.426742E-11     4    0 2 0 2
 58  -2.277357E-03 -2.277357E-03  -7.369647E-15 -1.618026E-12     4    1 0 3 0
 59   4.801924E-05  4.801924E-05  -5.282877E-16 -5.500791E-12     4    0 1 3 0
 60  -1.450438E-03 -1.450438E-03  -1.302766E-14 -4.490942E-12     4    1 0 2 1
 61   1.638583E-03  1.638583E-03   2.058114E-15  6.280165E-13     4    0 1 2 1
 62  -2.592644E-03 -2.592644E-03   2.618782E-14  5.050407E-12     4    1 0 1 2
 63  -3.613684E-06 -3.613684E-06   3.953497E-15  5.470175E-10     4    0 1 1 2
 64   2.446081E-04  2.446081E-04   9.204090E-15  1.881395E-11     4    1 0 0 3
 65  -1.324957E-05 -1.324957E-05  -5.490427E-16 -2.071927E-11     4    0 1 0 3
 66   2.170333E-05  2.170333E-05  -9.245487E-16 -2.129970E-11     4    0 0 4 0
 67   3.204881E-04  3.204881E-04   4.918348E-16  7.673215E-13     4    0 0 3 1
 68  -6.768599E-06 -6.768599E-06   3.248101E-16  2.399390E-11     4    0 0 2 2
 69  -1.057257E-05 -1.057257E-05  -5.685120E-16 -2.688617E-11     4    0 0 1 3
 70   7.748120E-07  7.748120E-07  -1.044007E-16 -6.737162E-11     4    0 0 0 4
```

Table B-3: (cont.) Numerical *vs* Theoretical x_2

```
***********************************************************************************
   I  A Coefficient B Coefficient  Difference      Frac. Diff.    Order Exponents

  71  -5.980071E-02 -5.980071E-02  1.283592E-10  1.073225E-09     5    5 0 0 0
  72  -1.377188E-02 -1.377188E-02 -6.717741E-12 -2.438934E-10     5    4 1 0 0
  73   4.431817E-02  4.431817E-02 -1.084584E-13 -1.223633E-12     5    3 2 0 0
  74  -7.452074E-04 -7.452074E-04  7.503598E-14  5.034570E-11     5    2 3 0 0
  75   1.597816E-18  8.518858E-14 -8.518699E-14 -9.999625E-01     5    1 4 0 0
  76   9.166667E-03  9.166667E-03  6.702538E-16  3.655930E-14     5    0 5 0 0
  77  -7.884741E-02 -7.884741E-02 -1.093968E-11 -6.937247E-11     5    4 0 1 0
  78  -6.995130E-02 -6.995130E-02 -3.738674E-12 -2.672341E-11     5    3 1 1 0
  79   7.133536E-02  7.133536E-02  9.319802E-14  6.532385E-13     5    2 2 1 0
  80   1.960387E-05  1.960387E-05  7.521235E-15  1.918303E-10     5    1 3 1 0
  81  -6.631456E-03 -6.631456E-03  9.949723E-16  7.501915E-14     5    0 4 1 0
  82   3.703922E-02  3.703922E-02 -4.641219E-11 -6.265277E-10     5    4 0 0 1
  83   2.797222E-03  2.797222E-03  1.977649E-12  3.535023E-10     5    3 1 0 1
  84  -1.296318E-02 -1.296318E-02 -5.418496E-14 -2.089955E-12     5    2 2 0 1
  85   2.156426E-04  2.156426E-04 -1.023660E-14 -2.373511E-11     5    1 3 0 1
  86                -6.777280E-15                                 5    0 4 0 1
  87   3.510804E-02  3.510804E-02 -1.750388E-12 -2.492859E-11     5    3 0 2 0
  88  -6.532541E-02 -6.532541E-02  1.812751E-13  1.387478E-12     5    2 1 2 0
  89  -1.448817E-03 -1.448817E-03  5.593966E-14  1.930529E-11     5    1 2 2 0
  90   1.918833E-03  1.918833E-03 -6.748887E-16 -1.758592E-13     5    0 3 2 0
  91   3.844111E-02  3.844111E-02  3.110446E-12  4.045728E-11     5    3 0 1 1
  92   1.627161E-02  1.627161E-02  6.706267E-13  2.060727E-11     5    2 1 1 1
  93  -1.032880E-02 -1.032880E-02 -1.412932E-14 -6.839771E-13     5    1 2 1 1
  94  -2.836412E-06 -2.836412E-06 -9.680814E-16 -1.706525E-10     5    0 3 1 1
  95  -8.527845E-03 -8.527845E-03  6.479123E-12  3.798804E-10     5    3 0 0 2
  96   1.443852E-04  1.443852E-04 -2.134734E-13 -7.392500E-10     5    2 1 0 2
  97   9.341455E-04  9.341455E-04  4.029071E-15  2.156554E-12     5    1 2 0 2
  98  -1.560027E-05 -1.560027E-05  7.758326E-16  2.486600E-11     5    0 3 0 2
  99  -2.165097E-03 -2.165097E-03 -9.306249E-16 -2.149153E-13     5    2 0 3 0
 100   7.109310E-04  7.109310E-04  1.958623E-14  1.377506E-11     5    1 1 3 0
 101   5.548561E-04  5.548561E-04 -4.885415E-16 -4.402416E-13     5    0 2 3 0
 102  -7.968109E-03 -7.968109E-03  3.292835E-13  2.066259E-11     5    2 0 2 1
 103   9.472637E-03  9.472637E-03 -2.397063E-14 -1.265256E-12     5    1 1 2 1
 104  -7.792863E-06 -7.792863E-06 -3.627526E-15 -2.327467E-10     5    0 2 2 1
 105  -6.091094E-03 -6.091094E-03 -3.211878E-13 -2.636537E-11     5    2 0 1 2
 106  -8.941819E-04 -8.941819E-04 -3.864983E-14 -2.161184E-11     5    1 1 1 2
 107  -2.566165E-05 -2.566165E-05  6.695059E-16  1.304487E-11     5    0 2 1 2
 108   8.594988E-04  8.594988E-04 -4.338960E-13 -2.524122E-10     5    2 0 0 3
 109  -5.059980E-05 -5.059980E-05  9.201593E-15  9.092520E-11     5    1 1 0 3
 110   1.980409E-06  1.980409E-06 -7.625891E-17 -1.925332E-11     5    0 2 0 3
 111   1.668446E-04  1.668446E-04  1.013159E-14  3.036234E-11     5    1 0 4 0
 112  -2.212752E-04 -2.212752E-04 -2.783011E-16 -6.288576E-13     5    0 1 4 0
 113   9.969017E-04  9.969017E-04 -3.847562E-16 -1.929760E-13     5    1 0 3 1
 114  -1.875374E-05 -1.875374E-05 -1.525154E-15 -4.066265E-11     5    0 1 3 1
 115   4.002195E-04  4.002195E-04 -2.301640E-14 -2.875472E-11     5    1 0 2 2
 116  -2.284983E-04 -2.284983E-04  8.500789E-16  1.860143E-12     5    0 1 2 2
 117   3.022149E-04  3.022149E-04  1.442729E-14  2.386925E-11     5    1 0 1 3
 118   6.181782E-07  6.181782E-07  7.505460E-16  6.070628E-10     5    0 1 1 3
 119  -3.086106E-05 -3.086106E-05  1.413413E-14  2.289961E-10     5    1 0 0 4
 120   1.861421E-06  1.861421E-06 -1.509018E-16 -4.053404E-11     5    0 1 0 4
 121  -9.702908E-06 -9.702908E-06 -8.811514E-17 -4.540656E-12     5    0 0 5 0
 122  -1.233607E-05 -1.233607E-05 -5.637616E-16 -2.285013E-11     5    0 0 4 1
 123  -6.907672E-05 -6.907672E-05  3.447932E-17  2.495727E-13     5    0 0 3 2
 124   2.126961E-06  2.126961E-06  5.445079E-16  1.280014E-10     5    0 0 2 3
 125   1.906008E-06  1.906008E-06 -2.486666E-16 -6.523232E-11     5    0 0 1 4
 126  -1.380505E-07 -1.380505E-07 -1.783268E-16 -6.458753E-10     5    0 0 0 5
```

Table B-4: Numerical *vs* Theoretical y_2 (continue next page)

```
A = TYC2       NO =   6, NV =   4, ina = 65
B = DYC2       NO =   6, NV =   4, inb = 61
*******************************************************************************
  I  A Coefficient B Coefficient  Difference     Frac. Diff.    Order Exponents

  1  -1.100000E+00 -1.100000E+00  -8.043566E-14 -3.656166E-14    0    0 0 0 0

  2   1.100000E+00  1.100000E+00  -5.093703E-13 -2.315320E-13    1    1 0 0 0
  3  -1.000000E-01 -1.000000E-01   1.164589E-13  5.822946E-13    1    0 1 0 0
  4                 1.530893E-13                                 1    0 0 1 0
  5  -1.591549E-01 -1.591549E-01   3.728268E-14  1.171270E-13    1    0 0 0 1

  6  -4.471244E-03 -4.471244E-03   4.502156E-13  5.034567E-11    2    2 0 0 0
  7   3.834759E-17  2.044528E-12  -2.044489E-12 -9.999625E-01    2    1 1 0 0
  8   5.500000E-01  5.500000E-01   4.021783E-14  3.656166E-14    2    0 2 0 0
  9   1.176232E-04  1.176232E-04   4.512696E-14  1.918284E-10    2    1 0 1 0
 10  -1.591549E-01 -1.591549E-01   2.387673E-14  7.501097E-14    2    0 1 1 0
 11   1.293856E-03  1.293856E-03  -6.141959E-14 -2.373510E-11    2    1 0 0 1
 12                -1.626546E-13                                 2    0 1 0 1
 13   1.151300E-02  1.151300E-02  -4.049712E-15 -1.758757E-13    2    0 0 2 0
 14  -1.701847E-05 -1.701847E-05  -5.808556E-15 -1.706544E-10    2    0 0 1 1
 15  -9.360160E-05 -9.360160E-05   4.654997E-15  2.486601E-11    2    0 0 0 2

 16  -4.228411E-03 -4.228411E-03  -6.237058E-13 -7.375180E-11    3    3 0 0 0
 17  -1.967414E-01 -1.967414E-01  -1.277766E-12 -3.247324E-12    3    2 1 0 0
 18  -5.500000E-01 -5.500000E-01   2.546852E-13  2.315320E-13    3    1 2 0 0
 19   1.666667E-02  1.666667E-02  -1.940982E-14 -5.822946E-13    3    0 3 0 0
 20   2.886885E-02  2.886885E-02  -4.399194E-13 -7.619274E-12    3    2 0 1 0
 21  -1.552839E-01 -1.552839E-01   3.381323E-14  1.088755E-13    3    1 1 1 0
 22                -7.654467E-14                                 3    0 2 1 0
 23   1.856503E-03  1.856503E-03   1.335639E-13  3.597191E-11    3    2 0 0 1
 24   2.834913E-02  2.834913E-02   1.040526E-13  1.835199E-12    3    1 1 0 1
 25   7.957747E-02  7.957747E-02  -1.864134E-14 -1.171270E-13    3    0 2 0 1
 26   3.397514E-02  3.397514E-02   1.490127E-14  2.192968E-13    3    1 0 2 0
 27  -1.687898E-05 -1.687898E-05  -1.900974E-14 -5.631188E-10    3    0 1 2 0
 28  -4.201904E-03 -4.201904E-03   8.627615E-14  1.026632E-11    3    1 0 1 1
 29  -1.841343E-04 -1.841343E-04   5.319801E-15  1.444544E-11    3    0 1 1 1
 30  -2.716670E-04 -2.716670E-04  -1.678412E-14 -3.089099E-11    3    1 0 0 2
 31   1.687898E-05  1.687898E-05   1.494377E-15  4.426742E-11    3    0 1 0 2
 32  -4.801924E-05 -4.801924E-05   5.282877E-16  5.500791E-12    3    0 0 3 0
 33  -1.638583E-03 -1.638583E-03  -2.058114E-15 -6.280165E-13    3    0 0 2 1
 34   3.613684E-06  3.613684E-06  -3.953497E-15 -5.470175E-10    3    0 0 1 2
 35   1.324957E-05  1.324957E-05   5.490427E-16  2.071927E-11    3    0 0 0 3

 36   1.377188E-02  1.377188E-02   6.717741E-12  2.438934E-10    4    4 0 0 0
 37  -8.863633E-02 -8.863633E-02   2.169168E-13  1.223633E-12    4    3 1 0 0
 38   2.235622E-03  2.235622E-03  -2.251077E-13 -5.034566E-11    4    2 2 0 0
 39  -6.391264E-18 -3.407543E-13   3.407479E-13  9.999625E-01    4    1 3 0 0
 40  -4.583333E-02 -4.583333E-02  -3.352353E-15 -3.657112E-14    4    0 4 0 0
 41   6.995130E-02  6.995130E-02   3.738676E-12  2.672342E-11    4    3 0 1 0
 42  -1.426707E-01 -1.426707E-01  -1.863856E-13 -6.532021E-13    4    2 1 1 0
 43  -5.881162E-05 -5.881162E-05  -2.256348E-14 -1.918284E-10    4    1 2 1 0
 44   2.652582E-02  2.652582E-02  -3.979456E-15 -7.501097E-14    4    0 3 1 0
 45  -2.797222E-03 -2.797222E-03  -1.977649E-12 -3.535023E-10    4    3 0 0 1
 46   2.592637E-02  2.592637E-02   1.083682E-13  2.089922E-12    4    2 1 0 1
 47  -6.469278E-04 -6.469278E-04   3.070984E-14  2.373513E-11    4    1 2 0 1
 48                 2.710917E-14                                 4    0 3 0 1
 49   6.532541E-02  6.532541E-02  -1.812734E-13 -1.387465E-12    4    2 0 2 0
 50   2.897633E-03  2.897633E-03  -1.118794E-13 -1.930530E-11    4    1 1 2 0
 51  -5.756499E-03 -5.756499E-03   2.024856E-15  1.758757E-13    4    0 2 2 0
 52  -1.627161E-02 -1.627161E-02  -6.706272E-13 -2.060728E-11    4    2 0 1 1
 53   2.065760E-02  2.065760E-02   2.825734E-14  6.839456E-13    4    1 1 1 1
 54   8.509237E-06  8.509237E-06   2.904278E-15  1.706544E-10    4    0 2 1 1
 55  -1.443852E-04 -1.443852E-04   2.134734E-13  7.392500E-10    4    2 0 0 2
 56  -1.868291E-03 -1.868291E-03  -8.058278E-15 -2.156591E-12    4    1 1 0 2
 57   4.680080E-05  4.680080E-05  -2.327498E-15 -2.486601E-11    4    0 2 0 2
 58  -7.109310E-04 -7.109310E-04  -1.958626E-14 -1.377508E-11    4    1 0 3 0
 59  -1.109712E-03 -1.109712E-03   9.771101E-16  4.402538E-13    4    0 1 3 0
 60  -9.472637E-03 -9.472637E-03   2.397063E-14  1.265256E-12    4    1 0 2 1
 61   1.558573E-05  1.558573E-05   7.255051E-15  2.327467E-10    4    0 1 2 1
 62   8.941819E-04  8.941819E-04   3.864988E-14  2.161187E-11    4    1 0 1 2
 63   5.132329E-05  5.132329E-05  -1.339011E-15 -1.304487E-11    4    0 1 1 2
 64   5.059980E-05  5.059980E-05  -9.201600E-15 -9.092526E-11    4    1 0 0 3
 65  -3.960818E-06 -3.960818E-06   1.525110E-16  1.925247E-11    4    0 1 0 3
 66   2.212752E-04  2.212752E-04   2.783045E-16  6.288653E-13    4    0 0 4 0
 67   1.875374E-05  1.875374E-05   1.525157E-15  4.066272E-11    4    0 0 3 1
 68   2.284983E-04  2.284983E-04  -8.500789E-16 -1.860143E-12    4    0 0 2 2
 69  -6.181782E-07 -6.181782E-07  -7.505460E-16 -6.070628E-10    4    0 0 1 3
 70  -1.861421E-06 -1.861421E-06   1.509016E-16  4.053399E-11    4    0 0 0 4
```

Table B-4: (cont.) Numerical *vs* Theoretical y_2

```
*****************************************************************************************
  I   A Coefficient B Coefficient  Difference     Frac. Diff.    Order Exponents

 71    3.020390E-02  3.020390E-02  -2.293078E-11 -3.795997E-10    5    5 0 0 0
 72   -7.671478E-02 -7.671478E-02  -7.050211E-12 -4.595080E-11    5    4 1 0 0
 73    2.114206E-03  2.114206E-03   3.118527E-13  7.375175E-11    5    3 2 0 0
 74    3.279024E-02  3.279024E-02   2.129625E-13  3.247346E-12    5    2 3 0 0
 75    4.583333E-02  4.583333E-02  -2.122347E-14 -2.315288E-13    5    1 4 0 0
 76   -8.333333E-04 -8.333333E-04   9.704965E-16  5.822979E-13    5    0 5 0 0
 77    1.194191E-01  1.194191E-01  -1.092030E-11 -4.572259E-11    5    4 0 1 0
 78   -1.185267E-01 -1.185267E-01   1.593232E-12  6.720988E-12    5    3 1 1 0
 79   -1.443443E-02 -1.443443E-02   2.199590E-13  7.619251E-12    5    2 2 1 0
 80    2.588065E-02  2.588065E-02  -5.635249E-15 -1.088699E-13    5    1 3 1 0
 81                  6.378727E-15                                 5    0 4 1 0
 82   -1.207100E-02 -1.207100E-02   7.581365E-12  3.140323E-10    5    4 0 0 1
 83    3.438781E-02  3.438781E-02   2.061113E-12  2.996865E-11    5    3 1 0 1
 84   -9.282517E-04 -9.282517E-04  -6.678183E-14 -3.597183E-11    5    2 2 0 1
 85   -4.724855E-03 -4.724855E-03  -1.734225E-14 -1.835215E-12    5    1 3 0 1
 86   -6.631456E-03 -6.631456E-03   1.553336E-15  1.171188E-13    5    0 4 0 1
 87    1.010308E-01  1.010308E-01  -2.610273E-13 -1.291820E-12    5    3 0 2 0
 88    1.322212E-02  1.322212E-02   1.843623E-13  6.971738E-12    5    2 1 2 0
 89   -1.698757E-02 -1.698757E-02  -7.450204E-15 -2.192840E-13    5    1 2 2 0
 90    2.813163E-06  2.813163E-06   3.168276E-15  5.631164E-10    5    0 3 2 0
 91   -4.260646E-02 -4.260646E-02   2.428437E-12  2.849846E-11    5    3 0 1 1
 92    3.556726E-02  3.556726E-02  -3.584945E-13 -5.039670E-12    5    2 1 1 1
 93    2.100952E-03  2.100952E-03  -4.313802E-14 -1.026630E-11    5    1 2 1 1
 94    3.068905E-05  3.068905E-05  -8.867054E-16 -1.444661E-11    5    0 3 1 1
 95    1.347964E-03  1.347964E-03  -9.847443E-13 -3.652709E-10    5    3 0 0 2
 96   -5.097111E-03 -5.097111E-03  -2.290608E-13 -2.246967E-11    5    2 1 0 2
 97    1.358335E-04  1.358335E-04   8.392084E-15  3.089107E-11    5    1 2 0 2
 98   -2.813163E-06 -2.813163E-06  -2.490720E-16 -4.426904E-11    5    0 3 0 2
 99   -3.243739E-03 -3.243739E-03   3.145125E-13  4.847994E-11    5    2 0 3 0
100   -2.277357E-03 -2.277357E-03  -7.369647E-15 -1.618026E-12    5    1 1 3 0
101    2.400962E-05  2.400962E-05  -2.641443E-16 -5.500800E-12    5    0 2 3 0
102   -2.662067E-02 -2.662067E-02   5.286266E-14  9.928875E-13    5    2 0 2 1
103   -1.450438E-03 -1.450438E-03  -1.302756E-14 -4.490905E-12    5    1 1 2 1
104    8.192913E-04  8.192913E-04   1.028976E-15  6.279669E-13    5    0 2 2 1
105    4.814730E-03  4.814730E-03  -2.378050E-13 -2.469557E-11    5    2 0 1 2
106   -2.592644E-03 -2.592644E-03   2.618782E-14  5.050407E-12    5    1 1 1 2
107   -1.806842E-06 -1.806842E-06   1.976748E-15  5.470176E-10    5    0 2 1 2
108    1.448434E-05  1.448434E-05   6.435217E-14  2.221439E-09    5    2 0 0 3
109    2.446081E-04  2.446081E-04   9.204090E-15  1.881395E-11    5    1 1 0 3
110   -6.624783E-06 -6.624783E-06  -2.745218E-16 -2.071930E-11    5    0 2 0 3
111    1.069426E-03  1.069426E-03  -4.740051E-15 -2.216166E-12    5    1 0 4 0
112    2.170333E-05  2.170333E-05  -9.245487E-16 -2.129970E-11    5    0 1 4 0
113    3.585799E-04  3.585799E-04  -3.826215E-14 -5.335233E-11    5    1 0 3 1
114    3.204881E-04  3.204881E-04   4.918348E-16  7.673215E-13    5    0 1 3 1
115    1.944869E-03  1.944869E-03  -3.141232E-15 -8.075689E-13    5    1 0 2 2
116   -6.768599E-06 -6.768599E-06   3.248101E-16  2.399390E-11    5    0 1 2 2
117   -1.670461E-04 -1.670461E-04   1.229353E-14  3.679683E-11    5    1 0 1 3
118   -1.057257E-05 -1.057257E-05  -5.685120E-16 -2.688617E-11    5    0 1 1 3
119   -8.782695E-06 -8.782695E-06  -2.282558E-15 -1.299463E-10    5    1 0 0 4
120    7.748120E-07  7.748120E-07  -1.044007E-16 -6.737162E-11    5    0 1 0 4
121   -5.162599E-06 -5.162599E-06  -8.552714E-17 -8.283341E-12    5    0 0 5 0
122   -9.559471E-05 -9.559471E-05   2.789805E-16  1.459184E-12    5    0 0 4 1
123   -4.651451E-06 -4.651451E-06   1.101279E-15  1.183801E-10    5    0 0 3 2
124   -3.120939E-05 -3.120939E-05   7.124140E-17  1.141346E-12    5    0 0 2 3
125    9.309422E-08  9.309422E-08  -2.604519E-16 -1.398862E-09    5    0 0 1 4
126    2.595217E-07  2.595217E-07   3.324602E-17  6.405248E-11    5    0 0 0 5
```

Table B-5: Numerical *vs* Theoretical p_{x2} (continue next page)

```
A = TPX2       NO =    6, NV =    4, ina = 64
B = DPX2       NO =    6, NV =    4, inb = 60
*****************************************************************************
  I  A Coefficient B Coefficient  Difference    Frac. Diff.   Order Exponents

  1   6.283185E+00  6.283185E+00  -5.256906E-13 -4.183313E-14   0    0 0 0 0

  2   6.283185E+00  6.283185E+00  -2.726153E-12 -2.169403E-13   1    1 0 0 0
  3  -2.650395E-16 -7.285839E-13   7.283188E-13  9.992727E-01   1    0 1 0 0
  4   3.834759E-17  9.943088E-13  -9.942705E-13 -9.999229E-01   1    0 0 1 0
  5                -2.040252E-13                                1    0 0 0 1

  6   6.155483E+00  6.155483E+00  -2.785217E-12 -2.262386E-13   2    2 0 0 0
  7   2.650395E-16  1.285502E-11  -1.285476E-11 -9.999588E-01   2    1 1 0 0
  8  -3.141593E+00 -3.141593E+00   2.628453E-13  4.183313E-14   2    0 2 0 0
  9  -7.996264E-02 -7.996264E-02   2.183531E-13  1.365345E-12   2    1 0 1 0
 10                -1.589007E-13                                2    0 1 1 0
 11  -8.869795E-01 -8.869795E-01   3.880646E-13  2.187562E-13   2    1 0 0 1
 12  -3.834759E-17 -9.348078E-13   9.347694E-13  9.999180E-01   2    0 1 0 1
 13   6.575332E-02  6.575332E-02  -4.484607E-14 -3.410175E-13   2    0 0 2 0
 14  -1.928250E-04 -1.928250E-04  -1.955529E-14 -5.070736E-11   2    0 0 1 1
 15  -5.258863E-04 -5.258863E-04   1.894142E-15  1.800904E-12   2    0 0 0 2

 16   5.855440E+00  5.855440E+00  -6.433853E-12 -5.493911E-13   3    3 0 0 0
 17   1.123784E+00  1.123784E+00  -8.117035E-12 -3.611474E-12   3    2 1 0 0
 18  -3.141593E+00 -3.141593E+00   1.363076E-12  2.169403E-13   3    1 2 0 0
 19   4.417325E-17  1.214306E-13  -1.213865E-13 -9.992727E-01   3    0 3 0 0
 20  -3.123423E-01 -3.123423E-01  -1.173721E-12 -1.878901E-12   3    2 0 1 0
 21   8.869795E-01  8.869795E-01   3.022860E-13  1.704019E-13   3    1 1 1 0
 22  -1.917379E-17 -4.971544E-13   4.971352E-13  9.999229E-01   3    0 2 1 0
 23  -1.701913E+00 -1.701913E+00   1.727812E-12  5.076089E-13   3    2 0 0 1
 24  -1.619299E-01 -1.619299E-01   1.521901E-12  4.699259E-12   3    1 1 0 1
 25                 1.020126E-13                                3    0 2 0 1
 26   1.987343E-01  1.987343E-01  -4.794082E-14 -1.206154E-13   3    1 0 2 0
 27   9.641250E-05  9.641250E-05  -1.274423E-13 -6.609219E-10   3    0 1 2 0
 28   3.290966E-02  3.290966E-02   2.816775E-13  4.279556E-12   3    1 0 1 1
 29   1.051773E-03  1.051773E-03  -5.503581E-14 -2.616336E-11   3    0 1 1 1
 30   1.226390E-01  1.226390E-01  -1.464696E-13 -5.971578E-13   3    1 0 0 2
 31  -9.641250E-05 -9.641250E-05  -5.215723E-14 -2.704900E-10   3    0 1 0 2
 32  -8.644768E-04 -8.644768E-04   1.636766E-15  9.466800E-13   3    0 0 3 0
 33  -1.901598E-02 -1.901598E-02   4.863731E-15  1.278853E-13   3    0 0 2 1
 34   1.029750E-04  1.029750E-04  -1.196647E-14 -5.810377E-11   3    0 0 1 2
 35   1.484450E-04  1.484450E-04   2.797909E-15  9.424058E-12   3    0 0 0 3

 36   5.248884E+00  5.248884E+00   2.749534E-11  2.619161E-12   4    4 0 0 0
 37   2.753858E+00  2.753858E+00  -1.811384E-12 -3.288812E-13   4    3 1 0 0
 38  -3.077742E+00 -3.077742E+00   1.392608E-12  2.262386E-13   4    2 2 0 0
 39  -4.417325E-17 -2.142505E-12   2.142461E-12  9.999588E-01   4    1 3 0 0
 40   2.617994E-01  2.617994E-01  -2.190609E-14 -4.183755E-14   4    0 4 0 0
 41  -9.096820E-01 -9.096820E-01   1.427569E-11  7.846529E-12   4    3 0 1 0
 42   2.588892E+00  2.588892E+00  -1.690592E-12 -3.265088E-13   4    2 1 1 0
 43   3.998132E-02  3.998132E-02  -1.091766E-13 -1.365345E-12   4    1 2 1 0
 44                 2.648229E-14                                4    0 3 1 0
 45  -2.332066E+00 -2.332066E+00  -7.447043E-12 -1.596662E-12   4    3 0 0 1
 46  -6.345470E-01 -6.345470E-01   1.462747E-12  1.152591E-12   4    2 1 0 1
 47   4.434898E-01  4.434898E-01  -1.940323E-13 -2.187562E-13   4    1 2 0 1
 48   6.391264E-18  1.558014E-13  -1.557950E-13 -9.999180E-01   4    0 3 0 1
 49   3.395090E-01  3.395090E-01  -1.017200E-12 -1.498046E-12   4    2 0 2 0
 50  -1.635842E-02 -1.635842E-02  -6.639403E-13 -2.029354E-11   4    1 1 2 0
 51  -3.287666E-02 -3.287666E-02   2.242304E-14  3.410175E-13   4    0 2 2 0
 52   1.943032E-01  1.943032E-01  -2.550071E-12 -6.562094E-12   4    2 0 1 1
 53  -2.442262E-01 -2.442262E-01   1.509001E-13  3.089352E-13   4    1 1 1 1
 54   9.641250E-05  9.641250E-05   9.777647E-15  5.070736E-11   4    0 2 1 1
 55   3.437385E-01  3.437385E-01   6.953327E-13  1.011427E-12   4    2 0 0 2
 56   3.390786E-02  3.390786E-02  -1.404554E-13 -2.071133E-12   4    1 1 0 2
 57   2.629432E-04  2.629432E-04  -9.470709E-16 -1.800904E-12   4    0 2 0 2
 58  -7.712473E-03 -7.712473E-03  -5.077026E-14 -3.291438E-12   4    1 0 3 0
 59   6.338661E-03  6.338661E-03   3.131393E-15  2.470074E-13   4    0 1 3 0
 60  -8.603333E-02 -8.603333E-02   1.604515E-13  9.324962E-13   4    1 0 2 1
 61  -1.029750E-04 -1.029750E-04   4.847874E-14  2.353908E-10   4    0 1 2 1
 62  -8.800394E-03 -8.800394E-03   1.355496E-13  7.701335E-12   4    1 0 1 2
 63  -4.453351E-04 -4.453351E-04  -1.260324E-15 -1.415029E-12   4    0 1 1 2
 64  -1.659720E-02 -1.659720E-02  -2.632573E-14 -7.930777E-13   4    1 0 0 3
 65   3.657371E-05  3.657371E-05   1.441540E-16  1.970733E-12   4    0 1 0 3
 66   1.046086E-03  1.046086E-03   8.657083E-16  4.137844E-13   4    0 0 4 0
 67   4.918242E-04  4.918242E-04   4.572690E-15  4.648704E-12   4    0 0 3 1
 68   4.105017E-03  4.105017E-03  -6.131055E-15 -7.467758E-13   4    0 0 2 2
 69  -3.499706E-05 -3.499706E-05  -2.344216E-15 -3.349162E-11   4    0 0 1 3
 70  -3.118517E-05 -3.118517E-05   4.790742E-16  7.681121E-12   4    0 0 0 4
```

Table B-5: (cont.) Numerical *vs* Theoretical p_{x2}

****

I	A Coefficient	B Coefficient	Difference	Frac. Diff.	Order	Exponents
71	4.227924E+00	4.227924E+00	-1.400624E-10	-1.656397E-11	5	5 0 0 0
72	4.785582E+00	4.785582E+00	-3.476996E-11	-3.632784E-12	5	4 1 0 0
73	-2.927720E+00	-2.927720E+00	3.216927E-12	5.493911E-13	5	3 2 0 0
74	-1.872974E-01	-1.872974E-01	1.352835E-12	3.611461E-12	5	2 3 0 0
75	2.617994E-01	2.617994E-01	-1.135828E-13	-2.169271E-13	5	1 4 0 0
76	-2.208662E-18	-6.071532E-15	6.069324E-15	9.992727E-01	5	0 5 0 0
77	-2.111674E+00	-2.111674E+00	-2.757899E-11	-6.530126E-12	5	4 0 1 0
78	4.917592E+00	4.917592E+00	6.704082E-12	6.816428E-13	5	3 1 1 0
79	1.561712E-01	1.561712E-01	5.868604E-13	1.878901E-12	5	2 2 1 0
80	-1.478299E-01	-1.478299E-01	-5.037637E-14	-1.703862E-13	5	1 3 1 0
81	1.597816E-18	4.142953E-14	-4.142794E-14	-9.999229E-01	5	0 4 1 0
82	-2.612109E+00	-2.612109E+00	4.625550E-11	8.854054E-12	5	4 0 0 1
83	-1.690174E+00	-1.690174E+00	9.667600E-12	2.859943E-12	5	3 1 0 1
84	8.509564E-01	8.509564E-01	-8.639062E-13	-5.076089E-13	5	2 2 0 1
85	2.698831E-02	2.698831E-02	-2.536491E-13	-4.699240E-12	5	1 3 0 1
86		-8.501351E-15			5	0 4 0 1
87	3.854559E-01	3.854559E-01	-3.963892E-12	-5.141822E-12	5	3 0 2 0
88	-1.017042E-01	-1.017042E-01	4.260099E-13	2.094358E-12	5	2 1 2 0
89	-9.936713E-02	-9.936713E-02	2.397041E-14	1.206154E-13	5	1 2 2 0
90	-1.606875E-05	-1.606875E-05	2.124041E-14	6.609230E-10	5	0 3 2 0
91	6.880163E-01	6.880163E-01	5.655795E-12	4.110219E-12	5	3 0 1 1
92	-1.059049E+00	-1.059049E+00	-1.435019E-12	-6.775032E-13	5	2 1 1 1
93	-1.645483E-02	-1.645483E-02	-1.408387E-13	-4.279556E-12	5	1 2 1 1
94	-1.752954E-04	-1.752954E-04	9.172347E-15	2.616254E-11	5	0 3 1 1
95	5.990532E-01	5.990532E-01	-6.000811E-12	-5.008579E-12	5	3 0 0 2
96	1.877013E-01	1.877013E-01	-1.017412E-12	-2.710189E-12	5	2 1 0 2
97	-6.131948E-02	-6.131948E-02	7.323482E-14	5.971578E-13	5	1 2 0 2
98	1.606875E-05	1.606875E-05	8.692944E-15	2.704922E-10	5	0 3 0 2
99	-3.187457E-02	-3.187457E-02	1.396990E-12	2.191387E-11	5	2 0 3 0
100	4.424876E-02	4.424876E-02	-5.470017E-14	-6.180983E-13	5	1 1 3 0
101	4.322384E-04	4.322384E-04	-8.183694E-16	-9.466644E-13	5	0 2 3 0
102	-1.977308E-01	-1.977308E-01	9.211729E-13	2.329361E-12	5	2 0 2 1
103	6.989329E-03	6.989329E-03	4.056542E-14	2.901954E-12	5	1 1 2 1
104	9.507992E-03	9.507992E-03	-2.431649E-15	-1.278739E-13	5	0 2 2 1
105	-6.897910E-02	-6.897910E-02	-6.187464E-13	-4.485028E-12	5	2 0 1 2
106	4.912228E-02	4.912228E-02	1.103223E-13	1.122936E-12	5	1 1 1 2
107	-5.148751E-05	-5.148751E-05	5.983237E-15	5.810377E-11	5	0 2 1 2
108	-6.025057E-02	-6.025057E-02	3.950269E-13	3.278201E-12	5	2 0 0 3
109	-6.203401E-03	-6.203401E-03	4.059242E-14	3.271788E-12	5	1 1 0 3
110	-7.422251E-05	-7.422251E-05	-1.398961E-15	-9.424104E-12	5	0 2 0 3
111	4.510775E-03	4.510775E-03	-2.189720E-14	-2.427210E-12	5	1 0 4 0
112	-1.219511E-04	-1.219511E-04	-5.280380E-15	-2.164957E-11	5	0 1 4 0
113	5.650613E-03	5.650613E-03	-1.770451E-13	-1.566601E-11	5	1 0 3 1
114	-2.725732E-03	-2.725732E-03	3.346986E-15	6.139610E-13	5	0 1 3 1
115	2.472916E-02	2.472916E-02	-6.831688E-14	-1.381302E-12	5	1 0 2 2
116	5.147049E-05	5.147049E-05	-3.348641E-15	-3.252971E-11	5	0 1 2 2
117	1.907022E-03	1.907022E-03	4.146130E-14	1.087069E-11	5	1 0 1 3
118	1.246527E-04	1.246527E-04	-2.874369E-15	-1.152951E-11	5	0 1 1 3
119	2.195773E-03	2.195773E-03	-1.350856E-14	-3.076038E-12	5	1 0 0 4
120	-9.701224E-06	-9.701224E-06	-1.426545E-16	-7.352398E-12	5	0 1 0 4
121	-4.105443E-05	-4.105443E-05	6.160132E-17	7.502396E-13	5	0 0 5 0
122	-6.098217E-04	-6.098217E-04	1.364089E-15	1.118433E-12	5	0 0 4 1
123	-1.734589E-04	-1.734589E-04	5.139054E-15	1.481347E-11	5	0 0 3 2
124	-7.836986E-04	-7.836986E-04	1.623755E-15	1.035957E-12	5	0 0 2 3
125	9.583279E-06	9.583279E-06	-1.050838E-15	-5.482662E-11	5	0 0 1 4
126	5.777869E-06	5.777869E-06	1.711293E-16	1.480904E-11	5	0 0 0 5

Table B-6: Numerical *vs* Theoretical p_{y2} (continue next page)

```
A = TPY2        NO =   6, NV =   4, ina = 66
B = DPY2        NO =   6, NV =   4, inb = 62
********************************************************************************
 I  A Coefficient B Coefficient  Difference     Frac. Diff.    Order Exponents

 1   2.650395E-16  7.285839E-13  -7.283188E-13 -9.992727E-01    0    0 0 0 0

 2  -2.650395E-16 -1.285502E-11   1.285476E-11  9.999588E-01    1    1 0 0 0
 3   6.283185E+00  6.283185E+00  -5.256906E-13 -4.183313E-14    1    0 1 0 0
 4                 1.589007E-13                                 1    0 0 1 0
 5   3.834759E-17  9.348078E-13  -9.347694E-13 -9.999180E-01    1    0 0 0 1

 6  -1.123784E+00 -1.123784E+00   8.117035E-12  3.611474E-12    2    2 0 0 0
 7   6.283185E+00  6.283185E+00  -2.726153E-12 -2.169403E-13    2    1 1 0 0
 8  -1.325197E-16 -3.642919E-13   3.641594E-13  9.992727E-01    2    0 2 0 0
 9  -8.869795E-01 -8.869795E-01  -3.022860E-13 -1.704019E-13    2    1 0 1 0
10   3.834759E-17  9.943088E-13  -9.942705E-13 -9.999229E-01    2    0 1 1 0
11   1.619299E-01  1.619299E-01  -1.521901E-12 -4.699259E-12    2    1 0 0 1
12                -2.040252E-13                                 2    0 1 0 1
13  -9.641250E-05 -9.641250E-05   1.274423E-13  6.609219E-10    2    0 0 2 0
14  -1.051773E-03 -1.051773E-03   5.503581E-14  2.616336E-11    2    0 0 1 1
15   9.641250E-05  9.641250E-05   5.215723E-14  2.704900E-10    2    0 0 0 2

16  -2.753858E+00 -2.753858E+00   1.811384E-12  3.288812E-13    3    3 0 0 0
17   6.155483E+00  6.155483E+00  -2.785217E-12 -2.262386E-13    3    2 1 0 0
18   1.325197E-16  6.427511E-12  -6.427379E-12 -9.999588E-01    3    1 2 0 0
19  -1.047198E+00 -1.047198E+00   8.762435E-14  4.183755E-14    3    0 3 0 0
20  -2.588892E+00 -2.588892E+00   1.690592E-12  3.265088E-13    3    2 0 1 0
21  -7.996264E-02 -7.996264E-02   2.183531E-13  1.365345E-12    3    1 1 1 0
22                -7.945034E-14                                 3    0 2 1 0
23   6.345470E-01  6.345470E-01  -1.462747E-12 -1.152591E-12    3    2 0 0 1
24  -8.869795E-01 -8.869795E-01   3.880646E-13  2.187562E-13    3    1 1 0 1
25  -1.917379E-17 -4.674039E-13   4.673847E-13  9.999180E-01    3    0 2 0 1
26   1.635842E-02  1.635842E-02   6.639403E-13  2.029354E-11    3    1 0 2 0
27   6.575332E-02  6.575332E-02  -4.484607E-14 -3.410175E-13    3    0 1 2 0
28   2.442262E-01  2.442262E-01  -1.509001E-13 -3.089352E-13    3    1 0 1 1
29  -1.928250E-04 -1.928250E-04  -1.955529E-14 -5.070736E-11    3    0 1 1 1
30  -3.390786E-02 -3.390786E-02   1.404554E-13  2.071133E-12    3    1 0 0 2
31  -5.258863E-04 -5.258863E-04   1.894142E-15  1.800904E-12    3    0 1 0 2
32  -6.338661E-03 -6.338661E-03  -3.131393E-15 -2.470074E-13    3    0 0 3 0
33   1.029750E-04  1.029750E-04  -4.847874E-14 -2.353908E-10    3    0 0 2 1
34   4.453351E-04  4.453351E-04   1.260324E-15  1.415029E-12    3    0 0 1 2
35  -3.657371E-05 -3.657371E-05  -1.441540E-16 -1.970733E-12    3    0 0 0 3

36  -4.785582E+00 -4.785582E+00   3.476996E-11  3.632784E-12    4    4 0 0 0
37   5.855440E+00  5.855440E+00  -6.433853E-12 -5.493911E-13    4    3 1 0 0
38   5.618922E-01  5.618922E-01  -4.058504E-12 -3.611461E-12    4    2 2 0 0
39  -1.047198E+00 -1.047198E+00   4.543310E-13  2.169271E-13    4    1 3 0 0
40   1.104331E-17  3.035766E-14  -3.034662E-14 -9.992727E-01    4    0 4 0 0
41  -4.917592E+00 -4.917592E+00  -6.704082E-12 -6.816428E-13    4    3 0 1 0
42  -3.123423E-01 -3.123423E-01  -1.173749E-12 -1.878946E-12    4    2 1 1 0
43   4.434898E-01  4.434898E-01   1.511430E-13  1.704019E-13    4    1 2 1 0
44  -6.391264E-18 -1.657181E-13   1.657117E-13  9.999229E-01    4    0 3 1 0
45   1.690174E+00  1.690174E+00  -9.667600E-12 -2.859943E-12    4    3 0 0 1
46  -1.701913E+00 -1.701913E+00   1.727812E-12  5.076089E-13    4    2 1 0 1
47  -8.096493E-02 -8.096493E-02   7.609503E-13  4.699259E-12    4    1 2 0 1
48                 3.400535E-14                                 4    0 3 0 1
49   1.017042E-01  1.017042E-01  -4.260099E-13 -2.094358E-12    4    2 0 2 0
50   1.987343E-01  1.987343E-01  -4.794082E-14 -1.206154E-13    4    1 1 2 0
51   4.820625E-05  4.820625E-05  -6.372135E-14 -6.609241E-10    4    0 2 2 0
52   1.059049E+00  1.059049E+00   1.435019E-12  6.775032E-13    4    2 0 1 1
53   3.290966E-02  3.290966E-02   2.816775E-13  4.279556E-12    4    1 1 1 1
54   5.258863E-04  5.258863E-04  -2.751790E-14 -2.616336E-11    4    0 2 1 1
55  -1.877013E-01 -1.877013E-01   1.017412E-12  2.710189E-12    4    2 0 0 2
56   1.226390E-01  1.226390E-01  -1.464696E-13 -5.971578E-13    4    1 1 0 2
57  -4.820625E-05 -4.820625E-05  -2.607872E-14 -2.704911E-10    4    0 2 0 2
58  -4.424876E-02 -4.424876E-02   5.470017E-14  6.180983E-13    4    1 0 3 0
59  -8.644768E-04 -8.644768E-04   1.636766E-15  9.466800E-13    4    0 1 3 0
60  -6.989329E-03 -6.989329E-03  -4.056542E-14 -2.901954E-12    4    1 0 2 1
61  -1.901598E-02 -1.901598E-02   4.863731E-15  1.278853E-13    4    0 1 2 1
62  -4.912228E-02 -4.912228E-02  -1.103223E-13 -1.122936E-12    4    1 0 1 2
63   1.029750E-04  1.029750E-04  -1.196647E-14 -5.810377E-11    4    0 1 1 2
64   6.203401E-03  6.203401E-03  -4.059242E-14 -3.271788E-12    4    1 0 0 3
65   1.484450E-04  1.484450E-04   2.797909E-15  9.424058E-12    4    0 1 0 3
66   1.219511E-04  1.219511E-04   5.280380E-15  2.164957E-11    4    0 0 4 0
67   2.725732E-03  2.725732E-03  -3.347041E-15 -6.139710E-13    4    0 0 3 1
68  -5.147049E-05 -5.147049E-05   3.348641E-15  3.252971E-11    4    0 0 2 2
69  -1.246527E-04 -1.246527E-04   2.874369E-15  1.152951E-11    4    0 0 1 3
70   9.701224E-06  9.701224E-06   1.426545E-16  7.352398E-12    4    0 0 0 4
```

Table B-6: (cont.) Numerical *vs* Theoretical p_{y2}

```
********************************************************************************
  I  A Coefficient B Coefficient  Difference     Frac. Diff.   Order Exponents

 71  -7.057481E+00 -7.057481E+00  -6.799530E-10 -4.817250E-11    5    5 0 0 0
 72   5.248884E+00  5.248884E+00   2.749534E-11  2.619161E-12    5    4 1 0 0
 73   1.376929E+00  1.376929E+00  -9.056922E-13 -3.288812E-13    5    3 2 0 0
 74  -1.025914E+00 -1.025914E+00   4.642120E-13  2.262432E-13    5    2 3 0 0
 75  -1.104331E-17 -5.356271E-13   5.356161E-13  9.999588E-01    5    1 4 0 0
 76   5.235988E-02  5.235988E-02  -4.381044E-15 -4.183589E-14    5    0 5 0 0
 77  -7.545949E+00 -7.545949E+00   5.691814E-11  3.771437E-12    5    4 0 1 0
 78  -9.096820E-01 -9.096820E-01   1.427569E-11  7.846529E-12    5    3 1 1 0
 79   1.294446E+00  1.294446E+00  -8.452961E-13 -3.265088E-13    5    2 2 1 0
 80   1.332711E-02  1.332711E-02  -3.639493E-14 -1.365448E-12    5    1 3 1 0
 81                 6.619705E-15                                 5    0 4 1 0
 82   3.439954E+00  3.439954E+00   2.255659E-10  3.278618E-11    5    4 0 0 1
 83  -2.332066E+00 -2.332066E+00  -7.447043E-12 -1.596662E-12    5    3 1 0 1
 84  -3.172735E-01 -3.172735E-01   7.313594E-13  1.152569E-12    5    2 2 0 1
 85   1.478299E-01  1.478299E-01  -6.468437E-14 -2.187797E-13    5    1 3 0 1
 86   1.597816E-18  3.895040E-14  -3.894880E-14 -9.999180E-01    5    0 4 0 1
 87   3.705795E-01  3.705795E-01   7.728325E-12  1.042735E-11    5    3 0 2 0
 88   3.395090E-01  3.395090E-01  -1.017200E-12 -1.498046E-12    5    2 1 2 0
 89  -8.179208E-03 -8.179208E-03  -3.319703E-13 -2.029355E-11    5    1 2 2 0
 90  -1.095889E-02 -1.095889E-02   7.474707E-15  3.410340E-13    5    0 3 2 0
 91   2.630256E+00  2.630256E+00  -1.455175E-11 -2.766223E-12    5    3 0 1 1
 92   1.943032E-01  1.943032E-01  -2.550071E-12 -6.562094E-12    5    2 1 1 1
 93  -1.221131E-01 -1.221131E-01   7.544659E-14  3.089210E-13    5    1 2 1 1
 94   3.213750E-05  3.213750E-05   3.259324E-15  5.070905E-11    5    0 3 1 1
 95  -6.035812E-01 -6.035812E-01  -2.863887E-11 -2.372412E-11    5    3 0 0 2
 96   3.437385E-01  3.437385E-01   6.953327E-13  1.011427E-12    5    2 1 0 2
 97   1.695393E-02  1.695393E-02  -7.022724E-14 -2.071120E-12    5    1 2 0 2
 98   8.764772E-05  8.764772E-05  -3.156909E-16 -1.800907E-12    5    0 3 0 2
 99  -1.579297E-01 -1.579297E-01   8.487655E-14  2.687162E-13    5    2 0 3 0
100  -7.712473E-03 -7.712473E-03  -5.077048E-14 -3.291452E-12    5    1 1 3 0
101   3.169331E-03  3.169331E-03   1.565805E-15  2.470245E-13    5    0 2 3 0
102  -6.331407E-02 -6.331407E-02  -1.625797E-12 -1.283914E-11    5    2 0 2 1
103  -8.603333E-02 -8.603333E-02   1.604515E-13  9.324962E-13    5    1 1 2 1
104  -5.148751E-05 -5.148751E-05   2.423959E-14  2.353929E-10    5    0 2 2 1
105  -2.829471E-01 -2.829471E-01   1.335460E-12  2.359911E-12    5    2 0 1 2
106  -8.800394E-03 -8.800394E-03   1.355478E-13  7.701236E-12    5    1 1 1 2
107  -2.226675E-04 -2.226675E-04  -6.301688E-16 -1.415044E-12    5    0 2 1 2
108   4.402247E-02  4.402247E-02   1.727197E-12  1.961723E-11    5    2 0 0 3
109  -1.659720E-02 -1.659720E-02  -2.632573E-14 -7.930777E-13    5    1 1 0 3
110   1.828685E-05  1.828685E-05   7.207700E-17  1.970733E-12    5    0 2 0 3
111   1.861368E-03  1.861368E-03  -4.316588E-14 -1.159520E-11    5    1 0 4 0
112   1.046086E-03  1.046086E-03   8.657083E-16  4.137844E-13    5    0 1 4 0
113   2.524411E-02  2.524411E-02  -1.279445E-14 -2.534146E-13    5    1 0 3 1
114   4.918242E-04  4.918242E-04   4.572704E-15  4.648718E-12    5    0 1 3 1
115   1.953214E-03  1.953214E-03   1.311885E-13  3.358273E-11    5    1 0 2 2
116   4.105017E-03  4.105017E-03  -6.131055E-15 -7.467758E-13    5    0 1 2 2
117   8.532288E-03  8.532288E-03  -5.178019E-14 -3.034367E-12    5    1 0 1 3
118  -3.499706E-05 -3.499706E-05  -2.344216E-15 -3.349162E-11    5    0 1 1 3
119  -1.045268E-03 -1.045268E-03  -4.980692E-14 -2.382496E-11    5    1 0 0 4
120  -3.118517E-05 -3.118517E-05   4.790708E-16  7.681067E-12    5    0 1 0 4
121  -1.880731E-04 -1.880731E-04   4.330337E-16  1.151238E-12    5    0 0 5 0
122  -8.508144E-05 -8.508144E-05   1.692228E-15  9.944753E-12    5    0 0 4 1
123  -7.754148E-04 -7.754148E-04   4.254003E-16  2.743050E-13    5    0 0 3 2
124   1.847613E-05  1.847613E-05  -3.515945E-15 -9.514829E-11    5    0 0 2 3
125   2.882712E-05  2.882712E-05   7.235724E-16  1.255020E-11    5    0 0 1 4
126  -2.183783E-06 -2.183783E-06   5.801024E-16  1.328205E-10    5    0 0 0 5
```

Appendix C

Numerical Results — Uniform B-Field

C.1 $\theta = 3\pi/4$, Nstep $= 30$

The Hamiltonian used for this pair of runs was the "particle in a uniform magnetic field". Initial conditions for this run were as follows:

```
%
% HJDAubfield: Particle in Uniform Magnetic Field
%
%          no =    6
%       nstep =   30
%       istep =    5
%         eps =  1.000000E-30
%
%         R_i =  1.100000E+00
%        Pr_i =  1.000000E-01
%
%      btgma0 =  2.000000E+00
%      btgma  =  2.000000E+00
%
%    Theta/Pi =  7.500000E-01
%
```

Table C-1: Reference Trajectory and Matrix at θ_2

```
 x2               r2
===============  ============================
 9.580246E-01;    -7.831684E-01,  2.959188E-01
-2.139009E-01;    -1.414214E+00, -7.425064E-01

 Symplecticity check: r2 * ( J r2^T J^-1) - I
=============================================
                   3.330669E-16,  0.000000E+00
                   0.000000E+00,  3.330669E-16
```

Table C-2: Generating Function DS_{12} for Nonlinear Part

```
DS12         NO =    6, NV =    2, INA = 30
*********************************************
   I   COEFFICIENT              ORDER   EXPONENTS

    ALL ORDER =   0 COMPONENTS ZERO
    ALL ORDER =   1 COMPONENTS ZERO

   1    1.00000000000000E+00    2      1 1

   2   -1.19902474675693E-01    3      3 0
   3   -1.88857659591831E-01    3      2 1
   4   -9.91561841765567E-02    3      1 2
   5   -5.91767734245905E-02    3      0 3

   6    2.70873076393160E-02    4      4 0
   7    5.68867383178616E-02    4      3 1
   8    4.48009780014250E-02    4      2 2
   9    1.56812774078993E-02    4      1 3
  10    1.27213614476274E-03    4      0 4

  11   -1.72558379916914E-02    5      5 0
  12   -4.52992760188530E-02    5      4 1
  13   -4.75670763441395E-02    5      3 2
  14   -2.49742043447617E-02    5      2 3
  15   -6.55611959563663E-03    5      1 4
  16   -2.28044523482362E-03    5      0 5

  17    9.71743155865738E-03    6      6 0
  18    3.06117348516066E-02    6      5 1
  19    4.01802980280223E-02    6      4 2
  20    2.81278859443401E-02    6      3 3
  21    1.10760155765804E-02    6      2 4
  22    2.32610198141271E-03    6      1 5
  23    1.04454375976331E-04    6      0 6
```

Table C-3: Comparison of Orbit Center Coordinates x_{*1} and x_{*2}

```
A = XC1          NO =    6, NV =     2, ina = 47
B = XC2          NO =    6, NV =     2, inb = 49
********************************************************************************
  I  A Coefficient B Coefficient  Difference      Frac. Diff.   Order Exponents

  1   5.000000E-02  5.000000E-02  -9.341486E-16 -9.341486E-15     0    0 0

  2                -1.068590E-15                                  1    1 0
  3   5.000000E-01  5.000000E-01   1.526557E-16  1.526557E-16     1    0 1

  4                 5.585810E-16                                  2    2 0

  5                -2.155394E-16                                  3    3 0

  6                 3.920475E-16                                  4    4 0

  7                -1.153157E-15                                  5    5 0
```

Table C-4: Comparison of Orbit Center Coordinates y_{*1} and y_{*2}

```
A = YC1          NO =    6, NV =     2, ina = 48
B = YC2          NO =    6, NV =     2, inb = 50
********************************************************************************
  I  A Coefficient B Coefficient  Difference      Frac. Diff.   Order Exponents

  1   1.012508E-01  1.012508E-01   7.147061E-16  3.529385E-15     0    0 0

  2   1.000000E+00  1.000000E+00  -5.828671E-16 -2.914335E-16     1    1 0
  3   2.503131E-02  2.503131E-02  -4.445229E-16 -8.879338E-15     1    0 1

  4  -1.888577E-01 -1.888577E-01  -3.816392E-17 -1.010388E-16     2    2 0
  5  -1.983124E-01 -1.983124E-01   1.110223E-16  2.799177E-16     2    1 1
  6  -5.206010E-02 -5.206010E-02   5.204170E-17  4.998233E-16     2    0 2

  7   5.688674E-02  5.688674E-02  -7.546047E-17 -6.632519E-16     3    3 0
  8   8.960196E-02  8.960196E-02   3.816392E-17  2.129636E-16     3    2 1
  9   4.704383E-02  4.704383E-02   5.724587E-17  6.084312E-16     3    1 2
 10   8.233162E-03  8.233162E-03   4.553649E-18  2.765432E-16     3    0 3

 11  -4.529928E-02 -4.529928E-02  -4.683753E-17 -5.169788E-16     4    4 0
 12  -9.513415E-02 -9.513415E-02  -1.387779E-17 -7.293799E-17     4    3 1
 13  -7.492261E-02 -7.492261E-02  -4.336809E-17 -2.894192E-16     4    2 2
 14  -2.622448E-02 -2.622448E-02  -1.301043E-18 -2.480588E-17     4    1 3
 15  -3.442168E-03 -3.442168E-03  -3.501973E-17 -5.086871E-15     4    0 4

 16   3.061173E-02  3.061173E-02   2.602085E-17  4.250143E-16     5    5 0
 17   8.036060E-02  8.036060E-02  -3.608225E-16 -2.245021E-15     5    4 1
 18   8.438366E-02  8.438366E-02  -3.243933E-16 -1.922133E-15     5    3 2
 19   4.430406E-02  4.430406E-02  -2.775558E-17 -3.132396E-16     5    2 3
 20   1.163051E-02  1.163051E-02   1.431147E-17  6.152554E-16     5    1 4
 21   1.221276E-03  1.221276E-03   1.241411E-17  5.082435E-15     5    0 5
```

C.2 $\theta = 2\pi$, Nstep $= 80$

Initial conditions for this second run were as follows:

```
%
% HJDAubfield: Particle in Uniform Magnetic Field
%
%        no =    6
%     nstep =   80
%     istep =    5
%       eps =  1.000000E-30
%
%       R_i =  1.100000E+00
%      Pr_i =  1.000000E-01
%
%    btgma0 =  2.000000E+00
%    btgma  =  2.000000E+00
%
%  Theta/Pi =  2.000000E+00
%
```

Table C-5: Reference Trajectory and Matrix at θ_2

```
 x2                 r2
===============   =============================
 1.100000E+00;      1.000000E+00,  1.037240E-15
 1.000000E-01;     -4.460940E-15,  1.000000E+00

 Symplecticity check: r2 * ( J r2^T J^-1) - I
===============================================
                    6.106227E-16,  0.000000E+00
                    0.000000E+00,  6.106227E-16
```

Table C-6: Generating Function DS_{12} for Nonlinear Part

```
DS12        NO =    6, NV =    2, INA = 30
*********************************************
   I   COEFFICIENT           ORDER   EXPONENTS

    ALL ORDER =    0 COMPONENTS ZERO
    ALL ORDER =    1 COMPONENTS ZERO

   1   1.00000000000000E+00   2      1 1

   2  -9.73649630983299E-17   3      3 0
   3  -2.94156275898013E-16   3      2 1
   4  -6.84970263060207E-16   3      1 2
   5  -2.32089625640220E-16   3      0 3

   6   1.03538494485060E-16   4      4 0
   7   2.31273081086090E-17   4      3 1
   8   4.79044082980069E-17   4      2 2
   9  -5.76525073441647E-17   4      1 3
  10  -2.84399772180143E-17   4      0 4

  11   5.40183941875706E-17   5      5 0
  12  -6.39739683078226E-17   5      4 1
  13   4.81378310947689E-18   5      3 2
  14  -2.28971268059336E-17   5      2 3
  15  -2.08236528802238E-17   5      1 4
  16  -1.59992295904328E-17   5      0 5

  17  -7.40507241315178E-16   6      6 0
  18  -3.00826260165498E-16   6      5 1
  19  -3.84638519297185E-16   6      4 2
  20  -1.59277776986345E-16   6      3 3
  21  -1.48979394264212E-18   6      2 4
  22  -2.33887665519119E-17   6      1 5
  23   6.15458737494442E-18   6      0 6
```

Table C-7: Comparison of Orbit Center Coordinates x_{*1} and x_{*2}

```
A = XC1        NO =    6, NV =    2, ina = 47
B = XC2        NO =    6, NV =    2, inb = 49
****************************************************************************
 I  A Coefficient  B Coefficient  Difference    Frac. Diff.    Order Exponents

 1   5.000000E-02  5.000000E-02  -7.832276E-16 -7.832276E-15    0     0 0

 2                -2.253355E-15                                 1     1 0
 3   5.000000E-01  5.000000E-01   3.816392E-16  3.816392E-16    1     0 1

 4                -1.460474E-16                                 2     2 0

 5                 2.070770E-16                                 3     3 0

 6                 1.350460E-16                                 4     4 0

 7                -2.221522E-15                                 5     5 0
```

Table C-8: Comparison of Orbit Center Coordinates y_{*1} and y_{*2}

```
A = YC1         NO =    6, NV =    2, ina = 48
B = YC2         NO =    6, NV =    2, inb = 50
********************************************************************************
  I  A Coefficient B Coefficient  Difference     Frac. Diff.    Order Exponents

  1    1.012508E-01  1.012508E-01  1.344411E-15  6.639014E-15     0     0 0

  2    1.000000E+00  1.000000E+00 -1.276756E-15 -6.383782E-16     1     1 0
  3    2.503131E-02  2.503131E-02 -1.423341E-15 -2.843121E-14     1     0 1

  4   -2.941563E-16 -7.311517E-18 -2.868448E-16 -9.514939E-01     2     2 0
  5   -1.369941E-15 -1.134157E-15 -2.357840E-16 -9.415930E-02     2     1 1
  6    1.254702E-01  1.254702E-01 -5.030698E-16 -2.004738E-15     2     0 2

  7    2.312731E-17  1.036682E-17  1.276049E-17  3.809770E-01     3     3 0
  8    9.580882E-17 -7.156170E-17  1.673705E-16  1.000000E+00     3     2 1
  9   -1.729575E-16 -1.873170E-16  1.435951E-17  3.985714E-02     3     1 2
 10    3.144617E-03  3.144617E-03  1.675092E-17  2.663428E-15     3     0 3

 11   -6.397397E-17  6.760756E-18 -7.073472E-17 -1.000000E+00     4     4 0
 12    9.627566E-18  9.752257E-17 -8.789501E-17 -8.202976E-01     4     3 1
 13   -6.869138E-17  1.501664E-17 -8.370802E-17 -1.000000E+00     4     2 2
 14   -8.329461E-17 -1.246914E-16  4.139680E-17  1.990364E-01     4     1 3
 15    7.960058E-03  7.960058E-03 -3.881444E-17 -2.438075E-15     4     0 4

 16   -3.008263E-16 -1.112152E-16 -1.896111E-16 -4.601747E-01     5     5 0
 17   -7.692770E-16  3.012662E-17 -7.994037E-16 -1.000000E+00     5     4 1
 18   -4.778333E-16 -9.881958E-17 -3.790138E-16 -6.572650E-01     5     3 2
 19   -5.959176E-18 -1.698271E-17  1.102353E-17  4.804981E-01     5     2 3
 20   -1.169438E-16 -4.265564E-17 -7.428820E-17 -4.654664E-01     5     1 4
 21    5.945501E-04  5.945501E-04  5.776087E-17  4.857528E-14     5     0 5
```

Appendix D

Numerical Results — Convergence Study

The Hamiltonian used for this series of runs was the "particle in a uniform magnetic field". Initial conditions were as follows:

```
%         no =     6
%      nstep =     8--128
%      istep =     2--6
%        eps =  1.000000E-30
%
%        R_i =  1.100000E+00
%       Pr_i =  1.000000E-01
%
%    btgma0 =  2.000000E+00
%    btgma  =  2.000000E+00
%
%   Theta/Pi =  7.500000E-01
```

Table D-1: Matrix Max-Norm of Symplecticity Error

			Istep		
Nstep	2	3	4	5	6
8	4.3255E-05	1.4004E-07	3.8900E-09	2.5965E-11	1.0636E-13
16	1.7103E-06	5.6948E-11	6.3016E-12	1.2795E-14	5.8287E-16
32	5.5742E-08	2.4746E-12	1.2185E-14	1.1102E-16	5.4123E-16
64	1.7545E-09	2.2787E-14	6.1062E-16	8.0491E-16	5.8287E-16
128	5.4844E-11	1.1380E-15	1.6098E-15	5.5511E-16	4.5797E-16

Table D-2: One-Norm of Error Measure — All Orders

	Istep				
Nstep	2	3	4	5	6
8	8.7363E-05	1.1586E-06	8.2371E-09	3.6458E-11	1.8395E-13
16	4.7546E-06	1.6343E-08	3.0054E-11	3.2756E-14	1.5609E-16
32	2.7570E-07	2.3849E-10	1.1050E-13	1.0717E-16	2.2938E-16
64	1.6593E-08	3.5820E-12	4.1035E-16	1.3580E-16	1.6327E-16
128	1.0212E-09	5.4930E-14	1.9134E-16	1.0803E-16	3.4782E-16

Table D-3: Max-Norm of Error Measure — All Orders

	Istep				
Nstep	2	3	4	5	6
8	9.6068E-04	8.1473E-06	6.2660E-08	3.5457E-10	1.5664E-12
16	5.7081E-05	1.2726E-07	3.1962E-10	4.4822E-13	1.4303E-15
32	3.4779E-06	1.9798E-09	1.3288E-12	9.2981E-16	2.8987E-15
64	2.1476E-07	3.0897E-11	5.2358E-15	8.5348E-16	1.2598E-15
128	1.3345E-08	4.8369E-13	1.6445E-15	1.1657E-15	3.8008E-15

Table D-4: One-Norm of Error Measure — Zeroth Order

	Istep				
Nstep	2	3	4	5	6
8	1.4585E-04	1.2221E-06	5.6287E-09	1.4608E-11	1.2657E-14
16	8.6334E-06	1.9119E-08	2.3363E-11	1.7544E-14	2.0010E-15
32	5.2445E-07	2.9721E-10	9.3363E-14	1.0339E-15	3.2196E-15
64	3.2331E-08	4.6308E-12	7.5981E-16	4.7184E-16	3.1572E-16
128	2.0073E-09	7.2026E-14	8.4394E-16	1.3548E-15	4.5198E-15

Table D-5: Max-Norm of Error Measure — Zeroth Order

	Istep				
Nstep	2	3	4	5	6
8	9.5403E-05	8.1925E-07	3.8990E-09	1.0734E-11	1.1559E-14
16	5.6817E-06	1.2722E-08	1.5789E-11	1.2055E-14	1.4303E-15
32	3.4703E-07	1.9768E-10	6.2488E-14	9.2981E-16	2.8987E-15
64	2.1460E-08	3.0803E-12	6.5573E-16	3.7990E-16	2.0817E-16
128	1.3345E-09	4.7467E-14	6.2884E-16	1.1657E-15	3.8008E-15

Table D-6: One-Norm of Error Measure — First Order

			Istep		
Nstep	2	3	4	5	6
8	3.7734E-04	3.1842E-06	1.5105E-08	4.3018E-11	5.2152E-14
16	2.2348E-05	4.9804E-08	6.1246E-11	4.3498E-14	6.0878E-16
32	1.3581E-06	7.7435E-10	2.4480E-13	2.6617E-16	5.6997E-16
64	8.3744E-08	1.2067E-11	9.2797E-16	4.4506E-16	6.0119E-16
128	5.2001E-09	1.8899E-13	9.8109E-16	4.2425E-16	1.4972E-15

Table D-7: Max-Norm of Error Measure — First Order

			Istep		
Nstep	2	3	4	5	6
8	9.6068E-04	8.1473E-06	3.8140E-08	1.0202E-10	9.0504E-14
16	5.7081E-05	1.2726E-07	1.5692E-10	1.0947E-13	1.0270E-15
32	3.4779E-06	1.9798E-09	6.2583E-13	4.1937E-16	7.3552E-16
64	2.1476E-07	3.0865E-11	2.1649E-15	8.5348E-16	1.2598E-15
128	1.3345E-08	4.8304E-13	1.6445E-15	7.2858E-16	1.7907E-15

Table D-8: One-Norm of Error Measure — Second Order

			Istep		
Nstep	2	3	4	5	6
8	7.6389E-05	6.5523E-07	3.1885E-09	1.0963E-11	3.8708E-14
16	4.5539E-06	1.0307E-08	1.2915E-11	9.2793E-15	2.5349E-16
32	2.7806E-07	1.6094E-10	5.1824E-14	1.6032E-16	2.3383E-16
64	1.7191E-08	2.5138E-12	1.4369E-16	1.8887E-16	3.6133E-16
128	1.0690E-09	3.9403E-14	2.9714E-16	2.6202E-16	6.5992E-16

Table D-9: Max-Norm of Error Measure — Second Order

			Istep		
Nstep	2	3	4	5	6
8	1.7013E-04	1.5262E-06	7.9260E-09	2.5574E-11	8.0630E-14
16	9.9042E-06	2.2670E-08	2.9305E-11	2.2067E-14	8.5522E-16
32	5.9660E-07	3.4293E-10	1.1024E-13	5.1695E-16	7.6848E-16
64	3.6610E-08	5.2672E-12	3.3307E-16	6.9389E-16	9.5757E-16
128	2.2675E-09	8.2092E-14	4.7184E-16	5.2042E-16	1.7087E-15

Table D-10: One-Norm of Error Measure — Third Order

	Istep				
Nstep	2	3	4	5	6
8	8.2636E-05	9.6916E-07	6.0277E-09	2.3532E-11	6.0571E-14
16	5.2976E-06	1.6329E-08	2.6309E-11	2.6570E-14	6.4239E-17
32	3.3389E-07	2.6025E-10	1.0578E-13	8.4093E-17	2.2602E-16
64	2.0941E-08	4.0932E-12	4.1068E-16	1.3799E-16	1.2640E-16
128	1.3110E-09	6.4109E-14	1.2097E-16	7.2398E-17	2.1489E-16

Table D-11: Max-Norm of Error Measure — Third Order

	Istep				
Nstep	2	3	4	5	6
8	2.7359E-04	3.2444E-06	1.9296E-08	7.0275E-11	1.6394E-13
16	1.7174E-05	5.2564E-08	7.9429E-11	7.2648E-14	3.0791E-16
32	1.0755E-06	8.2593E-10	3.1176E-13	2.9837E-16	9.4933E-16
64	6.7299E-08	1.2907E-11	1.1800E-15	5.4904E-16	2.7409E-16
128	4.2091E-09	2.0139E-13	2.4199E-16	1.6567E-16	5.5077E-16

Table D-12: One-Norm of Error Measure — Fourth Order

	Istep				
Nstep	2	3	4	5	6
8	7.2452E-05	9.7214E-07	6.2704E-09	2.6137E-11	1.7642E-13
16	3.4235E-06	1.1909E-08	1.9725E-11	1.9272E-14	7.5881E-17
32	1.8891E-07	1.6773E-10	7.1463E-14	5.9848E-17	1.9396E-16
64	1.1062E-08	2.4733E-12	2.7637E-16	1.3751E-16	1.4704E-16
128	6.9157E-10	3.7498E-14	1.5688E-16	3.5592E-17	1.5850E-16

Table D-13: Max-Norm of Error Measure — Fourth Order

	Istep				
Nstep	2	3	4	5	6
8	1.7701E-04	2.7549E-06	2.0074E-08	8.9340E-11	4.7668E-13
16	1.1032E-05	4.0305E-08	7.1566E-11	8.9522E-14	3.6299E-16
32	7.1724E-07	6.5739E-10	2.9129E-13	2.7886E-16	7.4116E-16
64	4.5839E-08	1.0501E-11	1.1848E-15	4.9570E-16	5.8200E-16
128	2.8984E-09	1.6574E-13	3.4261E-16	8.1532E-17	4.6491E-16

Table D-14: One-Norm of Error Measure — Fifth Order

Nstep	Istep 2	3	4	5	6
8	1.1610E-04	2.4595E-06	2.2098E-08	1.1163E-10	6.3326E-13
16	5.3578E-06	3.2108E-08	7.7456E-11	9.8486E-14	1.2593E-16
32	2.7116E-07	4.3630E-10	2.7031E-13	1.3913E-16	1.8295E-16
64	1.5052E-08	6.2609E-12	9.6639E-16	1.4506E-16	1.4773E-16
128	8.8032E-10	9.3651E-14	1.3560E-16	4.0867E-17	1.4217E-16

Table D-15: Max-Norm of Error Measure — Fifth Order

Nstep	Istep 2	3	4	5	6
8	4.2670E-04	7.4183E-06	6.2660E-08	3.5457E-10	1.5664E-12
16	1.6992E-05	1.1878E-07	3.1962E-10	4.4822E-13	4.2848E-16
32	9.5011E-07	1.9561E-09	1.3288E-12	5.9436E-16	7.8475E-16
64	5.9088E-08	3.0897E-11	5.2358E-15	5.1174E-16	5.0307E-16
128	3.6685E-09	4.8369E-13	4.0853E-16	1.0755E-16	5.1088E-16

Appendix E

Numerical Results — Lithium Lens

Table E-1: Contributions to $\sigma_2(3,3) := \langle P_{X,2}^2 \rangle$ made by each order. Columns labeled by `Ord` give order of contribution. Rows labeled by `Tot` give total contribution to $\sigma(3,3)$ summed through order `Ord`. Rows labeled by `Inc` give incremental contribution to $\sigma(3,3)$ made by order `Ord`.

For comparison, the value of $\sigma_2(1,3) := \langle X_2 P_{X,2} \rangle$ is also given; note that $\sigma(1,3)$ changes sign, as it should, as $\sigma(3,3)$ passes through the minimum near $L_{lens} \simeq (1.0826)\, L_0$, where $L_0 := \lambda/4$.

```
LLens         = 29.2794             (LLens/L0)    =  1.08244
Sigma_2(3,3) =  1.068445E-02       Sigma_2(1,3) =  5.140557E-04

Ord |      2            4            6            8           10
==========================================================================
Tot | 1.068086E-02  1.034107E-02  1.065238E-02  1.068151E-02  1.068445E-02
Inc | 1.068086E-02 -3.397882E-04  3.113170E-04  2.912617E-05  2.942961E-06

LLens         = 29.2815             (LLens/L0)    =  1.08252
Sigma_2(3,3) =  1.068440E-02       Sigma_2(1,3) =  2.724777E-04

Ord |      2            4            6            8           10
==========================================================================
Tot | 1.067701E-02  1.034085E-02  1.065232E-02  1.068146E-02  1.068440E-02
Inc | 1.067701E-02 -3.361656E-04  3.114788E-04  2.913545E-05  2.943701E-06

LLens         = 29.2837             (LLens/L0)    =  1.08260
Sigma_2(3,3) =  1.068438E-02       Sigma_2(1,3) =  3.089970E-05

Ord |      2            4            6            8           10
==========================================================================
Tot | 1.067320E-02  1.034065E-02  1.065229E-02  1.068144E-02  1.068438E-02
Inc | 1.067320E-02 -3.325426E-04  3.116407E-04  2.914474E-05  2.944438E-06

LLens         = 29.2859             (LLens/L0)    =  1.08268
Sigma_2(3,3) =  1.068439E-02       Sigma_2(1,3) = -2.106783E-04

Ord |      2            4            6            8           10
==========================================================================
Tot | 1.066941E-02  1.034049E-02  1.065230E-02  1.068145E-02  1.068439E-02
Inc | 1.066941E-02 -3.289191E-04  3.118026E-04  2.915402E-05  2.945178E-06

LLens         = 29.2880             (LLens/L0)    =  1.08276
Sigma_2(3,3) =  1.068444E-02       Sigma_2(1,3) = -4.522563E-04

Ord |      2            4            6            8           10
==========================================================================
ToT | 1.066566E-02  1.034036E-02  1.065233E-02  1.068149E-02  1.068444E-02
Inc | 1.066566E-02 -3.252952E-04  3.119644E-04  2.916330E-05  2.945918E-06
```

Appendix F

Figures

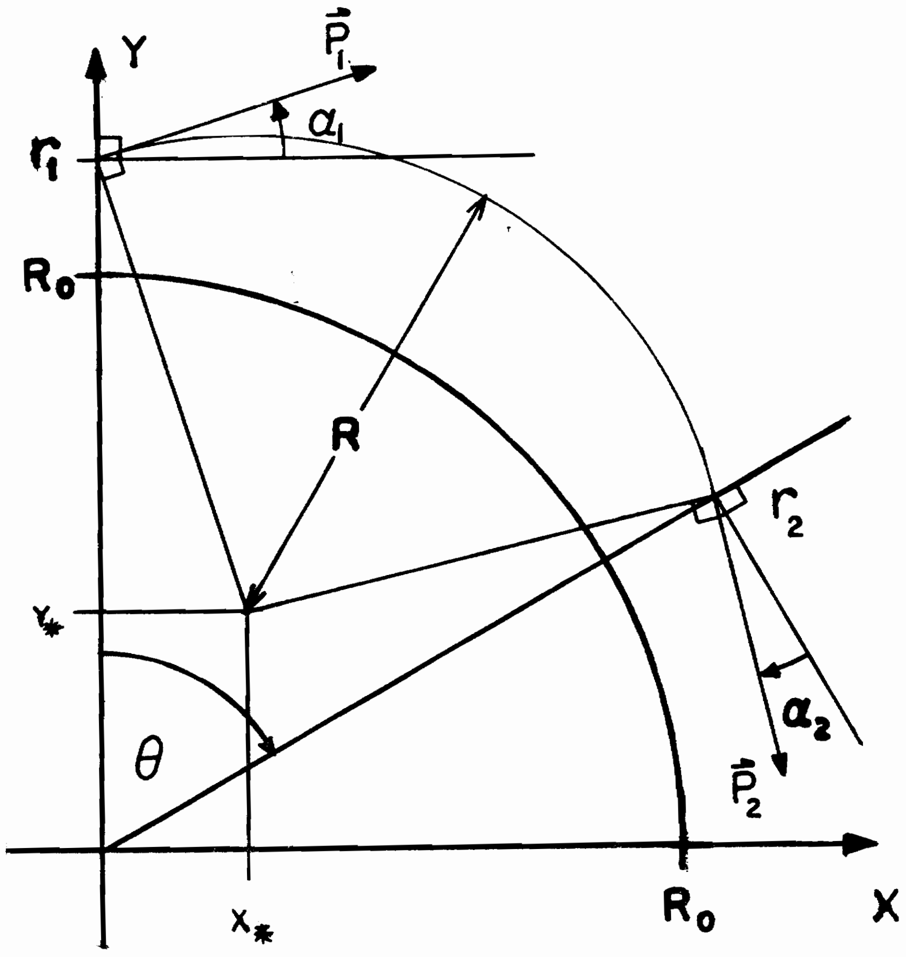

Figure F-1: Geometry for uniform magnetic field test problem. Note: θ is measured clockwise, α_1 and α_2 are measured counterclockwise.

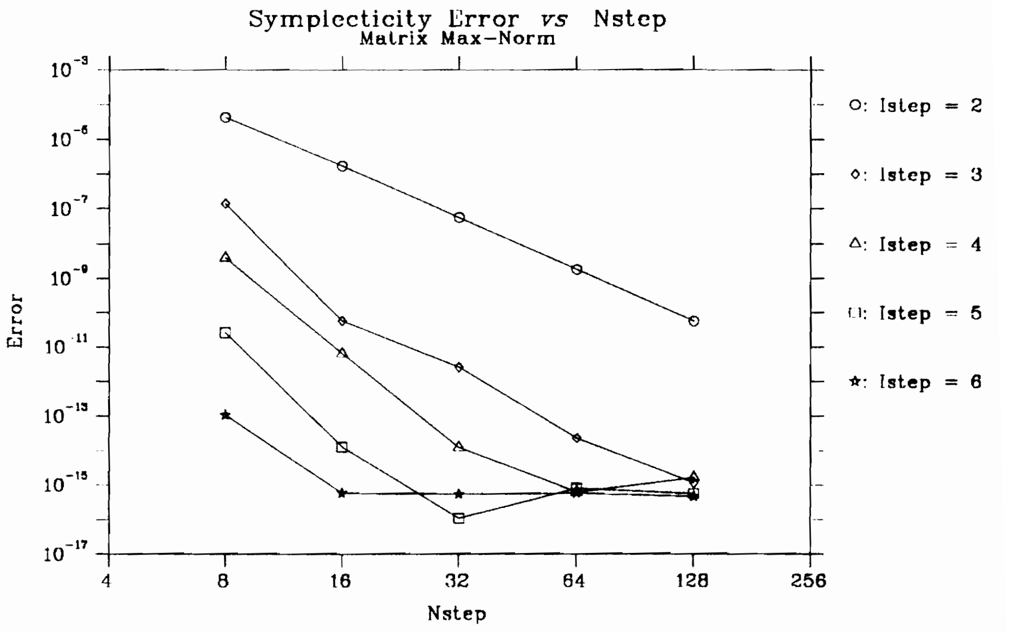

Figure F-2: Symplecticity Error *vs* **Nstep**

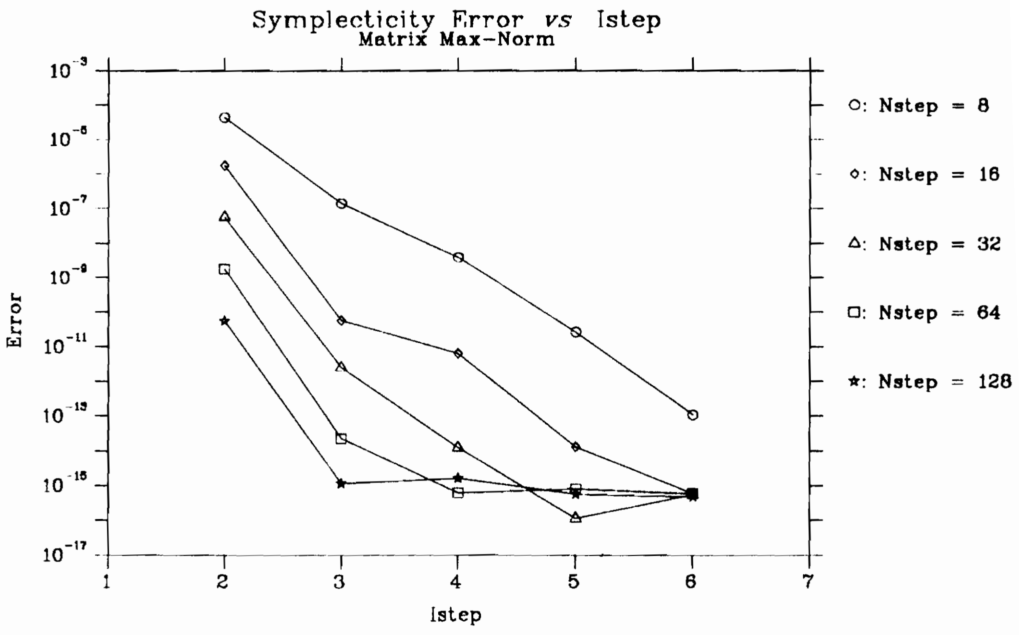

Figure F-3: Symplecticity Error *vs* **Istep**

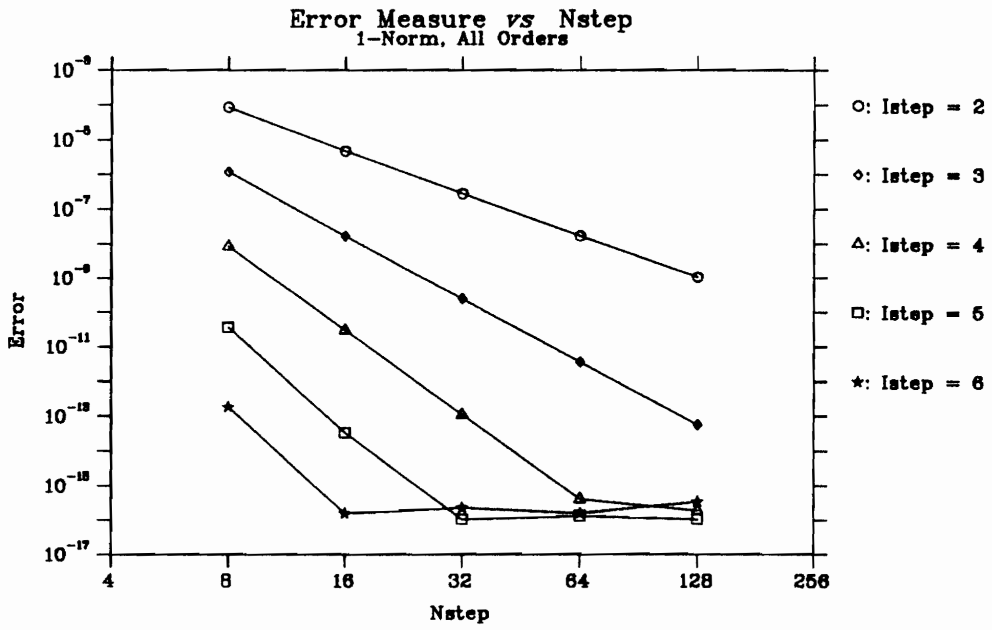

Figure F-4: 1-Norm of Error Measure *vs* **Nstep** (All Orders).

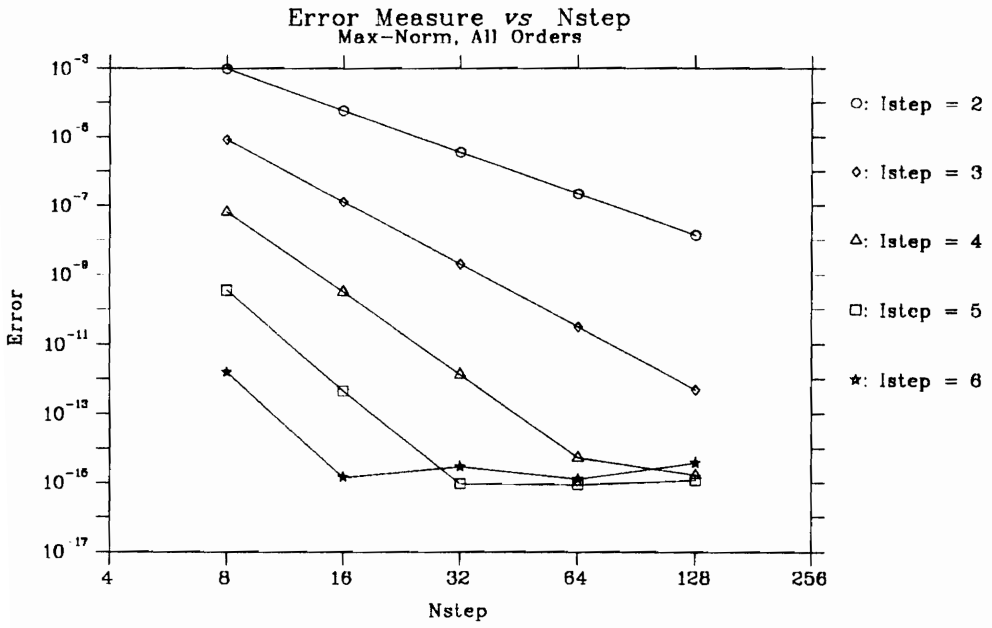

Figure F-5: Max-Norm of Error Measure *vs* **Nstep** (All Orders).

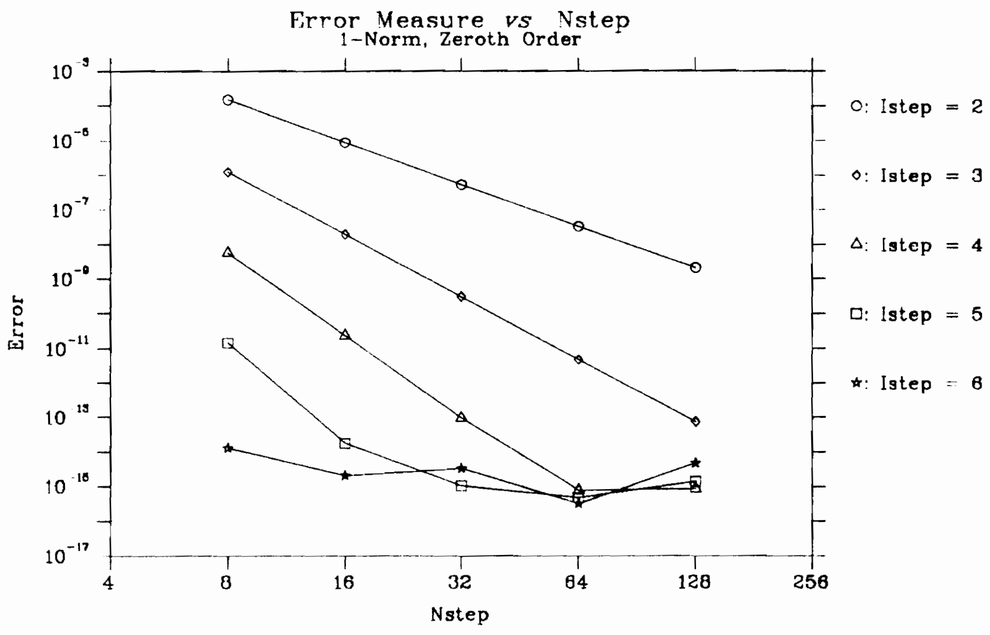

Figure F-6: 1-Norm of Error Measure *vs* **Nstep** (Zeroth Order).

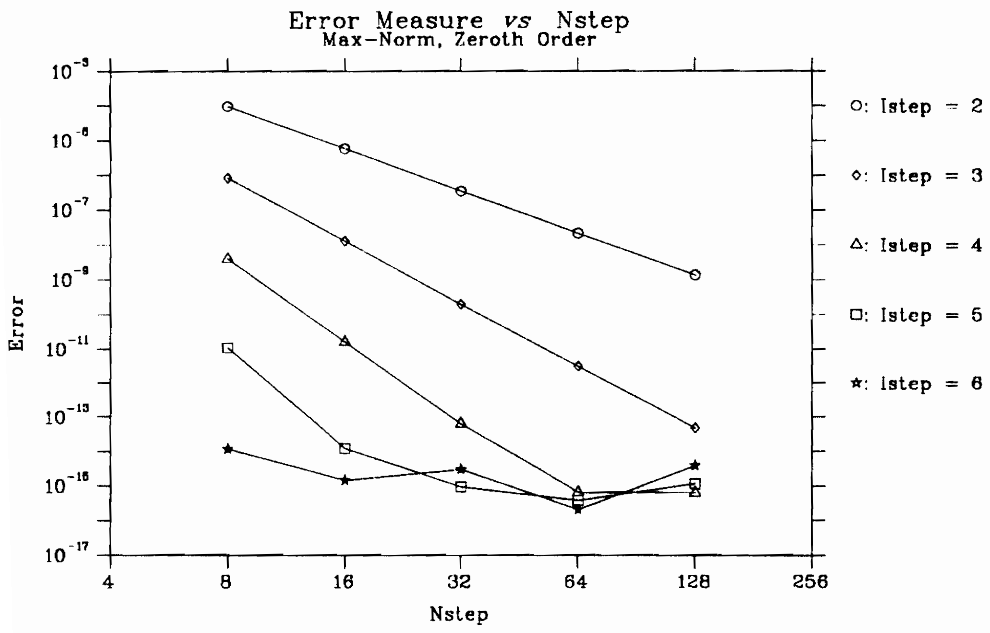

Figure F-7: Max-Norm of Error Measure *vs* **Nstep** (Zeroth Order).

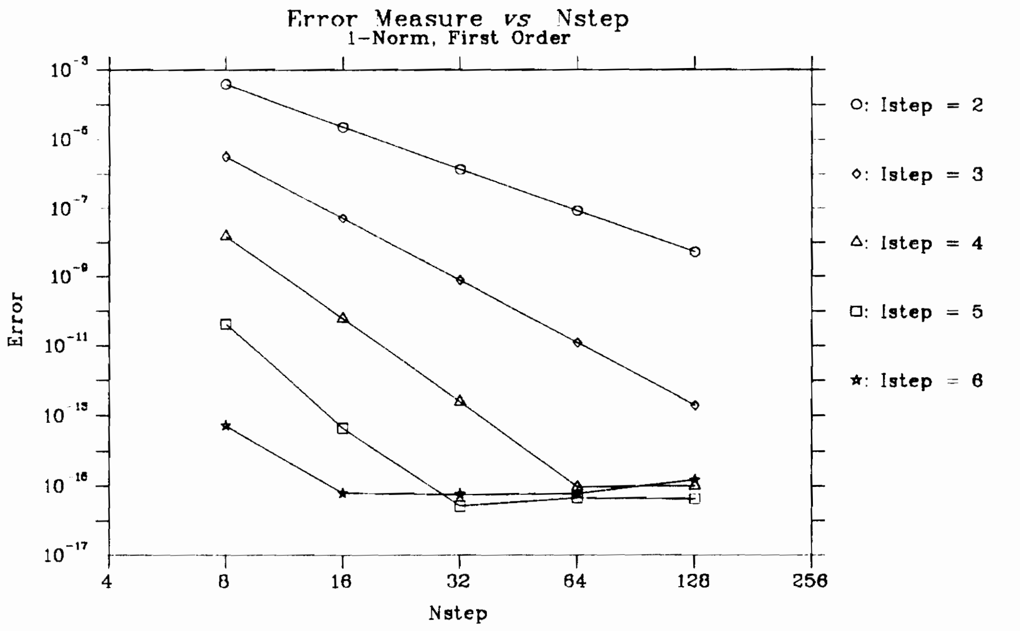

Figure F-8: 1-Norm of Error Measure *vs* **Nstep** (First Order).

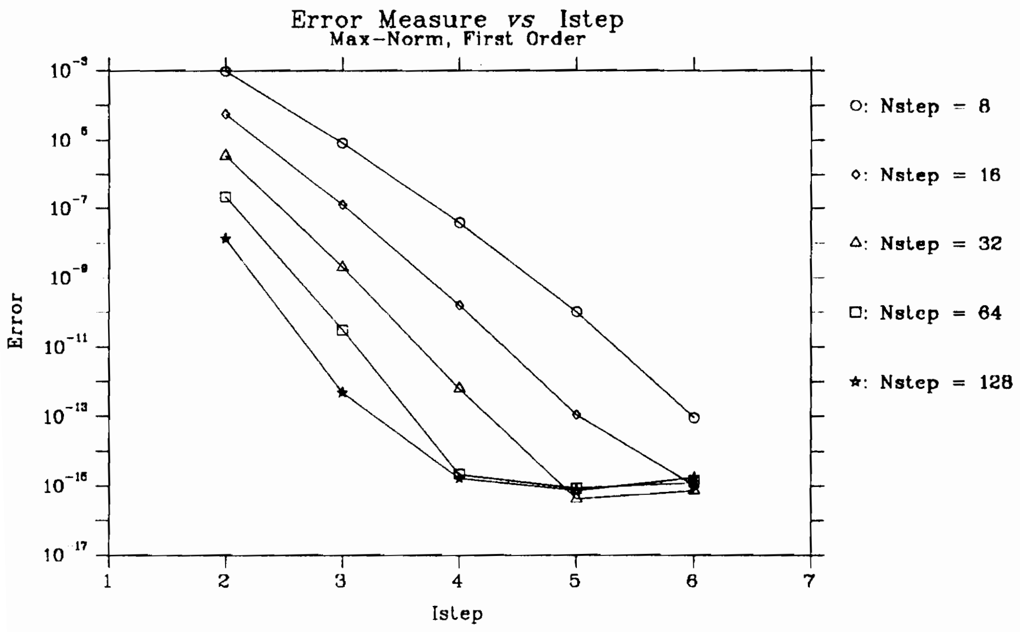

Figure F-9: Max-Norm of Error Measure *vs* **Nstep** (First Order).

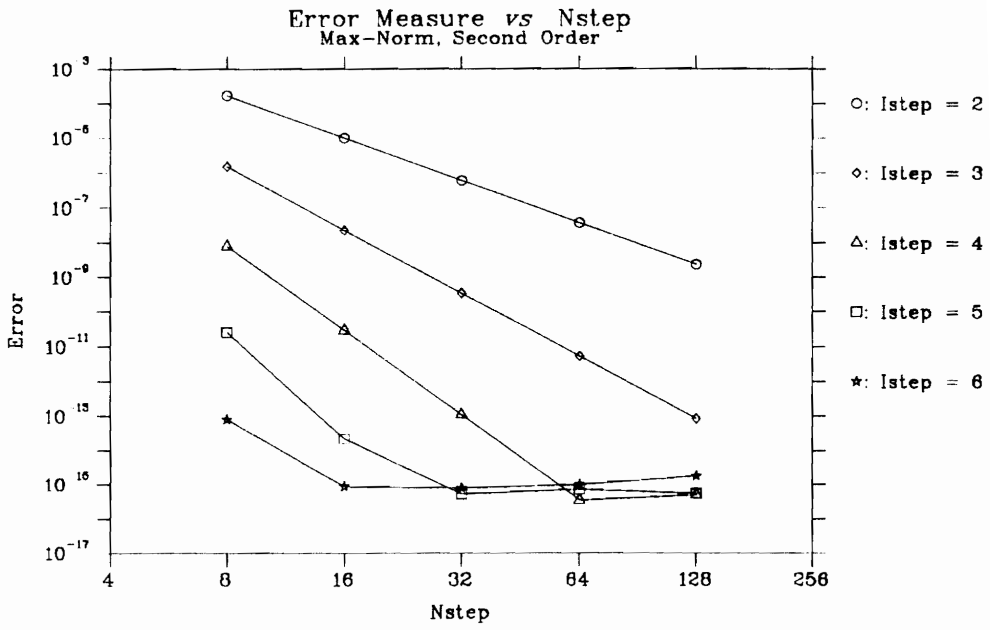

Figure F-10: 1-Norm of Error Measure *vs* **Nstep** (Second Order).

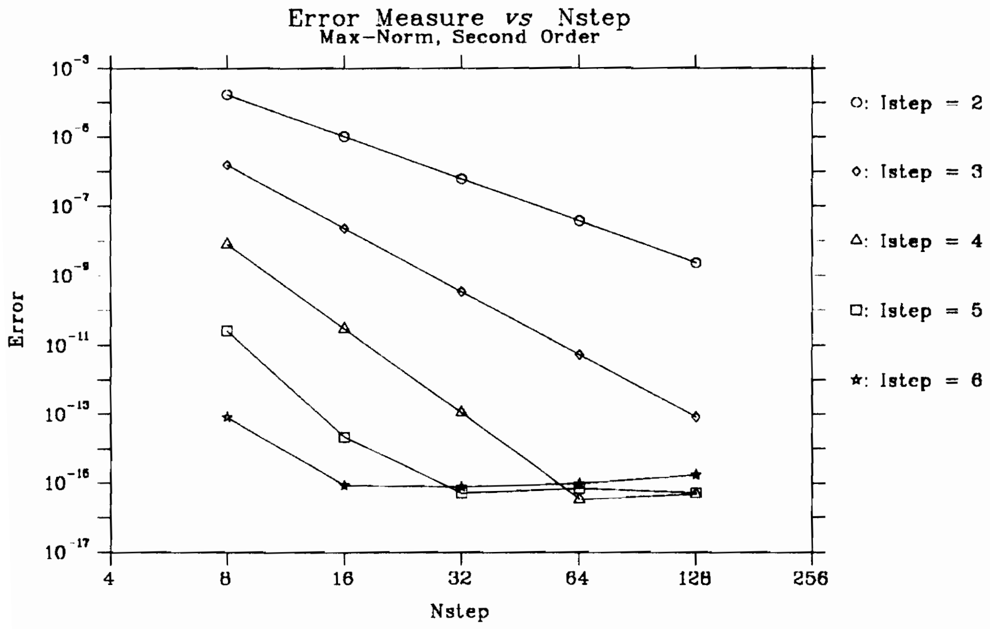

Figure F-11: Max-Norm of Error Measure *vs* **Nstep** (Second Order).

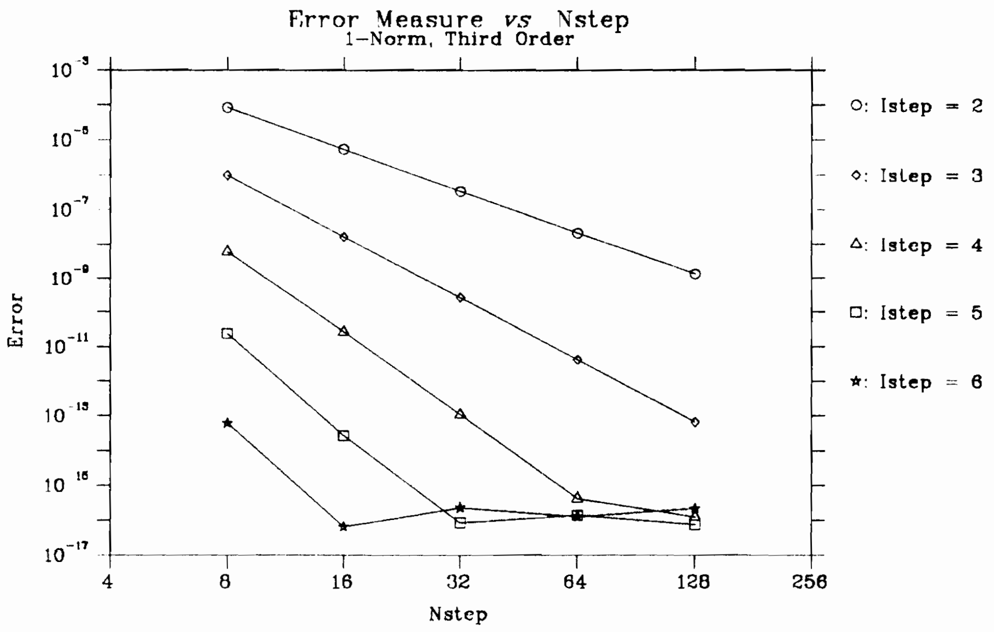

Figure F-12: 1-Norm of Error Measure *vs* **Nstep** (Third Order).

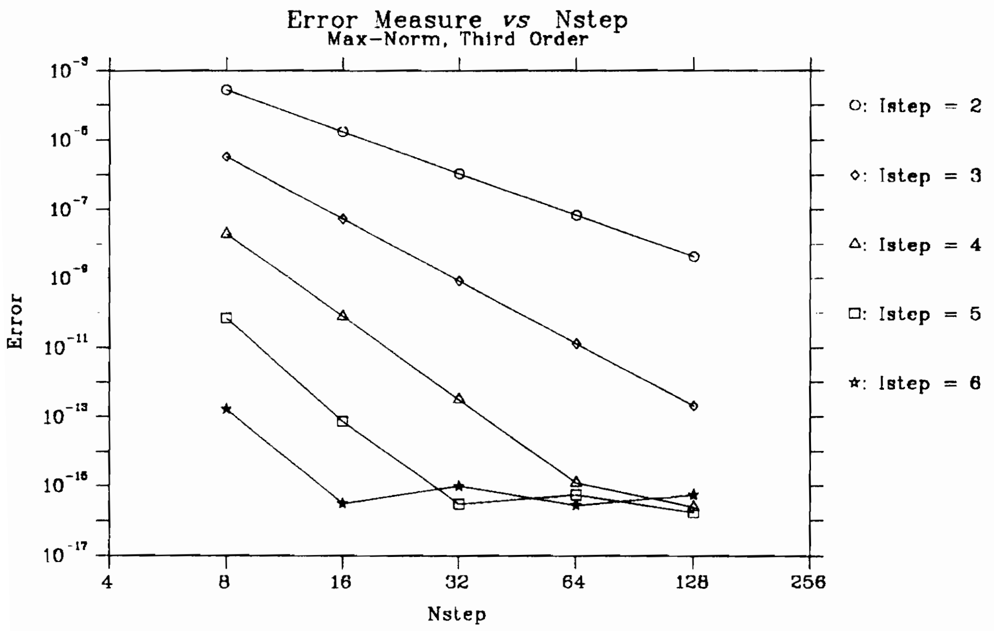

Figure F-13: Max-Norm of Error Measure *vs* **Nstep** (Third Order).

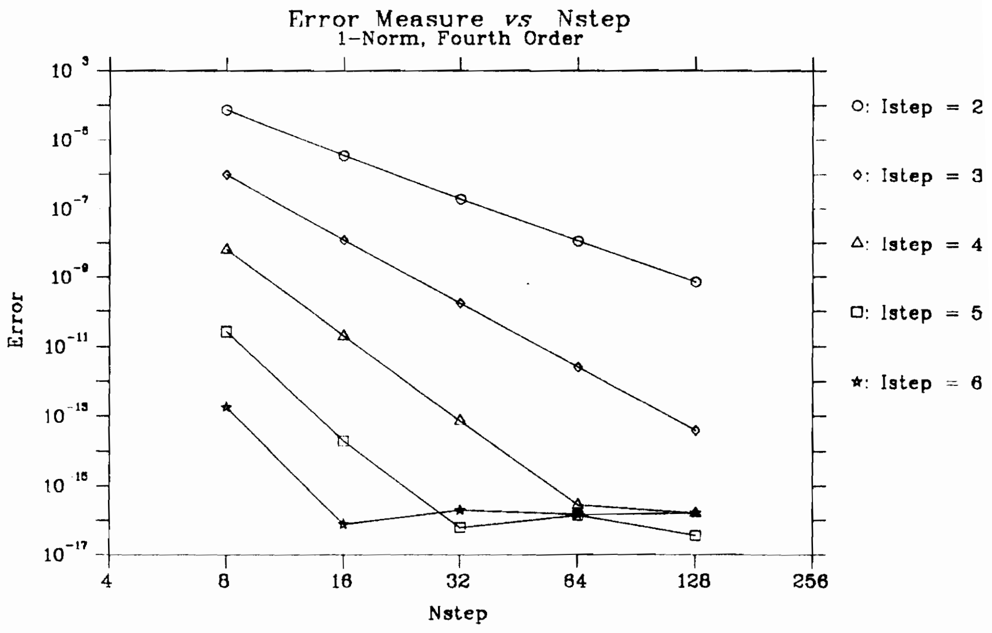

Figure F-14: 1-Norm of Error Measure *vs* **Nstep** (Fourth Order).

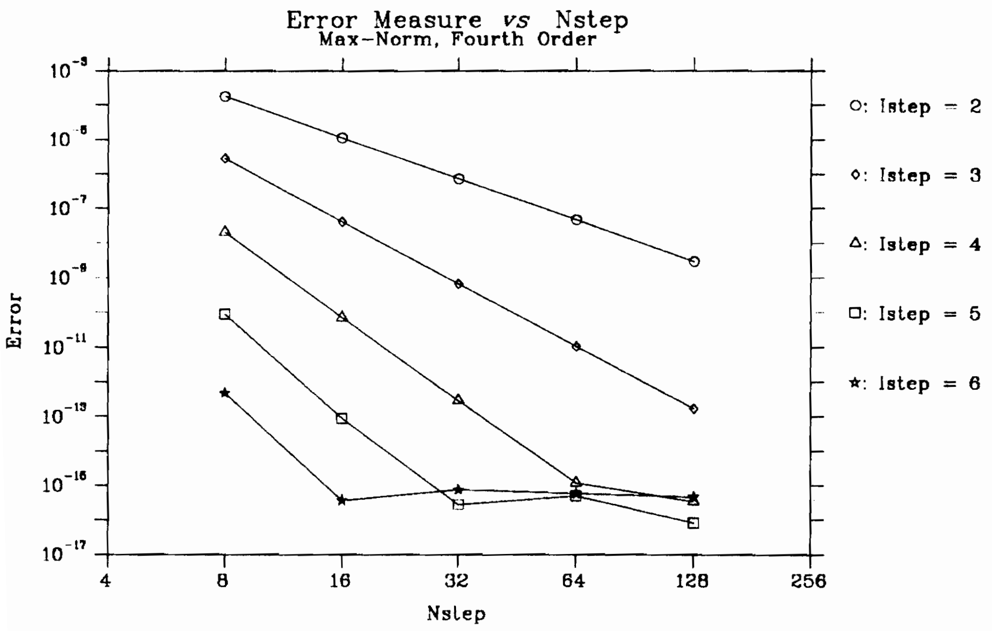

Figure F-15: Max-Norm of Error Measure *vs* **Nstep** (Fourth Order).

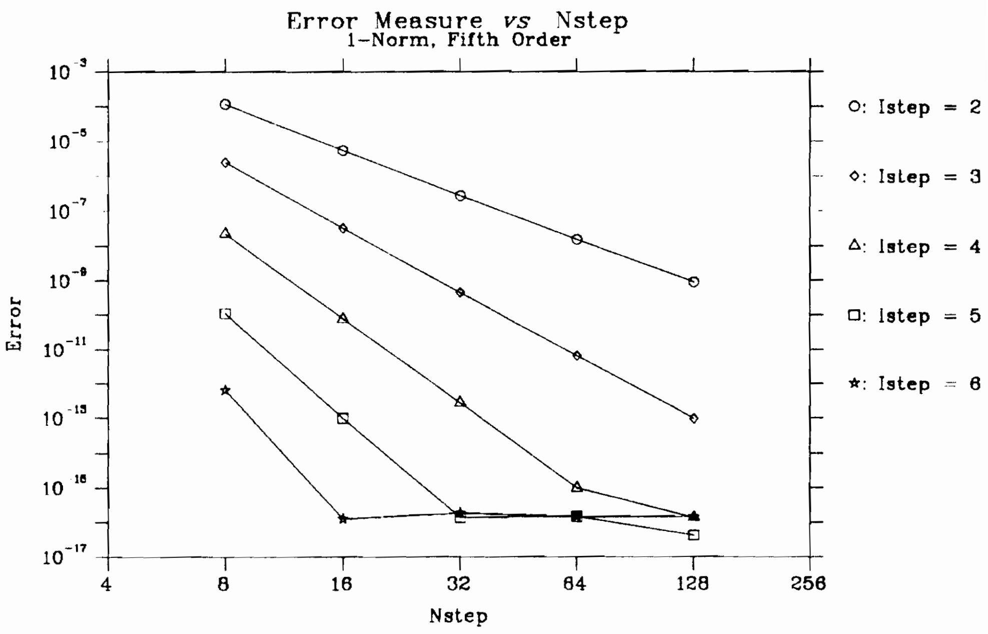

Figure F-16: 1-Norm of Error Measure *vs* **Nstep** (Fifth Order).

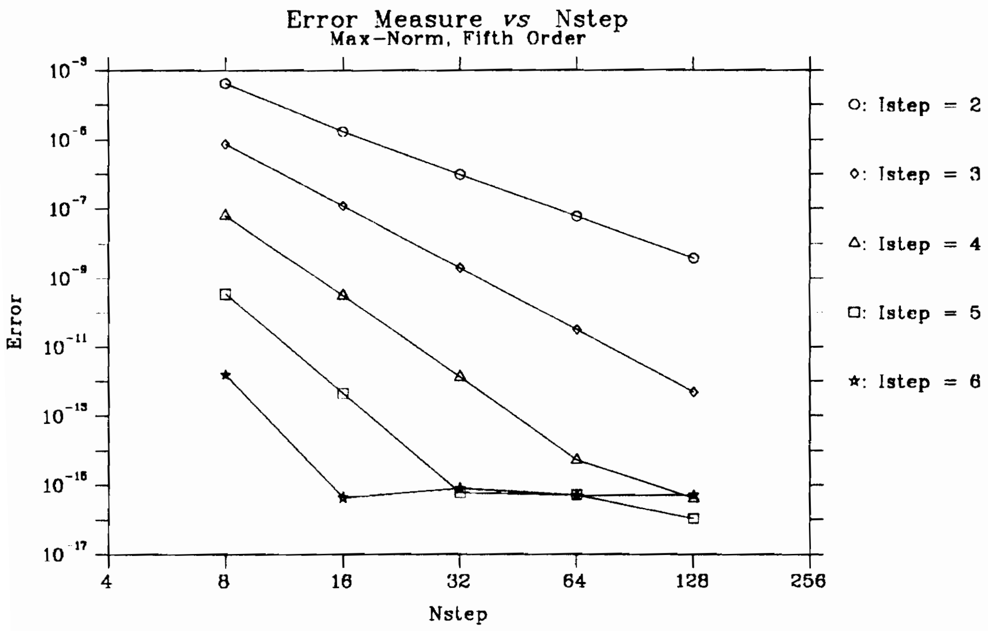

Figure F-17: Max-Norm of Error Measure *vs* **Nstep** (Fifth Order).

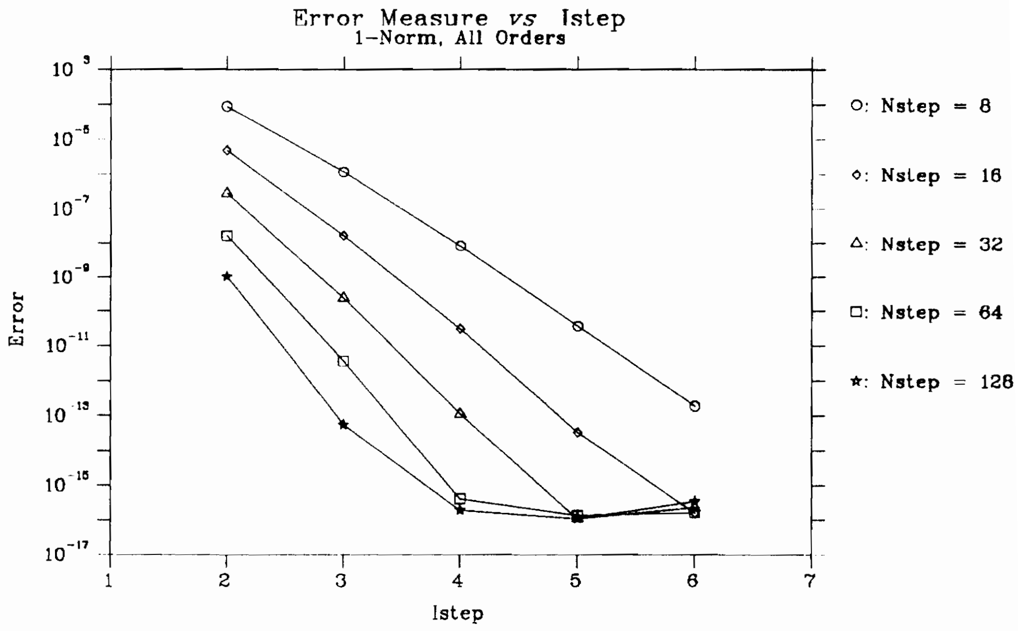

Figure F-18: 1-Norm of Error Measure *vs* **Istep** (All Orders).

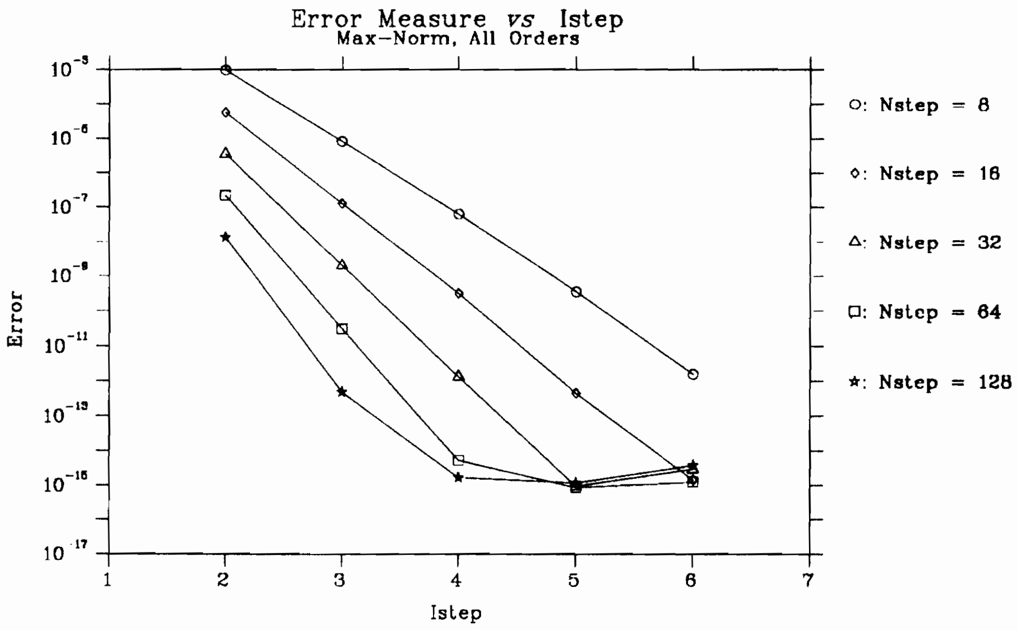

Figure F-19: Max-Norm of Error Measure *vs* **Istep** (All Orders).

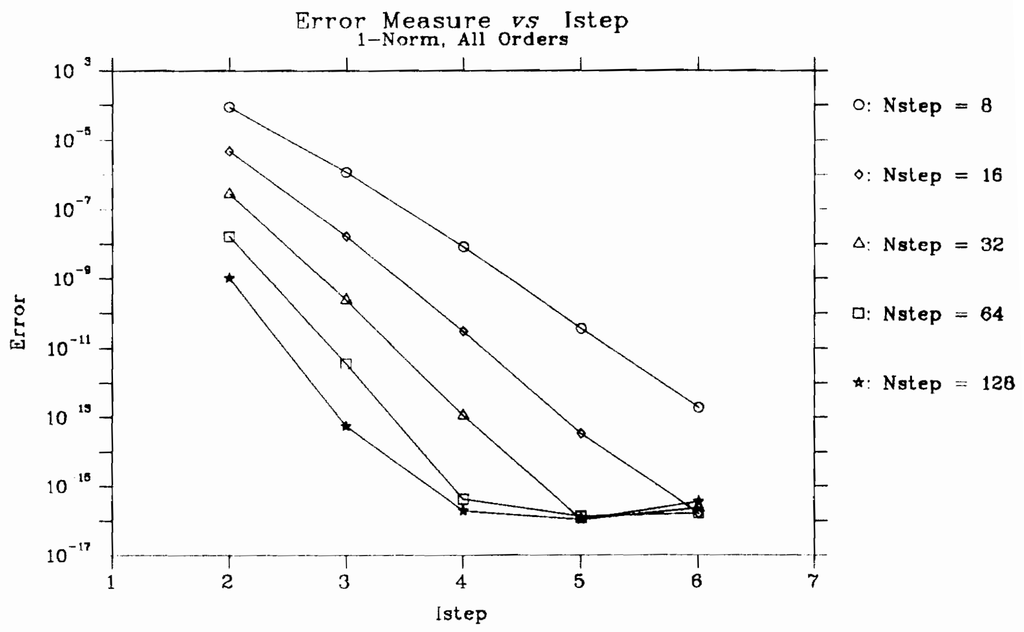

Figure F-20: 1-Norm of Error Measure *vs* **Istep** (ALL Orders).

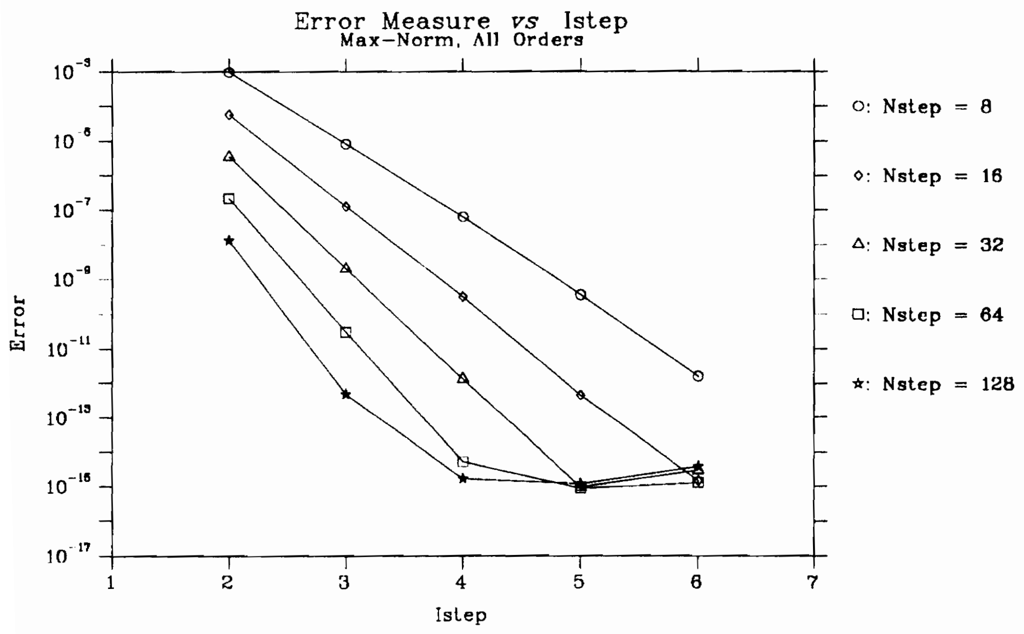

Figure F-21: Max-Norm of Error Measure *vs* **Istep** (All Orders).

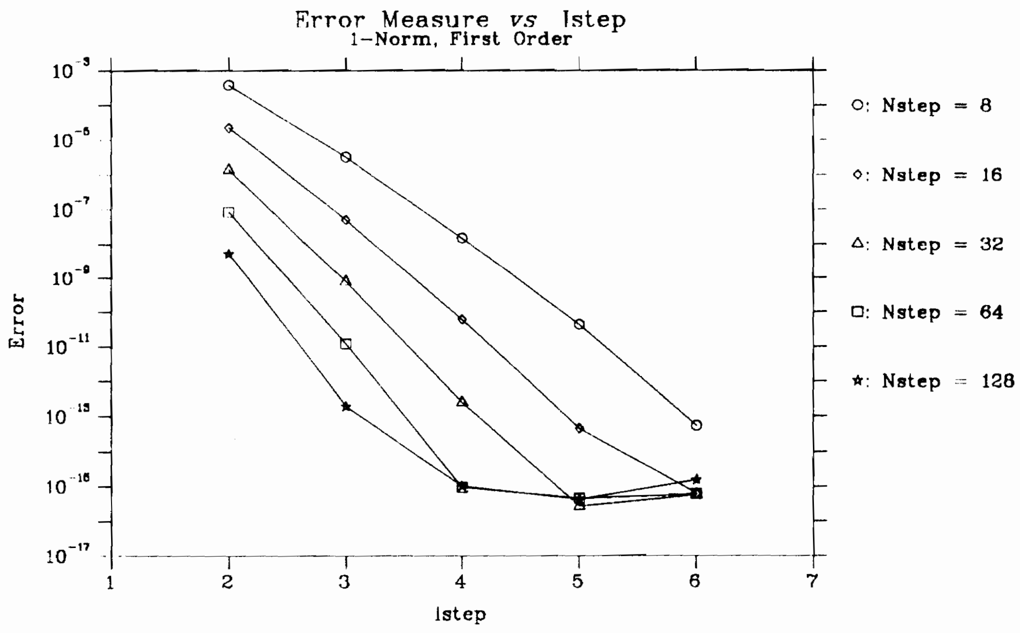

Figure F-22: 1-Norm of Error Measure *vs* **Istep** (First Order).

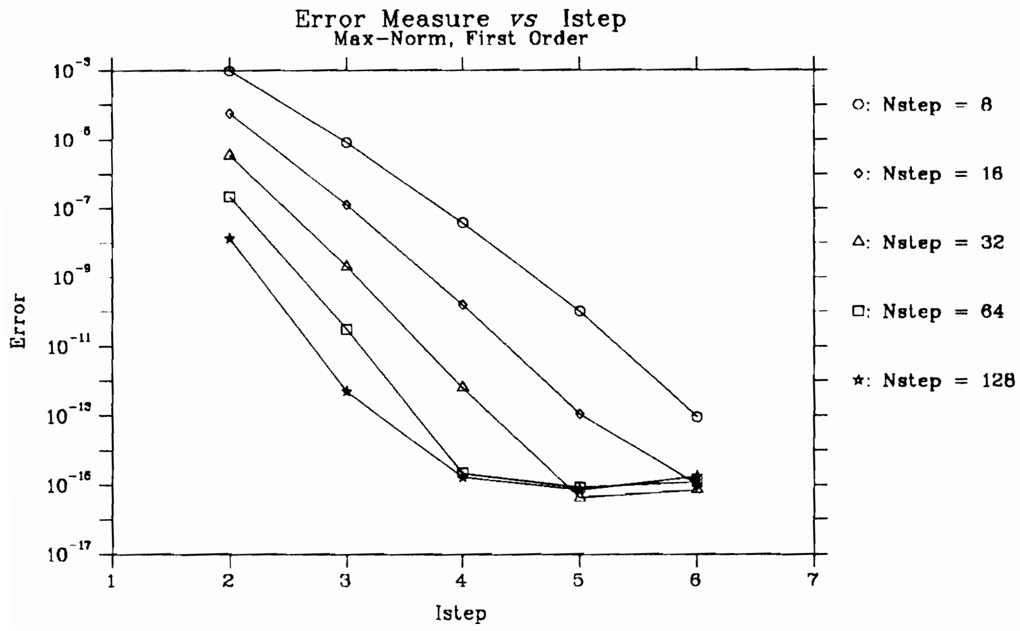

Figure F-23: Max-Norm of Error Measure *vs* **Istep** (First Order).

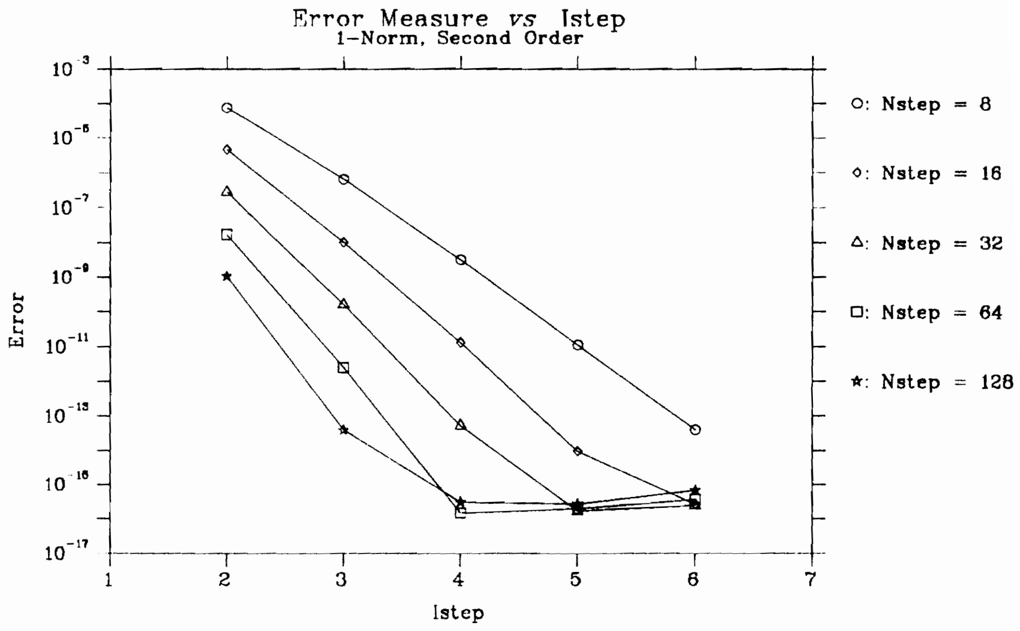

Figure F-24: 1-Norm of Error Measure *vs* **Istep** (Second Order).

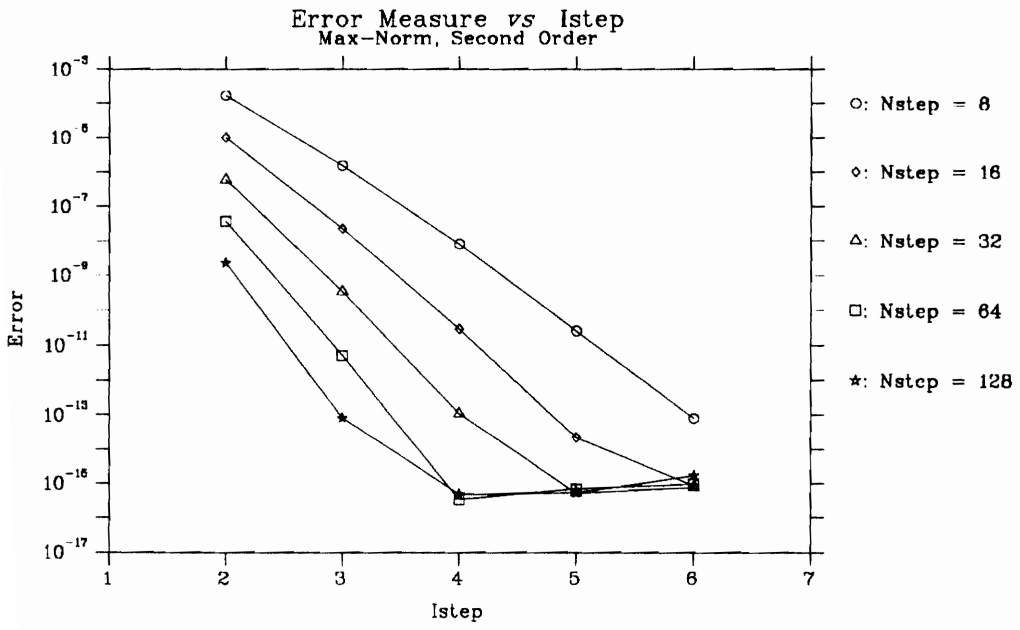

Figure F-25: Max-Norm of Error Measure *vs* **Istep** (Second Order).

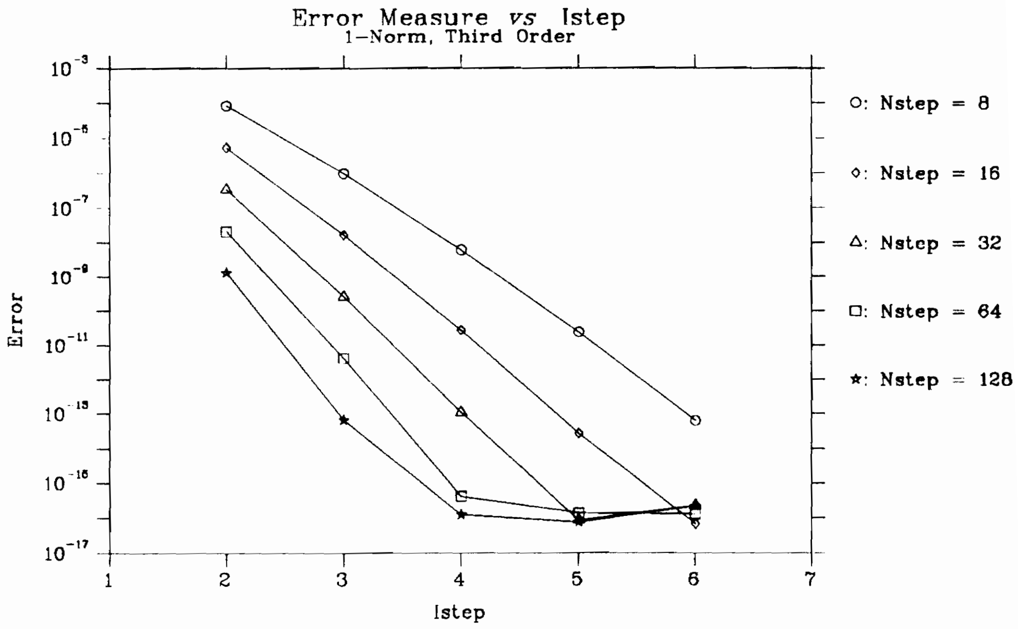

Figure F-26: 1-Norm of Error Measure *vs* **Istep** (Third Order).

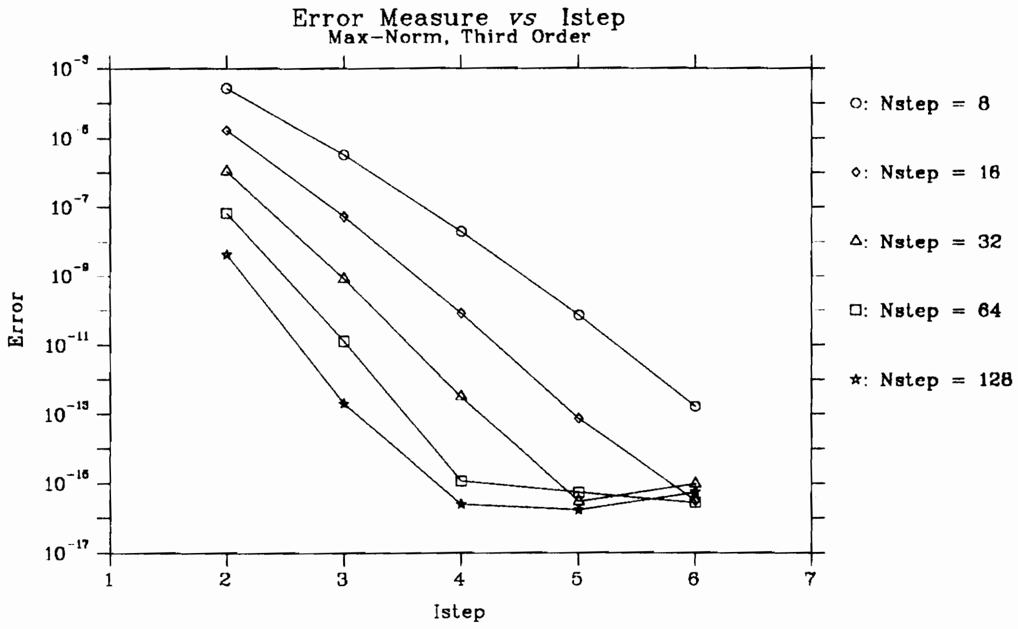

Figure F-27: Max-Norm of Error Measure *vs* **Istep** (Third Order).

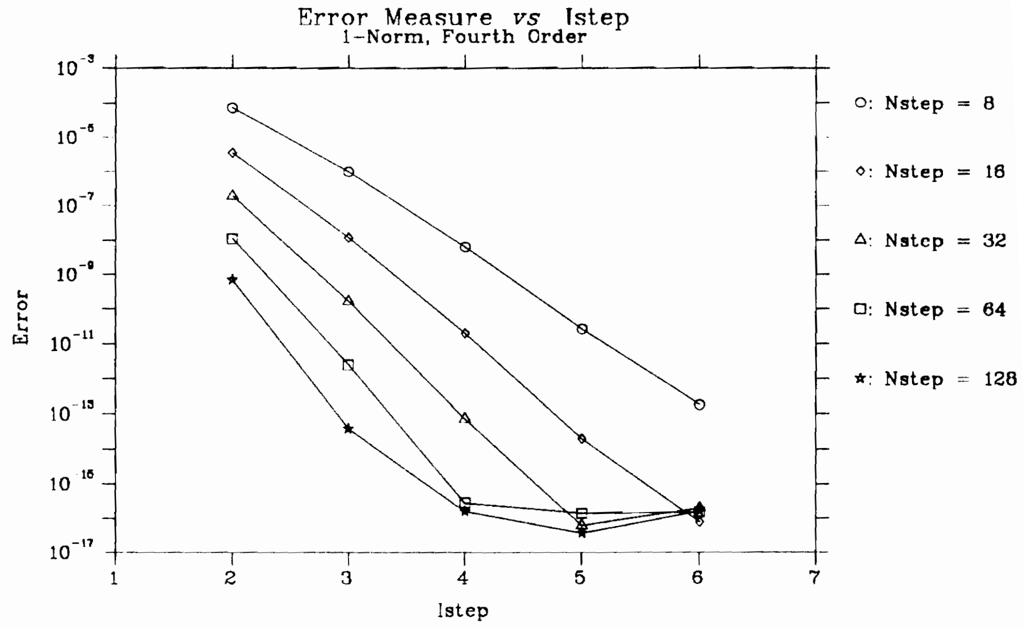

Figure F-28: 1-Norm of Error Measure *vs* **Istep** (Fourth Order).

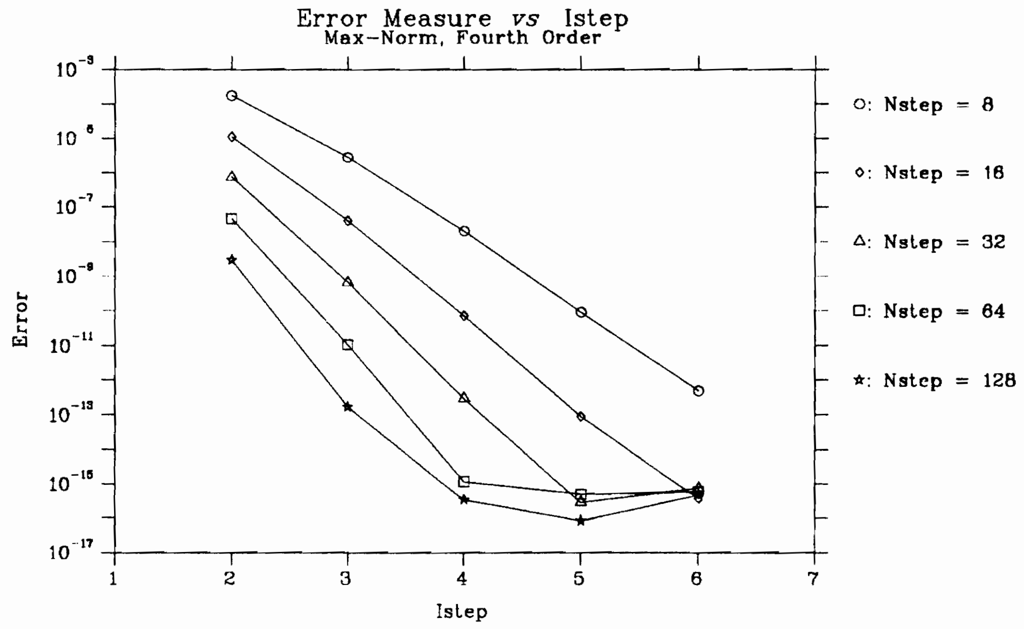

Figure F-29: Max-Norm of Error Measure *vs* **Istep** (Fourth Order).

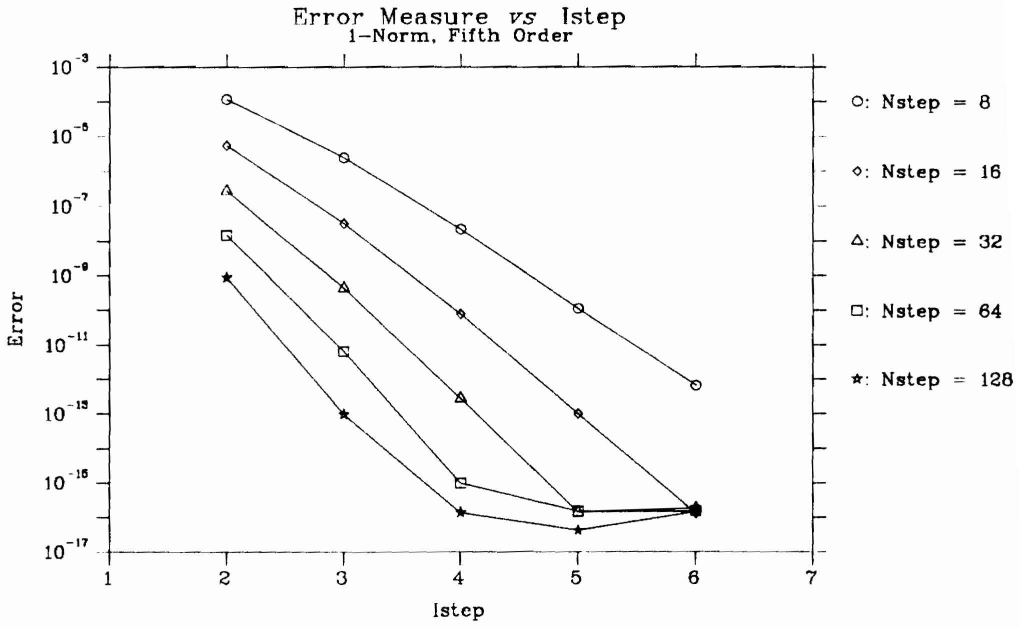

Figure F-30: 1-Norm of Error Measure *vs* **Istep** (Fifth Order).

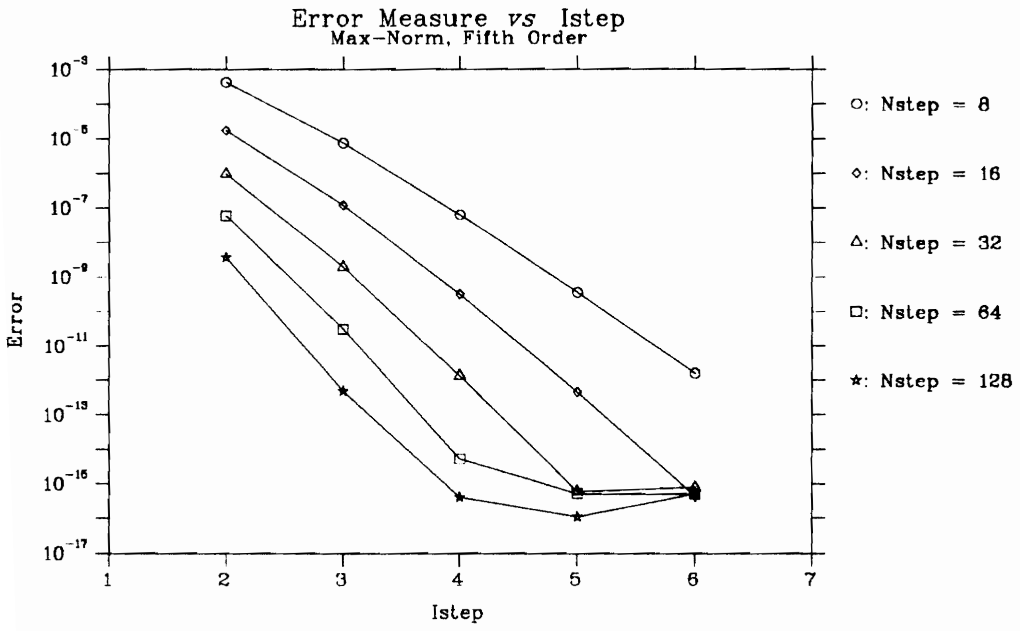

Figure F-31: Max-Norm of Error Measure *vs* **Istep** (Fifth Order).

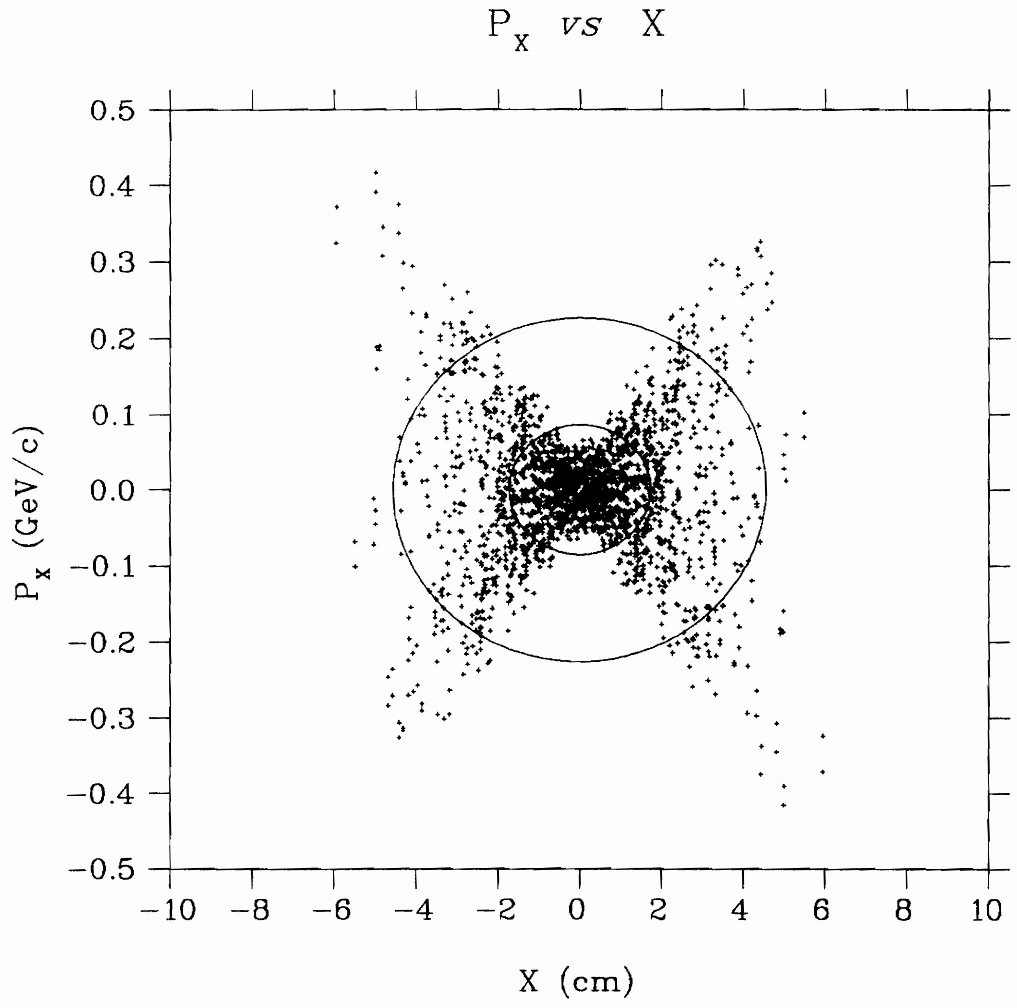

Figure F-32: Scatter-plot of x *vs* p_x for optimized lithium lens. Contours show 50% and 90% beam ellipses.

Appendix G

FORTRAN Code

G.1 Master Modules

G.1.1 HJDAdrift

```
      PROGRAM      HJDAdrift
      PARAMETER ( no=12, nd=2)
      EXTERNAL     drift
*
C=========================================================================
*
      CALL daini( no, (nd*2), (-10) )
      CALL HJDAdrive( drift )

      STOP
      END
*
```

```
C**********************************************************************
*
      SUBROUTINE HJDAdrive( hmltn )
      IMPLICIT NONE

      INTEGER             no, nd, nv, nstep, istep
      PARAMETER          ( no=12, nd=2, nv=nd*2, nstep=20, istep=4 )
      DOUBLE PRECISION    zero, one, two, four
      PARAMETER          ( zero=0.0d0, one=1.0d0, two=2.0d0, four=4.0d0)

      INTEGER            i, j

      DOUBLE PRECISION   z, gamma0, beta0, bgma0

      DOUBLE PRECISION   z1, x1(nv), r1(nv,nv)
      DOUBLE PRECISION   z2, x2(nv), r2(nv,nv)

      DOUBLE PRECISION   x0, ri(nv,nv), temp(nv,nv)

      DOUBLE PRECISION   f(nv), a(nv,nv)

      EXTERNAL           hmltn

      INTEGER            lx1, lx2, lx3, lx4, lx5, lx6, lx7, lx8
      INTEGER            isa1a, isa2a, isa3a
*DAINT(no,nv)            DS0, DS12
*DAINT(no,nv)            Dx(nv), Dy, Dz(nv)
*DAINT(no,nv)            Dx1(nv),   Dx2(nv)
*DAINT(no,nv)            Dtc1, Dxc1, Dpt1, Dpx1
*DAINT(no,nv)            Dtc2, Dxc2, Dpt2, Dpx2
*DAINT(no,nv)            Ttc2, Txc2, Tpt2, Tpx2
*DAINT(no,nv)            Dh, Dpz, Tst(nv), GS, DStheo, Diff
*
C======================================================================
*
      CALL daeps(1.0d-30)

        z1     = ZERO
        z2     = ONE
        z      = z2 - z1

        gamma0 = TWO
        beta0  = sqrt( ONE - ONE/gamma0/gamma0 )
        bgma0  = sqrt( gamma0*gamma0 - ONE )
*=====================================================================

       CALL HJDXident( no,nv,nd,nv, Dz, x1,r1,Ds0)

        x1(1) =   ZERO
        x1(2) =   ZERO
        x1(3) = - gamma0
        x1(4) =   ZERO

      CALL HJDAbsint( z1, x1,r1,DS0,  z2, x2,r2,DS12, Dz,Dx,
     &                     no,nv,nd,nv,   nstep,istep,  hmltn)

*=====================================================================

      CALL HJDAderiv( z2, x2,r2,DS12, f,a,Gs, no,nv,nd,nv, Dz,Dx2, hmltn)

*DA   DStheo = DS0 + z*Gs
*DA   Diff   = DS12 - DStheo
```

```
*=====================================================================
      CALL danot(no-1)

      CALL HJDX1( x1,r1,DS12, no,nv,nd,nv, Dz, Dx1)
      CALL HJDX2( x2,r2,DS12, no,nv,nd,nv, Dz, Dx2)

*DA   Dtc1 = Dx1(1)
*DA   Dxc1 = Dx1(2)
*DA   Dpt1 = Dx1(3)
*DA   Dpx1 = Dx1(4)

*DA   Dtc2 = Dx2(1)
*DA   Dxc2 = Dx2(2)
*DA   Dpt2 = Dx2(3)
*DA   Dpx2 = Dx2(4)

      CALL drift( z2, Dx2, no,nv,nd, Dh)

*DA   Dpz    = zero - Dh
      CALL dapri( Dpz, 10 )


*DA   Ttc2 = Dtc1 - Dpt1/Dpz
*DA   Txc2 = Dxc1 + Dpx1/Dpz
*DA   Tpt2 = Dpt1
*DA   Tpx2 = Dpx1

*
*=====================================================================
*
        WRITE( 10, '(1h1)')
        WRITE( 10,  '(a)' ) '                x1;   r1'
      DO i=1,nv
        WRITE( 10, 2001) x1(i), (r1(i,j), j=1,nv)
      END DO
        WRITE( 10, *)

        WRITE( 10,  '(a)' ) '                x2;   r2'
      DO i=1,nv
        WRITE( 10, 2001) x2(i), (r2(i,j), j=1,nv)
      END DO
        WRITE( 10, *)

      CALL sympinv( r2,nv, ri,nv, nd)
      CALL   mmmul( r2,nv, ri,nv, temp,nv, nv)

      DO i=1,nv
        temp(i,i) = temp(i,i) - ONE
      END DO

        WRITE( 10, '(a)')
     &        ' Symplecticity check:  r2 * ( J r2^T J^-1) - I'
      DO i=1,nv
        WRITE( 10, 2002) (temp(i,j), j=1,nv)
      END DO
      DO j=1,nv
      DO i=1,nv
        temp(i,j) = ZERO
      END DO
        temp(j,j) = ONE
      END DO

      temp(1,3) = z/bgma0/bgma0/bgma0
      temp(2,4) = z/beta0/gamma0

      WRITE( 10, '(1x)')
      DO i=1,nv
        WRITE( 10, 2002) ( temp(i,j), j=1,nv)
      END DO

      WRITE( 10, '(1x)')
      DO i=1,nv
        WRITE( 10, 2002) ( (R2(i,j)-temp(i,j)), j=1,nv)
      END DO
*
*
      WRITE( 10, '(1h1)')
      CALL mypri( DS12,  DStheo, 10)
*
*

      WRITE( 10, '(1h1)')
      CALL mypri( Dtc2, Ttc2, 10)
*
      WRITE( 10, '(1h1)')
      CALL mypri( Dxc2, Txc2, 10)
*
      WRITE( 10, '(1h1)')
      CALL mypri( Dpt2, Tpt2, 10)
*
      WRITE( 10, '(1h1)')
      CALL mypri( Dpx2, Tpx2, 10)

*
*
 2001 FORMAT(1x, 1pe15.8, ';  ', 1pe15.8, 4(', ', 1pe15.8))
 2002 FORMAT(1x, 15x,        3x, 1pe15.8, 4(', ', 1pe15.8))
*
      RETURN
      END
*
C**************************  End Of File  ******************************
```

G.1.2 HJDApolrsho

```
      PROGRAM      HJDApolrsho
      PARAMETER ( nd=2 )
      EXTERNAL     polrsho
*
C========================================================================
*
      WRITE(  *,       '(x, 11h      no = , $)')
      READ (  *,'(i10)')             no

      CALL daini( no, (nd*2), (-10) )
      CALL HJDAdrive( no, polrsho)

      STOP
      END
*

C************************************************************************
*
      SUBROUTINE HJDAdrive( no, hmltn)
      IMPLICIT NONE

      INTEGER             no, nd, nv, nstep, istep
      PARAMETER         ( nd=2, nv=nd*2)
      DOUBLE PRECISION  zero, one, two, four
      PARAMETER         ( zero=0.0d0, one=1.0d0, two=2.0d0, four=4.0d0)

      DOUBLE PRECISION  eps

      INTEGER           i, j

      DOUBLE PRECISION  t1, x1( nv), r1( nv, nv)
      DOUBLE PRECISION  t2, x2( nv), r2( nv, nv)

      DOUBLE PRECISION  x0, ri( nv, nv), temp( nv, nv)

      DOUBLE PRECISION  R_i, Theta_i, Pr_i, Ptheta_i
      DOUBLE PRECISION  ct, st

      DOUBLE PRECISION  pi, pi2, w
      COMMON            pi, pi2, w

      EXTERNAL          hmltn


      INTEGER           lx1, lx2, lx3, lx4, lx5, lx6, lx7, lx8, lx9
      INTEGER           isa1a, isa2a, isa3a

*DAINT(no,nv)           Ds0, Ds12
*DAINT(no,nv)           Dz(nv),     Dx(nv)
*DAINT(no,nv)           Dx1(nv),    Dx2(nv)
*DAINT(no,nv)           Dr1,  Dth1, Dpr1, Dpt1
*DAINT(no,nv)           Dr2,  Dth2, Dpr2, Dpt2
*DAINT(no,nv)           Dxc1, Dpx1, Dyc1, Dpy1
*DAINT(no,nv)           Dxc2, Dpx2, Dyc2, Dpy2
*DAINT(no,nv)           Txc2, Tpx2, Tyc2, Tpy2
*DAINT(no,nv)           Tst( nv)

*
*========================================================================
*======================================================================
*
        pi  = FOUR * atan( ONE)
        pi2 =  TWO * pi

        w   = pi2

        t1  = zero
        t2  = 2.0d0 / 3.0d0

      WRITE( 10, '(/a/)')
     &     ' HJDApolrsho: Harmonic Oscillator in Polar Coordinates '

        write( 10,      '(x, 11h      no = , i4)') no

        write(  *,      '(x, 11h   nstep = , $)')
        read  (  *,'(i10)')        nstep
        write( 10,      '(x, 11h   nstep = , i4)') nstep

        write(  *,      '(x, 11h   istep = , $)')
        read  (  *,'(i10)')        istep
        write( 10,      '(x, 11h   istep = , i4)') istep

        write(  *,      '(x, 11h     eps = , $)')
        read  (  *,'(1pe20.10)')     eps
        write( 10,      '(x, 11h     eps = , 1pe13.6)') eps
        CALL daeps( eps)

        write(  *,      '(x, 11h     R_i = , $)')
        read  (  *,'(1pe20.10)')     R_i
        write( 10,      '(x, 11h     R_i = , 1pe13.6)') R_i

        write(  *,      '(x, 11h    Pr_i = , $)')
        read  (  *,'(1pe20.10)')    Pr_i
        write( 10,      '(x, 11h    Pr_i = , 1pe13.6)') Pr_i

        write(  *,      '(x, 11hPtheta_i = , $)')
        read  (  *,'(1pe20.10)') Ptheta_i
        write( 10,      '(x, 11hPtheta_i = , 1pe13.6)') Ptheta_i

        write( 10,      '(x, 11h      t1 = , 1pe13.6)') t1
        write( 10,      '(x, 11h      t2 = , 1pe13.6)') t2

*

        Theta_i  = ZERO
        Pr_i     = Pr_i     * w * R_i
        Ptheta_i = Ptheta_i * w * R_i * R_i

      CALL HJDXident( no,nv,nd,nv, Dz, x1,r1,Ds0)

        x1(1) = R_i
        x1(2) = Theta_i
        x1(3) = Pr_i
        x1(4) = Ptheta_i

*
*======================================================================
*
      CALL HJDAbsint( t1, x1,r1,Ds0,  t2, x2,r2,Ds12,
     &                Dz,Dx,  no,nv,nd,nd,nd*2, nstep,istep,  hmltn)
*
*======================================================================
*=====================================================================
*
      CALL danot(no-1)

      CALL HJDX1( x1,r1,DS12, no,nv,nd,nv, Dz, Dx1)
      CALL HJDX2( x2,r2,DS12, no,nv,nd,nv, Dz, Dx2)
*
*=====================================================================
*
      st   = sin(w*(t2-t1))
      ct   = cos(w*(t2-t1))

*DA   Dr1  = Dx1(1)
*DA   Dth1 = Dx1(2)
*DA   Dpr1 = Dx1(3)
*DA   Dpt1 = Dx1(4)

*DA   Dxc1 =  Dr1  * cos( Dth1)
*DA   Dyc1 =  Dr1  * sin( Dth1)
*DA   Dpx1 =  Dpr1 * cos( Dth1) - Dpt1 * sin( Dth1) / Dr1
*DA   Dpy1 =  Dpr1 * sin( Dth1) + Dpt1 * cos( Dth1) / Dr1

*DA   Dr2  = Dx2(1)
*DA   Dth2 = Dx2(2)
*DA   Dpr2 = Dx2(3)
*DA   Dpt2 = Dx2(4)

*DA   Dxc2 =  Dr2  * cos( Dth2)
*DA   Dyc2 =  Dr2  * sin( Dth2)
*DA   Dpx2 =  Dpr2 * cos( Dth2) - Dpt2 * sin( Dth2) / Dr2
*DA   Dpy2 =  Dpr2 * sin( Dth2) + Dpt2 * cos( Dth2) / Dr2

*DA   Txc2 =  Dxc1*ct + Dpx1*st/w
*DA   Tyc2 =  Dyc1*ct + Dpy1*st/w
*DA   Tpx2 =  Dpx1*ct - Dxc1*st*w
*DA   Tpy2 =  Dpy1*ct - Dyc1*st*w

*DA   tst(1) =   Txc2  -  Dxc2
*DA   tst(2) =   Tyc2  -  Dyc2
*DA   tst(3) = ( Tpx2  -  Dpx2 ) / w
*DA   tst(4) = ( Tpy2  -  Dpy2 ) / w

*
*=====================================================================
*

      WRITE( 10, '(1h1)')

      WRITE( 10, '(a)') '            x1;   r1'
      DO i=1,nv
        WRITE( 10, 2001) x1(i), (r1(i,j), j=1,nv)
      END DO
      WRITE( 10, *)
*
      WRITE( 10, '(a)') '            x2;   r2'
      DO i=1,nv
        WRITE( 10, 2001) x2(i), (r2(i,j), j=1,nv)
      END DO
      WRITE( 10, *)
*
```

```
*
      CALL sympinv( r2,nv, ri,nv, nd)
      CALL   mmmul( r2,nv, ri,nv, temp,nv, nv)
        WRITE( 10, '(a)') ' Symplecticity check:  r2 * ( J r2^T J^-1) '
      DO i=1,nv
        WRITE( 10, 2002) (temp(i,j), j=1,nv)
      END DO
*
      WRITE( 10, '(1h1)')
      CALL dapri( Ds0, 10)
      WRITE( 10, *)
      DO i=1,nv
        CALL dapri( Dx1(i), 10)
        WRITE( 10, *)
      END DO
*
      WRITE( 10, '(1h1)')
      CALL dapri( Ds12, 10)
      WRITE( 10, *)
      DO i=1,nv
        CALL dapri( Dx2(i), 10)
        WRITE( 10, *)
      END DO
*
      WRITE( 10, '(1h1)')
      DO i=1,nd*2
        CALL dapri( tst(i), 10)
      END DO
*
      WRITE( 10, '(1h1)')
      CALL dapri( Txc2, 10)
      CALL dapri( Dxc2, 10)
      WRITE( 10, '(1h1)')
      CALL dapri( Tyc2, 10)
      CALL dapri( Dyc2, 10)
      WRITE( 10, '(1h1)')
      CALL dapri( Tpx2, 10)
      CALL dapri( Dpx2, 10)
      WRITE( 10, '(1h1)')
      CALL dapri( Tpy2, 10)
      CALL dapri( Dpy2, 10)
*
```

```
*
      WRITE( 10, '(1h1)')
      CALL mypri( Txc2, Dxc2, 10)
      CALL mypri( Tyc2, Dyc2, 10)
      CALL mypri( Tpx2, Dpx2, 10)
      CALL mypri( Tpy2, Dpy2, 10)
*
 2001 FORMAT(1x, 1pe13.6, ';    ', 1pe13.6, 3(', ', 1pe13.6))
 2002 FORMAT(1x, 13x,     '     ', 1pe13.6, 3(', ', 1pe13.6))
*
      RETURN
      END
*
C***************************  End Of File  ******************************
```

G.1.3 HJDAubfield

```
      PROGRAM     HJDAubfield
      PARAMETER ( nd=1 )
      EXTERNAL    ubfield
*
C==========================================================================
*
      WRITE(  *,       '(x, 11h       no = , $)')
      READ (  *,'(i10)')              no

      CALL daini( no, (nd*2), (-10) )
      CALL HJDAdrive( no, ubfield)

      STOP
      END
*

C**************************************************************************
*
      SUBROUTINE HJDAdrive( no, hmltn)
      IMPLICIT NONE

      INTEGER              no, nd, nv, nstep, istep
      PARAMETER          ( nd=1, nv=nd*2)
      DOUBLE PRECISION     zero, one, two, four
      PARAMETER          ( zero=0.0d0, one=1.0d0, two=2.0d0, four=4.0d0)

      DOUBLE PRECISION     eps

      INTEGER              i, j

      DOUBLE PRECISION     t1, x1( nv), r1( nv, nv)
      DOUBLE PRECISION     t2, x2( nv), r2( nv, nv)

      DOUBLE PRECISION     x0, ri( nv, nv), temp( nv, nv)

      DOUBLE PRECISION     R_i, Pr_i, Theta, Pratio
      DOUBLE PRECISION     Cth, Sth

      DOUBLE PRECISION     Enorm,          Emax
      DOUBLE PRECISION     Qnorm((-1):10), Qmax((-1):10)
      DOUBLE PRECISION     Tnorm((-1):10), Tmax((-1):10)

      DOUBLE PRECISION     pi, pi2
      COMMON               pi, pi2

      DOUBLE PRECISION     beta0, gamma0, btgma0, const,
     &                     beta,  gamma,  btgma,  bgbg

      COMMON /ubfield/     beta0, gamma0, btgma0, const,
     &                     beta,  gamma,  btgma,  bgbg

      SAVE   /ubfield/

      EXTERNAL             hmltn


      INTEGER              lx1, lx2, lx3, lx4, lx5, lx6, lx7, lx8, lx9
      INTEGER              isa1a, isa2a, isa3a
```

```
*DAINT(no,nv)           Ds0, Ds12
*DAINT(no,nv)           Dz(nv),     Dx(nv)
*DAINT(no,nv)           Dx1(nv),    Dx2(nv)
*DAINT(no,nv)           Dr1, Dpr1,  Dr2, Dpr2
*DAINT(no,nv)           Ca1, Sa1,   Ca2, Sa2
*DAINT(no,nv)           Xc1, Yc1,   Xc2, Yc2,   Xc2p, Yc2p
*DAINT(no,nv)           Tst(nv)
*
*=========================================================================
*
       pi    = FOUR * atan( ONE)
       pi2   =  TWO * pi

       write( 10, '(/a/)')
    &         ' HJDAubfield: Particle in Uniform Magnetic Field '

       write( 10,       '(x, 11h      no = , i4)')    no

       write(  *,       '(x, 11h   nstep = , $)')
       read (  *,'(i10)')          nstep
       write( 10,       '(x, 11h   nstep = , i4)') nstep

       write(  *,       '(x, 11h   istep = , $)')
       read (  *,'(i10)')          istep
       write( 10,       '(x, 11h   istep = , i4)') istep

       write(  *,       '(x, 11h     eps = , $)')
       read (  *,'(1pe20.10)')       eps
       write( 10,       '(x, 11h     eps = , 1pe13.6)') eps
       write( 10, *)
       CALL daeps( eps)

       write(  *,       '(x, 11h     R_i = , $)')
       read (  *,'(1pe20.10)')       R_i
       write( 10,       '(x, 11h     R_i = , 1pe13.6)') R_i

       write(  *,       '(x, 11h    Pr_i = , $)')
       read (  *,'(1pe20.10)')      Pr_i
       write( 10,       '(x, 11h    Pr_i = , 1pe13.6)') Pr_i
       write( 10, *)

       write(  *,       '(x, 11h  btgma0 = , $)')
       read (  *,'(1pe20.10)')    btgma0
       write( 10,       '(x, 11h  btgma0 = , 1pe13.6)') btgma0

       write(  *,       '(x, 11h  btgma  = , $)')
       read (  *,'(1pe20.10)')    btgma
       write( 10,       '(x, 11h  btgma  = , 1pe13.6)') btgma
       write( 10, *)

       write(  *,       '(x, 11hTheta/Pi = , $)')
       read (  *,'(1pe20.10)')     Theta
       write( 10,       '(x, 11hTheta/Pi = , 1pe13.6)') Theta

*
```

```
*
        Theta  = Theta * pi
        const  = btgma0 / TWO
        bgbg   = btgma * btgma
        gamma0 = sqrt( ONE + btgma0 * btgma0 )
        gamma  = sqrt( ONE + btgma  * btgma  )
        beta0  = sqrt( ONE - ONE / ( gamma0 * gamma0 ) )
        beta   = sqrt( ONE - ONE / ( gamma  * gamma  ) )
        Pratio = btgma / btgma0
*
      CALL HJDXident( no,nv,nd,nv, Dz, x1,r1,Ds0)
        t1    = ZERO
        t2    = Theta
        x1(1) = R_i  * Pratio
        x1(2) = Pr_i * Pratio
*
*======================================================================
*
      CALL HJDAbsint( t1, x1,r1,Ds0,  t2, x2,r2,Ds12,
     &                Dz,Dx,  no,nv,nd,nd*2, nstep,istep,  hmltn)
*
*======================================================================
*
      CALL danot(no-1)
      CALL HJDX1( x1,r1,DS12, no,nv,nd,nv, Dz, Dx1)
      CALL HJDX2( x2,r2,DS12, no,nv,nd,nv, Dz, Dx2)
*
*======================================================================
*
      Sth   = sin( Theta )
      Cth   = cos( Theta )
*DA   Dr1   = Dx1(1)
*DA   Dpr1  = Dx1(2)
*DA   Sa1   = Dpr1 / btgma
*DA   Ca1   = sqrt( ONE - Sa1*Sa1 )
*DA   Dr2   = Dx2(1)
*DA   Dpr2  = Dx2(2)
*DA   Sa2   = Dpr2 / btgma
*DA   Ca2   = sqrt( ONE - Sa2*Sa2 )
*DA   Xc1   = Pratio * Sa1
*DA   Yc1   = Dr1 - Pratio * Ca1
*DA   Xc2p  = Pratio * Sa2
*DA   Yc2p  = Dr2 - Pratio * Ca2
*DA   Xc2   = Yc2p * Sth   +  Xc2p * Cth
*DA   Yc2   = Yc2p * Cth   -  Xc2p * Sth
*DA   tst(1) =   Xc2   -   Xc1
*DA   tst(2) =   Yc2   -   Yc1
*
*======================================================================
*======================================================================
*
      WRITE( 10, '(1h1)')

      WRITE( 10, '(a)') '            x1;    r1'
      DO i=1,nv
        WRITE( 10, 2001) x1(i), (r1(i,j), j=1,nv)
      END DO
      WRITE( 10, *)
*
      WRITE( 10, '(a)') '            x2;    r2'
      DO i=1,nv
        WRITE( 10, 2001) x2(i), (r2(i,j), j=1,nv)
      END DO
      WRITE( 10, *)

*

      CALL sympinv( r2,nv, ri,nv, nd)
      CALL   mmmul( r2,nv, ri,nv, temp,nv, nv)

      WRITE( 10, '(a)')
     &           ' Symplecticity check:  r2 * ( J r2^T J^-1)  -   I'

      DO i=1,nv
        temp(i,i) = temp(i,i) - ONE
      END DO

      DO i=1,nv
        WRITE( 10, 2002) (temp(i,j), j=1,nv)
      END DO

*

      Enorm = ZERO
      Emax  = ZERO
      DO j=1,nd*2
        DO i=1,nd*2
          Enorm =      Enorm + abs(temp(i,j))
          Emax  = max( Emax   , abs(temp(i,j)) )
        END DO
      END DO
*
      CALL norms( tst, no,nv,nd*2,nd*2, Tnorm, Tmax)

      WRITE( 10, '(1x)')
      WRITE( 10, '(1x)')
      WRITE( 10, '(1x)')
      WRITE( 10, 2003) Enorm, Emax

      WRITE( 10, '(1x)')
      WRITE( 10, 2003) Tnorm(-1), Tmax(-1)

      WRITE( 10, '(1x)')
      DO j=0,no
      WRITE( 10, 2003) Tnorm(j), Tmax(j)
      END DO

*
```

```
*
      WRITE( 10, '(1h1)')
      CALL dapri( Ds0, 10)
      WRITE( 10, *)

      DO i=1,nv
        CALL dapri( Dx1(i), 10)
        WRITE( 10, *)
      END DO
*
      WRITE( 10, '(1h1)')
      CALL dapri( Ds12, 10)
      WRITE( 10, *)
      DO i=1,nv
        CALL dapri( Dx2(i), 10)
        WRITE( 10, *)
      END DO

*
      WRITE( 10, '(1h1)')
      DO i=1,nd*2
        CALL dapri( tst(i), 10)
      END DO

*
      WRITE( 10, '(1h1)')
      CALL dapri( Xc1, 10)
      CALL dapri( Xc2, 10)

      WRITE( 10, '(1h1)')
      CALL dapri( Yc1, 10)
      CALL dapri( Yc2, 10)

*
      WRITE( 10, '(1h1)')
      CALL mypri( Xc1, Xc2, 10)
      CALL mypri( Yc1, Yc2, 10)

*
 2001 FORMAT(1x, 1pe13.6, ';   ', 1pe13.6, ', ', 1pe13.6)
 2002 FORMAT(1x, 13x,     '    ', 1pe13.6, ', ', 1pe13.6)
 2003 FORMAT(1x, 1pe13.6, ', ', 1pe13.6)
*
      RETURN
      END
*
C***************************  End Of File  ******************************
```

G.1.4 LiLens

```
      PROGRAM LiLens2
      IMPLICIT NONE

      INTEGER     NO, NV
      PARAMETER ( NO=10, NV=4 )

      DOUBLE PRECISION   ONE, TWO, FOUR, HALF
      PARAMETER        ( ONE=1.0d0, TWO=2.0d0, FOUR=4.0d0, HALF=0.5d0 )
*
      DOUBLE PRECISION   Pi, Pi2, HalfPi
      COMMON /PiBlok/    Pi, Pi2, HalfPi

      DOUBLE PRECISION   RatScl, RatP0,RatP02, RatPX,RatPX2, RatBR,RatR0
      COMMON /Lens/      RatScl, RatP0,RatP02, RatPX,RatPX2, RatBR,RatR0
*
      EXTERNAL    LiLens
*
C=========================================================================
      pi     = FOUR * atan(ONE)
      pi2    = TWO  * pi
      HalfPi = HALF * Pi
C=========================================================================
*
      CALL DAini( NO,NV, (-10) )
      CALL HJDAdrive( LiLens )
*
      STOP
      END
*


C*************************************************************************
*
      SUBROUTINE HJDAdrive( LiLens )
      IMPLICIT NONE

      INTEGER            NO, NV, ND
      INTEGER            Nstep1, Nstep2, Istep, Nslice, ItMax
      PARAMETER        ( NO=10, NV=4, ND=2 )
      PARAMETER        ( Nstep1=25, Nstep2=1, Istep=6,
     &                   Nslice=10, ItMax=10 )

      INTEGER            Nprim, Nrays,NraysT, Nbuf,NbufT
      PARAMETER        ( Nprim=25, Nrays=4*Nprim, Nbuf=Nrays )
      PARAMETER        ( NraysT=Nrays*Nslice, NbufT=NraysT )

      DOUBLE PRECISION   ZERO, ONE, TWO, FOUR
      PARAMETER        ( ZERO=0.0d0, ONE=1.0d0, TWO=2.0d0, FOUR=4.0d0 )

      DOUBLE PRECISION   P0, Bmax, Brho, R0
      PARAMETER        ( P0      = 8.89d0   )       ! GeV/c
      PARAMETER        ( Bmax    = 100.0d0 )        ! kGauss
      PARAMETER        ( Brho    = 3335.6d0*P0 )    ! kGauss-cm
      PARAMETER        ( R0      = 1.0d0 )          ! cm

      INTEGER            i, j, k, n
      INTEGER            j1,j2,j3,j4, jjj, nnn, Jpin(NV)
      INTEGER            iter, islice
      INTEGER*4          Iseed
      DOUBLE PRECISION   x1(NV), R1(NV,NV)
      DOUBLE PRECISION   x2(NV), R2(NV,NV)
      DOUBLE PRECISION   RI(NV,NV), Temp(NV,NV)

      DOUBLE PRECISION   parm,

      DOUBLE PRECISION   z1, z2, DFdot, w(0:Nslice), a,b

      DOUBLE PRECISION   x01(NV),          x02(NV)
      DOUBLE PRECISION   Xbuf1(Nbuf,NV), Xbuf2(Nbuf,NV)
      DOUBLE PRECISION   XbufT(NbufT,NV)

      DOUBLE PRECISION   Scales(NV), StDevs(NV)
      DOUBLE PRECISION   Xavg(NV), Sigma1(NV,NV), Sigma2(NV,NV)
      DOUBLE PRECISION   DotProd(NO+1), DotByOrd(NO+1)
      DOUBLE PRECISION   TotProd(NO+1), TotByOrd(NO+1)

      DOUBLE PRECISION   Tnorm(-1:no), Tmax(-1:no)
*
*
*
      DOUBLE PRECISION   Pi, Pi2, HalfPi
      COMMON /PiBlok/    Pi, Pi2, HalfPi

      DOUBLE PRECISION   StDevX, StDevP, PZ
      COMMON /Beam/      StDevX, StDevP, PZ

      DOUBLE PRECISION   L0, L1, L2, Ltarg, LLens
      DOUBLE PRECISION   RatScl, RatP0,RatP02, RatPX,RatPX2, RatBR,RatR0
      COMMON /Lens/      RatScl, RatP0,RatP02, RatPX,RatPX2, RatBR,RatR0

      DOUBLE PRECISION   LXscale, PXscale, LZscale, PZscale
      COMMON /Scales/    LXscale, PXscale, LZscale, PZscale
*
      EXTERNAL           LiLens
*
      INTEGER            lx1, lx2, lx3

*DAINT(NO,NV)            DS0,DS12
*DAINT(NO,NV)            Dx1(NV),Dx2(NV), Dy, Dz(NV),Dx(NV)
*DAINT(NO,NV)            DxTild(NV), DxTay(NV), DxMom1, DxMom2

*
C======================================================================
*
      LXscale    = 1.0d0        ! cm
      PXscale    = 1.0d0        ! GeV/c
      LZscale    = 1.0d0        ! cm
      PZscale    = P0           ! GeV/c
*
      StDevX     = 0.5d-1       ! cm
      StDevP     = 1.0d0        ! GeV/c

      StDevs(1)  = StDevX / LXscale
      StDevs(2)  = StDevX / LXscale
      StDevs(3)  = StDevP / PXscale
      StDevs(4)  = StDevP / PXscale

      Scales(1)  = LXscale
      Scales(2)  = LXscale
      Scales(3)  = PXscale
      Scales(4)  = PXscale

      Pz         = P0
*
```

```
*
      RatScl     = (LZscale/LXscale) * (PZscale/PXscale)
      RatP0      = P0 / PZscale
      RatP02     = RatP0 * RatP0
      RatPX      = PXscale / PZscale
      RatPX2     = RatPX * RatPX
      RatBR      = Bmax*R0 /Brho
      RatR0      = R0 / LXscale
*
      L0         =  HalfPi    * R0 / sqrt( RatBR )
      LLens      = (1.0826d0) * L0
      Ltarg      = (5.00d0)        ! cm
*
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C ... Fifth-order Simpson's rule coefficients.
C------------------------------------------------------------------------
      w(0)         = 17.0d0 / 48.0d0
      w(1)         = 59.0d0 / 48.0d0
      w(Nslice-1) = 59.0d0 / 48.0d0
      w(Nslice)    = 17.0d0 / 48.0d0
      DO i=2,(Nstep-2),2
        w(i) = 43.0d0 / 48.0d0
      END DO
      DO i=3,(Nstep-3),2
        w(i) = 49.0d0 / 48.0d0
      END DO
*
C========================================================================
*
       CALL GaussLoad( StDevs, no,2*nd, DxMom1 )
       CALL DApri( DxMom1, 13 )
       CALL norms( DxMom1, no,nd*2,(1), Tnorm, Tmax)
       WRITE( 11, '(1x,a)') 'Norms of DxMom1'
       WRITE( 11, 2001) (Tnorm(n), n=2,NO,2), Tnorm(-1)
       WRITE( 11, 2001) (Tmax( n), n=2,NO,2), Tmax( -1)

         z1    = ZERO
         z2    = LLens - Ltarg

         CALL DAnot( NO )
         CALL HJDXident( NO,NV,ND,NV, Dz, x1,R1,DS0 )
         CALL HJDAbsint( z1, x1,R1,DS0, z2, x2,R2,DS12,  Dz,Dx,
     &                   NO,NV,ND,NV, Nstep1,Istep, LiLens )

         CALL DApri( DS12, 13 )
         CALL norms( DS12, no,nd*2,(1), Tnorm, Tmax)

         WRITE( 10, '(1x,a)') 'Norms of DS12'
         WRITE( 10, 2001) (Tnorm(n), n=2,NO,2), Tnorm(-1)
         WRITE( 10, 2001) (Tmax( n), n=2,NO,2), Tmax( -1)
*
C========================================================================

         DO n=1,(NO+1)
           TotProd(n)   = ZERO
           TotByOrd(n) = ZERO
         END DO

         DO j=1,NV
           DO i=1,NV
             Sigma2(i,j) = ZERO
           END DO
         END DO
```

```
C=========================================================================
*
         CALL DAnot( NO-1 )

         DO j=1,ND
           CALL DAder( (j+ND), DS12, Dz(j)     )
           CALL DAder( (j),    DS12, Dz(j+ND) )
           Jpin(j)    = 1
           Jpin(j+ND) = 0
         END DO

         CALL DApin( Dz,NV, DxTild,NV, Jpin )

         DO j=1,NV
           CALL DAcon( Dy, ZERO )
           DO k=1,NV
             a = R2(j,k)
             CALL DAcma( Dy, DxTild(k),a, Dy )
           END DO
           CALL DAcad( Dy, x2(j), DxTay(j) )
         END DO

         CALL DAnot( NO )
*=========================================================================

         DO j=1,NV
           DO i=1,(j-1)
             CALL DAmul( DxTay(i),DxTay(j), Dy )
             Sigma2(i,j) = w(0) * DFdot( Dy,DxMom1, DotProd,DotByOrd )
             Sigma2(j,i) = Sigma2(i,j)
           END DO
             CALL DAmul( DxTay(j),DxTay(j), Dy )
             Sigma2(j,j) = w(0) * DFdot( Dy,DxMom1, DotProd,DotByOrd )
         END DO

C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C ... Sigma2(4,4) = Sigma2(3,3) is my figure of merit;
C     copy contents of DotProd and DotByOrd (last one is (4,4)
C     resultant) into TotProd and TotByOrd.
C-------------------------------------------------------------------
         DO n=1,(NO+1)
           TotProd(n)  = w(0) * DotProd(n)
           TotByOrd(n) = w(0) * DotByOrd(n)
         END DO

         WRITE( 11, '(1x)')
         WRITE( 11, '(1x,a)') 'Sigma(4,4) by orders'
         WRITE( 11, 2003  ) (DotProd(n),  n=3,(NO+1),2)
         WRITE( 11, 2003  ) (DotByOrd(n), n=3,(NO+1),2)
*
C===================================================================
C===================================================================
*
         DO islice=1,Nslice
           z1 = z2
           z2 = z2 + Ltarg/dfloat(Nslice-1)
           DO j=1,NV
               x1(j) = x2(j)
             DO i=1,NV
               R1(i,j) = R2(i,j)
             END DO
           END DO
           CALL DAcop(DS12,DS0)
           CALL HJDAbsint( z1, x1,R1,DS0, z2, x2,R2,DS12,  Dz,Dx,
     &                     NO,NV,ND,NV, Nstep2,Istep, LiLens )
           CALL DApri( DS12, 13 )
           CALL norms( DS12, no,nd*2,(1), Tnorm, Tmax)
           WRITE( 10, '(1x)')
           WRITE( 10, 2001) (Tnorm(n), n=2,NO,2), Tnorm(-1)
           WRITE( 10, 2001) (Tmax( n), n=2,NO,2), Tmax( -1)
*========================================================================
*
           CALL DAnot( NO-1 )
           DO j=1,ND
             CALL DAder( (j+ND), DS12, Dz(j)     )
             CALL DAder( (j),    DS12, Dz(j+ND) )
             Jpin(j)    = 1
             Jpin(j+ND) = 0
           END DO
           CALL DApin( Dz,NV, DxTild,NV, Jpin )
           DO j=1,NV
             CALL DAcon( Dy, ZERO )
             DO k=1,NV
               a = R2(j,k)
               CALL DAcma( Dy, DxTild(k),a, Dy )
             END DO
             CALL DAcad( Dy, x2(j), DxTay(j) )
           END DO
           CALL DAnot( NO )
*========================================================================
           DO j=1,NV
             DO i=1,(j-1)
               CALL DAmul( DxTay(i),DxTay(j), Dy )
               Sigma2(i,j) = Sigma2(i,j)
     &             + w(islice) * DFdot( Dy,DxMom1, DotProd,DotByOrd )
               Sigma2(j,i) = Sigma2(i,j)
             END DO
               CALL DAmul( DxTay(j),DxTay(j), Dy )
               Sigma2(j,j) = Sigma2(j,j)
     &             + w(islice) * DFdot( Dy,DxMom1, DotProd,DotByOrd )
           END DO
```

```
            WRITE( 11, '(1x)')
            WRITE( 11, 2003  ) (DotProd(n),  n=3,(NO+1),2)
            WRITE( 11, 2003  ) (DotByOrd(n), n=3,(NO+1),2)

            DO n=1,(NO+1)
              TotProd(n)  = TotProd(n)  + w(islice)*DotProd(n)
              TotByOrd(n) = TotByOrd(n) + w(islice)*DotByOrd(n)
            END DO

          END DO
*
C==================================================================
*
          WRITE( 11, '(1x)')
          WRITE( 11, '(1x)')
          WRITE( 11, '(1x,a)') 'Total Sigma(4,4) by orders'
          WRITE( 11, 2003  ) (TotProd(n),  n=3,(NO+1),2)
          WRITE( 11, 2003  ) (TotByOrd(n), n=3,(NO+1),2)
*
 2001 FORMAT( 1x, 1pe13.6, 7(1x, 1pe13.6,:))
 2002 FORMAT( 1x, 'iter = ',i3, ', LLens = ', 1pg13.6,
     &            ', (LLens/L0) = ', 1pg13.6,
     &            ', S33 = ', 1pg13.6, ', S13 = ', 1pg13.6 )
 2003 FORMAT( 1x, 9(1pg13.6,:,1x))
 2012 FORMAT( 1x, 1pe13.6, ';    ', 1pe13.6,:, 3 (', ', 1pe13.6,:))
*
      RETURN
      END
*
```

```
C************************************************************************
*
      SUBROUTINE LiLens( z, Dx, no,nd, Dh)
      IMPLICIT NONE

      DOUBLE PRECISION   ZERO, ONE, TWO, FOUR
      PARAMETER        ( ZERO=0.0d0, ONE=1.0d0, TWO=2.0d0, FOUR=4.0d0 )
*
      INTEGER            no, nd
      DOUBLE PRECISION   z, s, w, R
*
      DOUBLE PRECISION   RatScl, RatP0,RatP02, RatPX,RatPX2, RatBR,RatR0
      COMMON /Lens/      RatScl, RatP0,RatP02, RatPX,RatPX2, RatBR,RatR0

*
      INTEGER            lx2

*DAEXT(no,nd*2)          Dx(nd*2), Dh
*DAINT(no,nd*2)          Dx2, Dy2, Dpx2, Dpy2

*
C========================================================================
*
      w = RatP0 * RatBR /RatR0/RatR0 /TWO
*
      CALL DAsqr( Dx(1), Dx2  )
      CALL DAsqr( Dx(2), Dy2  )
      CALL DAsqr( Dx(3), Dpx2 )
      CALL DAsqr( Dx(4), Dpy2 )

*DA   Dh = RatScl * ( w*(Dx2+Dy2) - sqrt(RatP02 - RatPX2*(Dpx2+Dpy2)) )

      RETURN
      END
*
C****************************  END OF FILE  ****************************
```

G.2 The Hamiltonian Modules

G.2.1 Drift

```
      SUBROUTINE drift( z, Dx, no,nv,nd, Dh)
      DOUBLE PRECISION  zero, one
      PARAMETER        ( zero=0.0d0, one=1.0d0 )
      DOUBLE PRECISION  z
*     DOUBLE PRECISION  beta, gamma, btgma
*     COMMON /relfac/   beta, gamma, btgma
*DAEXT(NO,NV)          DX( ND*2), DH
*DAINT(NO,NV)          PT2, PX2
C==========================================================================
*DA   pt2    = Dx(3) * Dx(3)
*DA   px2    = Dx(4) * Dx(4)
*DA   Dh     = zero - sqrt( pt2 - px2 - one )
      RETURN
      END
*
C*************************  END OF FILE  *******************************
```

G.2.2 PolrSHO

```
C***********************************************************************
*
      SUBROUTINE polrsho( t, Dx, no,nv,nd, Dh)
      INCLUDE             'daparm.inc/LIST'

      DOUBLE PRECISION     onehaf
      PARAMETER          ( onehaf=0.5d0 )

      DOUBLE PRECISION     t

      DOUBLE PRECISION     pi, pi2, w, w2
      COMMON               pi, pi2, w

      DOUBLE PRECISION     r, rsqr
      INTEGER              jj(LNV), kount
      DATA                (jj(i), i=1,LNV) /LNV*0/, kount /0/

*DAEXT(NO,NV)            DX( ND*2), DH
*DAINT(NO,NV)            R2, PR2, PTH2
C=======================================================================

      w2   = w*w

*DA   r2   = Dx(1)*Dx(1)
*DA   pr2  = Dx(3)*Dx(3)

*DA   pth2 = Dx(4)/Dx(1)
*DA   pth2 = pth2*pth2

*DA   Dh = onehaf * ( pr2 + pth2   +   w2*r2 )

      RETURN
      END
*
C*************************  END OF FILE  *******************************
```

G.2.3 UBfield

```
C************************************************************************
*
      SUBROUTINE ubfield( t, Dx, no,nv,nd, Dh)
      DOUBLE PRECISION  t

      DOUBLE PRECISION  pi, pi2
      COMMON            pi, pi2

      DOUBLE PRECISION  beta0, gamma0, btgma0, const,
     &                  beta,  gamma,  btgma,  bgbg

      COMMON /ubfield/  beta0, gamma0, btgma0, const,
     &                  beta,  gamma,  btgma,  bgbg

*DAEXT(no,nv)          Dx(nd*2), Dh
*DAINT(no,nv)          rho, prho

*
C======================================================================


*DA   rho   = Dx(1)
*DA   prho  = Dx(2)

*DA   Dh    = rho * ( const * rho - sqrt ( bgbg - prho*prho ) )

      RETURN
      END
*
C************************** END OF FILE ******************************
```

G.3 The Integration Module

G.3.1 HJDAbsint

```
C************************************************************************
*
      SUBROUTINE HJDAbsint( t1, x1,r1,Ds1,  t2, x2,r2,Ds2, Dz,Dx,
     &                      no,nv,nd,nj,  nstep,istep,   hmltn)
      IMPLICIT NONE

      INTEGER             NMAX
      PARAMETER         ( NMAX=10 )

      DOUBLE PRECISION    ZERO, ONE
      PARAMETER         ( ZERO=0.0d0, ONE=1.0d0 )

      INTEGER             no, nd, nj, nstep, istep

      DOUBLE PRECISION    t1, x1(nj),  r1(nj,nj)
      DOUBLE PRECISION    t2, x2(nj),  r2(nj,nj)

      DOUBLE PRECISION    H0, Hk
      DOUBLE PRECISION    t,  s
      DOUBLE PRECISION    x(NMAX),     r(NMAX,NMAX)
      DOUBLE PRECISION    f(NMAX),     a(NMAX,NMAX)
      DOUBLE PRECISION    Xnext(NMAX), Rnext(NMAX,NMAX)
      DOUBLE PRECISION    Xextr(NMAX), Rextr(NMAX,NMAX)
      DOUBLE PRECISION    Xdiff(NMAX), Rdiff(NMAX,NMAX)

      INTEGER             i,j,k,n, icall, knext

      EXTERNAL            hmltn

      INTEGER             lx3

*DAEXT(no,nv)        DS1, Dz(nd*2), Dx(nd*2)
*DAEXT(no,nv)        DS2
*DAINT(no,nv)        DS, GS, DSnext, DSextr, DSdiff

*
C========================================================================

      DO j=1,nd*2
          x(j)   = x1(j)
          CALL davar( Dz(j), ZERO, j)
        DO i=1,nd*2
          r(i,j) = r1(i,j)
        END DO
      END DO
      CALL dacop(DS1,DS)

        H0 = (t2-t1)/dfloat(nstep)
        t  =  t1

      DO n=1,nstep
        s = dfloat(n-1) / dfloat(nstep)
        t = (ONE-s)*t1 + (s)*t2
        CALL HJDAderiv(t, x,r,DS, f,a,Gs, no,nv,nd,NMAX, Dz,Dx, hmltn)
      DO icall=1,istep
        knext = 2*icall
        Hk = H0 / dfloat(knext)
```

```
         IF ( (t+Hk) .EQ. (t) ) THEN
           PAUSE 'Stepsize not significant in HJDABSINT.'
         END IF

         CALL HJDAmmid( t1,H0,knext, x,r,DS, f,a,GS,
     &        Xnext,Rnext,DSnext, no,nv,nd,NMAX, Dz,Dx, hmltn)

         CALL HJDAextr( icall,knext, Xnext,Xextr,Xdiff,
     &        Rnext,Rextr,Rdiff, DSnext,DSextr,DSdiff, no,nv,nd,NMAX, istep)
       END DO

       DO j=1,nd*2
           x(j)   = Xextr(j)
         DO i=1,nd*2
           r(i,j) = Rextr(i,j)
         END DO
       END DO
       CALL dacop(DSextr,DS)

       END DO
*
       DO j=1,nd*2
           x2(j)   = Xextr(j)
         DO i=1,nd*2
           r2(i,j) = Rextr(i,j)
         END DO
       END DO
       CALL  dacop(DSextr,DS2)
*
       RETURN
       END
*
```

G.3.2 HJDAmmid

```
C***********************************************************************
*
      SUBROUTINE HJDAmmid( t1,H0,nstep,  Xin,Rin,DSin, Fin,Ain,GSin,
     &                     Xout,Rout,DSout,  no,nv,nd,nj, Dz,Dx,   hmltn)
      IMPLICIT NONE
      EXTERNAL hmltn

      INTEGER              NMAX
      PARAMETER          ( NMAX=10 )

      DOUBLE PRECISION     OneHaf, Two
      PARAMETER          ( OneHaf=0.5d0, Two=2.0d0 )

      INTEGER              no,nv,nd,nj, nstep

      DOUBLE PRECISION     t1, H0
      DOUBLE PRECISION     t,  Hk, TwoH, temp
      DOUBLE PRECISION     Xin (NMAX), Rin (NMAX,NMAX)
      DOUBLE PRECISION     Xout(NMAX), Rout(NMAX,NMAX)
      DOUBLE PRECISION     Fin (NMAX), Ain (NMAX,NMAX)

      INTEGER              i, j, n
      DOUBLE PRECISION     Xm(NMAX),   Rm(NMAX,NMAX)
      DOUBLE PRECISION     Xn(NMAX),   Rn(NMAX,NMAX)
      DOUBLE PRECISION     Fn(NMAX),   An(NMAX,NMAX)


      INTEGER              lx2, isala
*DAEXT(no,nv)            DSin,  GSin, DSout,  Dz(nd*2), Dx(nd*2)
*DAINT(no,nv)            DSm, DSn, GSn, Dtemp
*=======================================================================

        Hk = H0/float(nstep)

      DO j=1,(nd*2)
          Xm(j)   = Xin(j)
          Xn(j)   = Xin(j)   + Hk*Fin(j)
        DO i=1,(nd*2)
          Rm(i,j) = Rin(i,j)
          Rn(i,j) = Rin(i,j) + Hk*Ain(i,j)
        END DO
      END DO
      CALL dacop( DSin, DSm)
      CALL dacma( DSin, GSin,Hk, DSn)

        t = t1 + Hk
      CALL HJDAderiv(t, Xn,Rn,DSn, Fn,An,GSn, no,nv,nd,NMAX, Dz,Dx, hmltn)
        TwoH = Two * Hk
      DO n=2,nstep
        DO j=1,(nd*2)
            temp    = Xm(j)    + TwoH*Fn(j)
            Xm(j)   = Xn(j)
            Xn(j)   = temp
          DO i=1,(nd*2)
            temp    = Rm(i,j) + TwoH*An(i,j)
            Rm(i,j) = Rn(i,j)
            Rn(i,j) = temp
          END DO
```

```
  END DO
  CALL dacma( DSm, GSn,TwoH, Dtemp)
  CALL dacop( DSn,   DSm)
  CALL dacop( Dtemp, DSn)

    t = t + Hk
CALL HJDAderiv(t, Xn,Rn,DSn, Fn,An,GSn, no,nv,nd,NMAX, Dz,Dx, hmltn)
END DO

DO j=1,(nd*2)
    Xout(j)   = OneHaf * ( Xm(j)   + Xn(j)   + Hk*Fn(j)   )
  DO i=1,(nd*2)
    Rout(i,j) = OneHaf * ( Rm(i,j) + Rn(i,j) + Hk*An(i,j) )
  END DO
END DO
CALL dacma( DSn, GSn,Hk,   Dtemp)
CALL daadd( DSm, Dtemp,    Dtemp)
CALL dacmu( Dtemp, OneHaf, DSout)

RETURN
END
```

G.3.3 HJDAextr

```
C**********************************************************************
*
      SUBROUTINE HJDAextr( icall,knext, Xnext,Xextr,Xdiff,
     &         Rnext,Rextr,Rdiff, Snext,Sextr,Sdiff, no,nv,nd,nj, nuse)
      IMPLICIT NONE

      DOUBLE PRECISION   ZERO, ONE
      PARAMETER        ( ZERO=0.0d0, ONE=1.0d0 )

      INTEGER            NMAX,    IMAX,    NCOL
      PARAMETER        ( NMAX=10, IMAX=11, NCOL=7 )

      INTEGER            icall, knext, no, nd, nj, nuse
      DOUBLE PRECISION   Xnext(NMAX), Xextr(NMAX), Xdiff(NMAX),
     &         Rnext(NMAX,NMAX), Rextr(NMAX,NMAX), Rdiff(NMAX,NMAX)

      INTEGER            kstack(IMAX), i,j,k, n, n1
      DOUBLE PRECISION   a1, a2, f1, f2, den, delta, temp
      DOUBLE PRECISION   Xd(NMAX),      Xq(NMAX,NCOL)
      DOUBLE PRECISION   Rd(NMAX,NMAX), Rq(NMAX,NMAX,NCOL)
      SAVE               Xq, Rq

      INTEGER       lx2, lx3, isala

*DAEXT(no,nv)       Snext, Sextr, Sdiff
*DAINT(no,nv)       Sq(NCOL),     Sd
*DAINT(no,nv)       Ddelta, Dtemp
*=======================================================================

      kstack(icall) = knext

      DO j=1,(nd*2)
          Xdiff(j)   = Xnext(j)
          Xextr(j)   = Xnext(j)
        DO i=1,(nd*2)
          Rdiff(i,j) = Rnext(i,j)
          Rextr(i,j) = Rnext(i,j)
        END DO
      END DO
      CALL dacop(Snext,Sdiff)
      CALL dacop(Snext,Sextr)

      IF ( icall .EQ. (1) ) THEN

        DO j=1,(nd*2)
            Xd(j)     = ZERO
            Xq(j,1)   = Xnext(j)
          DO k=2,NCOL
            Xq(j,k)   = ZERO
          END DO

          DO i=1,(nd*2)
            Rq(i,j,1) = Rnext(i,j)
          END DO
          DO k=2,NCOL
            Rq(i,j,k) = ZERO
          END DO
```

```
    END DO
    CALL dacop(Snext,Sq(1))
    DO k=2,NCOL
      CALL DAcon(Sq(k),ZERO)
    END DO
  ELSE
      n1 = min(icall,nuse)
    DO j=1,(nd*2)
        Xd(j)   = Xnext(j)
      DO i=1,(nd*2)
        Rd(i,j) = Rnext(i,j)
      END DO
    END DO
    CALL dacop(Snext,Sd)
    DO n=1,(n1-1)
      a1    = float(kstack(icall-n))
      a2    = float(knext)
      den   = ( a2-a1 ) * ( a2+a1 )
      f1    = ( a1*a1 ) / den
      f2    = ( a2*a2 ) / den
      delta = ( a1*a1 * a2*a2 ) / den
      DO j=1,(nd*2)
          temp     =  Xq(j,n)
          Xq(j,n)  =  Xdiff(j)
          delta    =  Xd(j) - temp
          Xdiff(j) =  f1 * delta
          Xd(j)    =  f2 * delta
          Xextr(j) =  Xextr(j) + Xdiff(j)
        DO i=1,(nd*2)
          temp       =  Rq(i,j,n)
          Rq(i,j,n)  =  Rdiff(i,j)
          delta      =  Rd(i,j) - temp
          Rdiff(i,j) =  f1 * delta
          Rd(i,j)    =  f2 * delta
          Rextr(i,j) =  Rextr(i,j) + Rdiff(i,j)
        END DO
      END DO
          CALL dacop( Sq(n), Dtemp)
          CALL dacop( Sdiff,  Sq(n))
          CALL dasub( Sd,     Dtemp,  Ddelta)
          CALL dacmu( Ddelta, f1,     Sdiff)
          CALL dacmu( Ddelta, f2,     Sd)
          CALL daadd( Sextr,  Sdiff,  Sextr)
    END DO
    DO j=1,(nd*2)
        Xq(j,n1)   = Xdiff(j)
      DO i=1,(nd*2)
```

```
            Rq(i,j,n1) = Rdiff(i,j)
          END DO
        END DO
        CALL dacop( Sdiff, Sq(n1))

      END IF

      RETURN
      END
*
C*************************   END OF FILE  ***************************
```

G.4 The Derivative Module

G.4.1 HJDAderiv

```
C***********************************************************************
*
      SUBROUTINE HJDAderiv(t, x,r,DS, f,a,Gs, no,nv,nd,nj, Dz,Dx, hmltn)
      IMPLICIT NONE

      INTEGER                 NMAX
      PARAMETER             ( NMAX=10 )

      DOUBLE PRECISION        ZERO
      PARAMETER             ( ZERO=0.0d0 )

      INTEGER                 no, nd, nj

      DOUBLE PRECISION        t
      DOUBLE PRECISION        x(nj), r(nj,nj)
      DOUBLE PRECISION        f(nj), a(nj,nj)
      DOUBLE PRECISION        rj(NMAX,NMAX)
      DOUBLE PRECISION        h1(NMAX), h2(NMAX,NMAX)

      EXTERNAL                hmltn
*
*
      INTEGER                 lx3

*DAEXT(no,nv)                 DS, Dz(nd*2), Dx(nd*2)
*DAEXT(no,nv)                 GS
*DAINT(no,nv)                 Dh, Dk
*=====================================================================

      CALL HJDX2(    x,r,DS,  no,nv,nd,nj,       Dz, Dx)
      CALL hmltn(    t,  Dx,  no,nv,nd,              Dh)
      CALL hamsplit( Dh, Dz,  no,nv,nd,NMAX, h1,h2,  Dk)

      CALL rjmul( r,nj,     rj,NMAX,            nd)
      CALL mvmul( rj,NMAX,  h1,NMAX,   f,nj,   nd*2)
      CALL mmmul( rj,NMAX,  h2,NMAX,   a,nj,   nd*2)

      CALL dasuc( Dk, ZERO, GS)

      RETURN
      END
*
```

G.4.2 HJDXident

```
C***********************************************************************
*
      SUBROUTINE HJDXident( no,nv,nd,nj,  Dz,   x,r,DS)
      IMPLICIT NONE

      DOUBLE PRECISION     ZERO, ONE
      PARAMETER          ( ZERO = 0.0d0, ONE = 1.0d0 )

      INTEGER              i, j,  no,nv,nd,nj

      DOUBLE PRECISION     x(nj), r(nj,nj)

      INTEGER              isa2a, isa3a

*DAEXT(no,nv)              DS, Dz(nd*2), Dtemp

C=======================================================================
*
      DO j=1,(nd*2)
        DO i=1,(nd*2)
          r( i, j) = ZERO
        END DO
          r( j, j) = ONE
          x( j)    = ZERO
      END DO

        CALL dacon( DS, ZERO)
      DO i=1,nd
        CALL davar( Dz(i),    ZERO,     (i)     )
        CALL davar( Dz(i+nd), ZERO,     (i+nd) )
        CALL damul( Dz(i),    Dz(i+nd), Dtemp )
        CALL daadd( DS,       Dtemp,    DS )
      END DO

      RETURN
      END
*
```

G.4.3 HJDX1

```
C**********************************************************************
*
      SUBROUTINE HJDX1( x,r,DS, no,nv,nd,nj,  Dz,  Dx)
      IMPLICIT NONE

      DOUBLE PRECISION     ZERO
      PARAMETER           ( ZERO=0.0d0 )

      INTEGER               no, nd, nj
      DOUBLE PRECISION      x(nj), r(nj,nj)

      INTEGER               i, j

      DOUBLE PRECISION      x0, a

      INTEGER               lx2, isala

*DAEXT(no,nv)          DS, Dz(nd*2), Dx(nd*2)
*DAINT(no,nv)          Dy
*======================================================================

      DO i=1,nd
        CALL dader(   (nd+i),    DS,   Dz(i) )
        CALL davar( Dz(nd+i), ZERO, (nd+i) )
      END DO

      DO i=1,nd*2
          CALL dacon( Dy, ZERO)
        DO j=1,nd*2
          a     =  r(i,j)
          CALL dacma( Dy, Dz(j),a, Dy )
        END DO
          x0    =  x(i)
          CALL dacad( Dy ,x0, Dx(i) )
      END DO

      RETURN
      END
*
```

G.4.4 HJDX2

```
C***********************************************************************
*
      SUBROUTINE HJDX2( x,r,DS, no,nv,nd,nj,  Dz,  Dx)
      IMPLICIT NONE

      DOUBLE PRECISION     ZERO
      PARAMETER          ( ZERO=0.0d0 )

      INTEGER              no, nd, nj
      DOUBLE PRECISION     x( nj), r( nj, nj)

      INTEGER              i, j

      DOUBLE PRECISION     x0, a

      INTEGER              1x2, isala

*DAEXT(no,nv)              DS, Dz(nd*2), Dx(nd*2)
*DAINT(no,nv)              Dy
*=======================================================================

      DO i=1,nd
        CALL davar( Dz(i), ZERO,      (i) )
        CALL dader(   (i),   DS, Dz(nd+i) )
      END DO

      DO i=1,nd*2
          CALL dacon( Dy, ZERO)
        DO j=1,nd*2
          a     =  r(i,j)
          CALL dacma( Dy, Dz(j),a, Dy )
        END DO
          x0    =  x(i)
          CALL dacad( Dy, x0, Dx(i) )
      END DO

      RETURN
      END
```

G.4.5 HamSplit

```
C***********************************************************************
*
      SUBROUTINE HamSplit( Dh, Dz, no,nv,nd,nj, h1,h2, Dk)
      IMPLICIT NONE
*
      INCLUDE   'DA:daparm.inc/LIST'
      INCLUDE   'DA:dapool.inc/LIST'
*
      DOUBLE PRECISION  ZERO, OneHaf, TWO
      PARAMETER       ( ZERO=0.0d0, OneHaf=0.5d0, TWO =2.0d0 )
*
      INTEGER           no, nd, nj
      INTEGER           i, j, nn(20)
      INTEGER           iordk, isav
      INTEGER*4         ik, ik1, ik2, jk

      INTEGER*4         inoh,invh,ipoh,ilmh,illh
      INTEGER*4         inok,invk,ipok,ilmk,illk

      DOUBLE PRECISION  a, h1(nj), h2(nj,nj)

      INTEGER           lx2
      INTEGER           isa1a, isa2a, isa4a

*DAEXT(no,nv)           Dh, Dz(nd*2), Dk, Dtemp
*=======================================================================
*
      CALL dainf(Dh,inoh,invh,ipoh,ilmh,illh)
      CALL dainf(Dk,inok,invk,ipok,ilmk,illk)

      IF ( ( invh .EQ. 0 ) .OR. ( invk .EQ. 0 ) ) THEN
         PRINT *, 'ERROR: HAMSPLIT called with CA VECTOR'
         CALL dadeb( 111, 'ERR hmsplt', 1)
      END IF

      CALL dachk(Dh,inoh,invh, '          ',-1,-1,Dk,inok,invk)

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C Find and subtract off the linear part of Dh
C----------------------------------------------------------------------

      CALL dacop(Dh,Dk)
      CALL dainf(Dk,inok,invk,ipok,ilmk,illk)

      DO 100 ik=ipok,ipok+illk-1

        ik1    = i1(ik)
        ik2    = i2(ik)

        iordk  = ieo( ia1(ik1) + ia2(ik2) )

        CALL dancd( ik1, ik2, nn)

        IF ( iordk .EQ. (1) ) THEN
          DO  i=1,20
            IF ( nn(i) .EQ. (1) ) THEN
              h1(i) = cc(ik)
              GOTO 100
            END IF
          END DO
        END IF

 100  CONTINUE
```

```
      DO i=1,nd*2
        a = -h1(i)
        CALL dacma( Dk, Dz(i),a, Dk)
      END DO

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C Find and subtract off the quadratic part of Dh.
C------------------------------------------------------------------------

      CALL dainf(Dk,inok,invk,ipok,ilmk,illk)

      DO 200 ik=ipok,ipok+illk-1

        ik1     = i1(ik)
        ik2     = i2(ik)

        iordk   = ieo( ia1(ik1) + ia2(ik2) )

        CALL dancd( ik1, ik2, nn)

        IF ( iordk .EQ. (2) ) THEN
            isav = 0
          DO  i=1,20
            IF ( nn(i) .EQ. (2) ) THEN
              h2(i,i) = cc(ik)
              GOTO 200
            ELSE IF ( nn(i) .EQ. (1)) THEN
              IF ( isav .EQ. (0) ) THEN
                isav = i
              ELSE
                h2( i, isav) = cc(ik)
                h2( isav, i) = cc(ik)
                GOTO 200
              END IF
            END IF
          END DO
        END IF

 200  CONTINUE

      DO i=1,nd*2
        DO j=i,nd*2
          a = -h2(i,j)
          CALL damul( Dz(i), Dz(j),   Dtemp )
          CALL dacma( Dk,     Dtemp,a, Dk )
        END DO
      END DO

      DO i=1,nd*2
        h2(i,i) = TWO*h2(i,i)
      END DO

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  Null out any {order} < (3) parts left by roundoff
C------------------------------------------------------------------------

      CALL dainf(Dk,inok,invk,ipok,ilmk,illk)

          jk      = ipok - 1

      DO 300 ik=ipok,ipok+illk-1

          ik1     = i1(ik)
          ik2     = i2(ik)
          iordk  = ieo( ia1(ik1) + ia2(ik2) )

        IF ( iordk .GE. (3) ) THEN
          jk     = jk + 1
          cc(jk) = cc(ik)
          i1(jk) = i1(ik)
          i2(jk) = i2(ik)
        END IF

 300  CONTINUE

        idall(Dk) = jk - ipok + 1

      IF ( idall(Dk) .GT. idalm(Dk) ) THEN
        PRINT*, 'ERROR in HAMSPLIT: max size of Dk exceeded'
        CALL dadeb(111,'ERR HmSpl',1)
      END IF

      RETURN
      END
*
C**************************  END OF FILE  *******************************
```

G.5 The Matrix Module

G.5.1 RJmul

```
C**********************************************************************
*
      SUBROUTINE RJmul( fa,na,  fj,nj,  nd)

      DOUBLE PRECISION  fa(na,na), fj(nj,nj)

      DO 20 j=1,nd
        DO 10 i=1,nd*2
          fj( i,    j)  =  - fa( i, j+nd)
          fj( i, j+nd)  =    fa( i,    j)
 10     CONTINUE
 20   CONTINUE

      RETURN
      END
*
```

G.5.2 MVmul

```
C**********************************************************************
*
      SUBROUTINE MVmul( fa,na,  vb,nb,  vc,nc,  nj)

      DOUBLE PRECISION   ZERO
      PARAMETER        ( ZERO=0.0d0 )

      DOUBLE PRECISION   a, fa(na,na), vb(nb), vc(nc)
*=======================================================================

      DO 20 i=1,nj

          a = ZERO
        DO 10 j=1,nj
          a = a + fa( i, j) * vb( j)
 10     CONTINUE
          vc( i) = a

 20   CONTINUE

      RETURN
      END
*
```

G.5.3 MMmul

```
C*********************************************************************
*
      SUBROUTINE MMmul( fa,na,  fb,nb,  fc,nc,  nj)

      DOUBLE PRECISION   ZERO
      PARAMETER         ( ZERO=0.0d0 )

      DOUBLE PRECISION   a,  fa(na,na), fb(nb,nb), fc(nc,nc)
*=====================================================================

      DO 30 i=1,nj
      DO 20 k=1,nj

          a = ZERO
        DO 10 j=1,nj
          a = a + fa( i, j) * fb( j, k)
 10     CONTINUE
          fc( i, k) = a

 20   CONTINUE
 30   CONTINUE

      RETURN
      END
*
```

G.5.4 SympInv

```
C*********************************************************************
*
      SUBROUTINE SympInv( fa,na,  fi,ni,  nd)
      DOUBLE PRECISION    fa(na,na), fi(ni,ni)

      DO 20 j=1,nd
        DO 10 i=1,nd

          fi(    j,    i)  =    fa( i+nd, j+nd)
          fi(    j, i+nd)  =  - fa( i,    j+nd)
          fi( j+nd,    i)  =  - fa( i+nd,    j)
          fi( j+nd, i+nd)  =    fa( i,       j)

 10     CONTINUE
 20   CONTINUE

      RETURN
      END
*
C*************************  END OF FILE  ******************************
```

G.6 The Tracking Module

G.6.1 DFeval

```
C***********************************************************************
*
      DOUBLE PRECISION FUNCTION DFeval( x0,Df, no,nv)
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  This function evaluates the result of the DA-vector Df acting
C  on the real vector x0 as a ``trunctated power-series''.
C-----------------------------------------------------------------------

      IMPLICIT NONE

      DOUBLE PRECISION   ZERO, ONE
      PARAMETER         ( ZERO=0.0d0, ONE=1.0d0 )

      INCLUDE           'daparm.inc/LIST'
      INCLUDE           'dapool.inc/LIST'
*
      INTEGER            no,nv
      DOUBLE PRECISION   x0(nv)

      INTEGER            i,j, jj(LNV)
      INTEGER            if,  Df,inof,invf,ipof,ilmf,illf
      DOUBLE PRECISION   q
*
C***********************************************************************
*
      CALL dainf(Df,inof,invf,ipof,ilmf,illf)
*
      IF ( invf .EQ. (0) ) THEN
         PRINT *, 'ERROR, DFeval called with CA vector'
         CALL dadeb( 111, 'ERR DADER ', (1))
      ENDIF
*
        DFeval = ZERO
      DO 100 if=ipof,(ipof+illf-1)
        IF ( ieo( ia1(i1(if)) + ia2(i2(if)) )  .GT.  no ) GOTO 100
        CALL dancd( i1(if),i2(if), jj)
        q = ONE
        DO i=1,nv
          DO j=1,jj(i)
            q = q*x0(i)
          END DO
        END DO
        DFeval = DFeval  +  q*cc(if)
 100  CONTINUE
*
      RETURN
      END
```

G.6.2 DFeval

```
C*************************************************************************
*
      DOUBLE PRECISION FUNCTION DFdot( ina,inb, DotProd, DotByOrd )
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C     This subroutine calculates the Euclidian "dot" product
C     of two DA vectors A and B; the resulting real scalar
C     quantity is returned as the value of "DFdot".
C
C-----------------------------------------------------------------------
      INCLUDE   'DA2:daimplct.inc/LIST'
      INCLUDE   'DA2:daparm.inc/LIST'
      INCLUDE   'DA2:dapool.inc/LIST'
      INCLUDE   'DA2:daname.inc/LIST'

      INTEGER             iord, ii, jj(20)
      DOUBLE PRECISION    Prod, DotProd(*), DotByOrd(*)
*
C=======================================================================
*
      CALL dainf( ina,inoa,inva,ipoa,ilma,illa)
      CALL dainf( inb,inob,invb,ipob,ilmb,illb)

      CALL dachk(ina,inoa,inva, '          ',-1,-1,inb,inob,invb)
*
        DFdot       = 0.0d0
      DO i=1,inoa
        DotProd(i)  = 0.0d0
        DotByOrd(i) = 0.0d0
      END DO
*
      IF ( (illa .EQ. 0) .OR. (illb .EQ. 0) ) RETURN
*
      ia      = ipoa
      ib      = ipob

      iamax   = ipoa + illa - 1
      ibmax   = ipob + illb - 1

      ja      = ia1(i1(ia)) + ia2(i2(ia))
      jb      = ia1(i1(ib)) + ia2(i2(ib))
C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     Compare DA ordering-indices ja and jb
C-----------------------------------------------------------------------
      DO 100 WHILE ( ( ia .LE. iamax ) .AND. ( ib .LE. ibmax ) )
      IF ( ja - jb ) 30, 20, 40

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     Both terms non-zero; accumulate product
C-----------------------------------------------------------------------
  20  CONTINUE
          iord = ieo(ia1(i1(ia))+ia2(i2(ia)))
        IF ( iord .LE. nocut ) THEN
          Prod  = cc(ia) * cc(ib)
          DFdot = DFdot  + Prod
          DotByOrd(iord+1) = DotByOrd(iord+1) +  Prod
        END IF

        ia = ia + 1
        ib = ib + 1
        IF ( ia .GT. iamax ) GOTO 101
        IF ( ib .GT. ibmax ) GOTO 101

        ja = ia1(i1(ia)) + ia2(i2(ia))
        jb = ia1(i1(ib)) + ia2(i2(ib))

      GOTO 100

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     ( ja < jb )  ==>  next nonzero B farther than next nonzero A;
C                       increment "ia" to catch up.
C----------------------------------------------------------------------
   30 CONTINUE
        ia = ia + 1
        ja = ia1(i1(ia)) + ia2(i2(ia))
      GOTO 100

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     ( ja > jb )  ==>  next nonzero A farther than next nonzero B;
C                       increment "ib" to catch up.
C----------------------------------------------------------------------
   40 CONTINUE
        ib = ib + 1
        jb = ia1(i1(ib)) + ia2(i2(ib))
      GOTO 100

C++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C     End comparison block
C----------------------------------------------------------------------
  100 CONTINUE
*
  101 CONTINUE
        DotProd(1) = DotByOrd(1)
      DO n=2,(nocut+1)
        DotProd(n) = DotByOrd(n) +  DotProd(n-1)
      END DO
*
      RETURN
      END
*
C****************************  END OF FILE  ****************************
```

G.7 Gaussload

```
C***********************************************************************
*
      SUBROUTINE GaussLoad( StDev, no,nv, DxMom )
      IMPLICIT NONE
*
      INTEGER                 no, nv
      DOUBLE PRECISION        StDev(*)
*
      INTEGER                 j1,j2,j3,j4, jj(4), j
      INTEGER                 n1,n2,n3,n4
      DOUBLE PRECISION        Xjj, GaussMom
*
      INTEGER                 DxMom
      CALL DAall( DxMom,1, 'DxMom     ', no,nv )
*
C=======================================================================
*
      IF ( nv .GT. (20) ) THEN
         PRINT *, 'ERROR IN GaussLoad, NV = ',NV
         RETURN
      END IF
*
        n4 = no
      DO 100 j4=0,n4,2
        n3     =  n4 - j4
        jj(4)  =  j4
      DO 100 j3=0,n3,2
        n2     =  n3 - j3
        jj(3)  =  j3
      DO 100 j2=0,n2,2
        n1     =  n2 - j2
        jj(2)  =  j2
      DO 100 j1=0,n1,2
        jj(1)  =  j1

        Xjj = GaussMom( StDev, no,nv, jj )
        CALL DApok( DxMom, jj, Xjj )

      WRITE( 13, 2001 ) Xjj, (jj(j),j=1,4)
 2001 FORMAT( 1x, 1pe13.6, 3x, 4i3 )

 100  CONTINUE
*
      RETURN
      END
*


C***********************************************************************
*
      DOUBLE PRECISION FUNCTION GaussMom( StDev, no,nv, jj )
      IMPLICIT NONE
*
      INTEGER               no, nv, jj(*)
      DOUBLE PRECISION      StDev(*)

      INTEGER               i, j
      DOUBLE PRECISION      x
*
C=======================================================================
*
```

```
          GaussMom = 1.0d0
      DO j=1,nv
        IF ( mod(jj(j),(2)) .NE. (0) ) THEN
          GaussMom = 0.0d0
          RETURN
        END IF
        x = StDev(j) * StDev(j)
        DO i=1,(jj(j)-1),2
          GaussMom = dfloat(i) * x * GaussMom
        END DO
      END DO
*
      RETURN
      END
```

G.8 RayGen

```
C***********************************************************************
*
      SUBROUTINE SigCalc( Xbuf,Nbuf, Nv,Nrays, Xavg,Sigma )
      IMPLICIT NONE

      DOUBLE PRECISION     ZERO
      PARAMETER          ( ZERO=0.0d0 )

      INTEGER              i,j,n, Nbuf, Nv, Nrays
      DOUBLE PRECISION     Xbuf(Nbuf,Nv), Xavg(Nv), Sigma(Nv,Nv)
*
C======================================================================
*
      DO j=1,Nv
          Xavg(j)     = ZERO
        DO i=1,Nv
          Sigma(i,j) = ZERO
        END DO
      END DO

      DO j=1,Nv
        DO n=1,Nrays
          Xavg(j)   = Xavg(j) + Xbuf(n,j)/dfloat( Nrays )
        END DO
      END DO

      DO j=1,Nv
        DO i=1,Nv
          DO n=1,Nrays
            Sigma(i,j) = Sigma(i,j) +
     &      (Xbuf(n,i)-Xavg(i)) * (Xbuf(n,j)-Xavg(j)) / dfloat(Nrays)
          END DO
        END DO
      END DO
*
      RETURN
      END
*


C***********************************************************************
*
      SUBROUTINE RayGen( Xbuf,Nbuf, Nprim, Iseed )
      IMPLICIT NONE

      INTEGER             NV, Nmax
      PARAMETER         ( NV=4, Nmax=4000 )

      DOUBLE PRECISION    ONE, TWO
      PARAMETER         ( ONE=1.0d0, TWO=2.0d0 )

      INTEGER             Nbuf, Nprim
      INTEGER*4           Iseed
      DOUBLE PRECISION    Xbuf(Nbuf,NV), Ybuf(Nmax,NV)

      INTEGER             n,nn,nnn, nsav
      DOUBLE PRECISION    s

      DOUBLE PRECISION    Sx, Spx, Pz
      COMMON /Beam/       Sx, Spx, Pz
      DOUBLE PRECISION   Xscale,PXscale, Zscale,PZscale
      COMMON /Scales/    Xscale,PXscale, Zscale,PZscale
*
C======================================================================
*
        CALL SphereGen( Ybuf,Nmax, Nprim, Iseed )

        nnn = 0
      DO nn=1,Nprim
        nnn = nnn+1
        Xbuf(nnn,1)   =  (Sx/Xscale)    * Ybuf(nn,1)
        Xbuf(nnn,2)   =  (Sx/Xscale)    * Ybuf(nn,2)
        Xbuf(nnn,3)   =  (Spx/PXscale) * Ybuf(nn,3)
        Xbuf(nnn,4)   =  (Spx/PXscale) * Ybuf(nn,4)

        nsav = nnn

        nnn = nnn+1
        Xbuf(nnn,1)   = - Xbuf(nsav,1)
        Xbuf(nnn,2)   = - Xbuf(nsav,2)
        Xbuf(nnn,3)   =   Xbuf(nsav,3)
        Xbuf(nnn,4)   =   Xbuf(nsav,4)

        nnn = nnn+1
        Xbuf(nnn,1)   =   Xbuf(nsav,1)
        Xbuf(nnn,2)   =   Xbuf(nsav,2)
        Xbuf(nnn,3)   = - Xbuf(nsav,3)
        Xbuf(nnn,4)   = - Xbuf(nsav,4)

        nnn = nnn+1
        Xbuf(nnn,1)   = - Xbuf(nsav,1)
        Xbuf(nnn,2)   = - Xbuf(nsav,2)
        Xbuf(nnn,3)   = - Xbuf(nsav,3)
        Xbuf(nnn,4)   = - Xbuf(nsav,4)

      END DO
*
      RETURN
      END
*


C*************************************************************************
*
      SUBROUTINE SphereGen( Ybuf,Nbuf, Nrays, Iseed)
      IMPLICIT NONE

      INTEGER            NV,   Nmax
      PARAMETER        ( NV=4, Nmax=4000 )

      DOUBLE PRECISION   ZERO, ONE, TWO, HALF, R0
      PARAMETER        ( ZERO=0.0d0, ONE=1.0d0, TWO=2.0d0, HALF=0.5d0 )
      PARAMETER        ( R0=3.0d0 )

      INTEGER            Nbuf, Nrays
      DOUBLE PRECISION   Zbuf(Nmax,NV), Ybuf(Nbuf,NV)

      INTEGER*4          Iseed
*     DATA               Iseed/-123454321/    ! used when routine autoinit'd
*     SAVE               Iseed                ! used when routine autoinit'd

      INTEGER            i,j,k, n, nout, Itrial

      DOUBLE PRECISION   a, z(NV), rho,   Rmax
      DOUBLE PRECISION   T(NV,NV), D(NV), E(NV)
      DOUBLE PRECISION   Xavg(NV), Sigma(NV,NV)
      DOUBLE PRECISION   S(NV,NV)
*
C===================================================================
*
        Itrial = 0
        nout   = 999
      DO WHILE ( ( Itrial .LT. (20) ) .AND. ( nout .GT. (0) ) )
        Itrial = Itrial + 1
        DO j=1,NV
          Xavg(j) = ZERO
        END DO
        DO n=1,Nrays
          CALL MakeRay( Iseed,R0, z )
            a = ZERO
          DO j=1,NV
            a = a + z(j)*Xavg(j)
          END DO
          DO j=1,NV
            IF ( a .LT. (ZERO) ) THEN
              zbuf(n,j) =   z(j)
              Xavg(j)   = Xavg(j) + z(j)
            ELSE
              zbuf(n,j) = - z(j)
              Xavg(j)   = Xavg(j) - z(j)
            END IF
          END DO
        END DO
*
          k    = 0
          nout = 999
        DO WHILE( ( k .LT. (Nrays/4) ) .AND. ( nout .GT. (0) ) )
          k    = k + 1
          CALL SigCalc( Zbuf,Nmax, NV,Nrays, Xavg,Sigma )
*
          DO j=1,NV
            DO n=1,Nrays
              Zbuf(n,j) = Zbuf(n,j) - Xavg(j)
            END DO
            DO i=1,NV
              T(i,j) = Sigma(i,j)
            END DO
          END DO

          CALL tred2( T,   NV,NV, D,E )
          CALL  tqli( D,E, NV,NV, T   )

          DO j=1,NV
            IF ( (D(j).GT.TWO) .OR. (D(j).LT.HALF) ) THEN
              PRINT *, 'Failure; restart at Itrial = ',Itrial
              nout=999
              GOTO 100   ! Restart
            END IF
          END DO
*
            Rmax = ZERO
            nout = 0
          DO n=1,Nrays
            DO j=1,NV
              a   = ZERO
              rho = ZERO
              DO i=1,NV
              a = a + Zbuf(n,i)*T(i,j)
            END DO
            Ybuf(n,j) = a / sqrt( D(j) )
            rho = rho + Ybuf(n,j)*Ybuf(n,j)
          END DO
          Rmax = max( rho, Rmax)
          IF ( rho .GT. R0*R0 ) THEN
            nout = nout + 1
            CALL MakeRay( Iseed,R0, z )
            DO j=1,NV
              Zbuf(n,j) = z(j)
            END DO
          END IF
        END DO   ! n=1,Nrays
      END DO     ! WHILE( (k.LT.(10)).AND.(nout.GT.(0)) )
*
      IF ( (k.EQ.(Nrays/4)).AND.(nout.NE.(0)) ) THEN
        PRINT *, 'bad rays after (Nrays/4) sweeps; restarting'
      ELSE
        RETURN
      END IF
 100  CONTINUE
      END DO
*
      PRINT *, 'ERROR in SphereGen: failure after Imax Restarts'
      STOP
      END
*


C***********************************************************************
*
      SUBROUTINE MakeRay( Iseed,R0, z )
      IMPLICIT NONE

      DOUBLE PRECISION    ONE, TWO
      PARAMETER         ( ONE=1.0d0, TWO=2.0d0 )

      INTEGER*4           Iseed
      DOUBLE PRECISION    R0, z(*)

      INTEGER             k
      DOUBLE PRECISION    a, u,v, r,p, Umin, Ran0, rho

      DOUBLE PRECISION    Pi, Pi2, HalfPi
      COMMON /PiBlok/     Pi, Pi2, HalfPi
*
C=======================================================================
      Umin = exp(-R0*R0/TWO)
C=======================================================================
*
      u = Pi2 * Ran0(Iseed)
      v = Pi2 * Ran0(Iseed)

      r = sqrt( - TWO * log( Umin + (ONE-Umin)*Ran0( Iseed ) ) )
      p = sqrt( - TWO * log( Umin + (ONE-Umin)*Ran0( Iseed ) ) )
*
      z(1) = r * cos( u )
      z(2) = r * sin( u )

      z(3) = p * cos( v )
      z(4) = p * sin( v )
*
      RETURN
      END
```

*

Vita

- **Name:** Gordon D. Pusch [illegible]
- **Date of Birth:** 14 August, 1958
- **Education**
  - ▹ **Ph.D. Physics** Mar 1990, Viginia Polytechnic Institute and State University, Blacksburg, VA.
  - ▹ **B.S. Physics** Dec 1980, Bradley University, Peoria, IL.
- **Areas of Specialization**
  - ▹ Accelerator Physics, particularly applications of differential and Lie algebraic techniques.
  - ▹ Computational Physics.
  - ▹ Numerical Analysis.
- **Related Experience**
  - ▹ 1989 Summer Internship with Instrumentation and Controls group, CEBAF.
  - ▹ 1987–1988 Summer Internship/Research Assistantship with AT-3 Accelerator Technology group, Los Alamos National Laboratory.

- **Publications**
  - ▷ *Differential Algebraic Method for Obtaining Approximate Solutions to the Hamilton-Jacobi Equation*, 1990 (to be submitted for publication).
  - ▷ *A New Test of the Weak Equivalence Principle*,
    (Third Prize essay, 1986 Gravity Research Foundation essay competition)
    General Relativity and Gravitation **19**, 225 (1987).
  - ▷ *Neutron-Antineutron Oscillations in a Periodically Varying Magnetic Field*
    Il Nuovo Cimento **74A**, 149 (1983).
- **Honors and Awards**
  - ▷ Author of third prize essay, 1986 Gravity Research Foundation essay competition.
  - ▷ Member $\Sigma\Pi\Sigma$ (American Physical Society Honor Society).