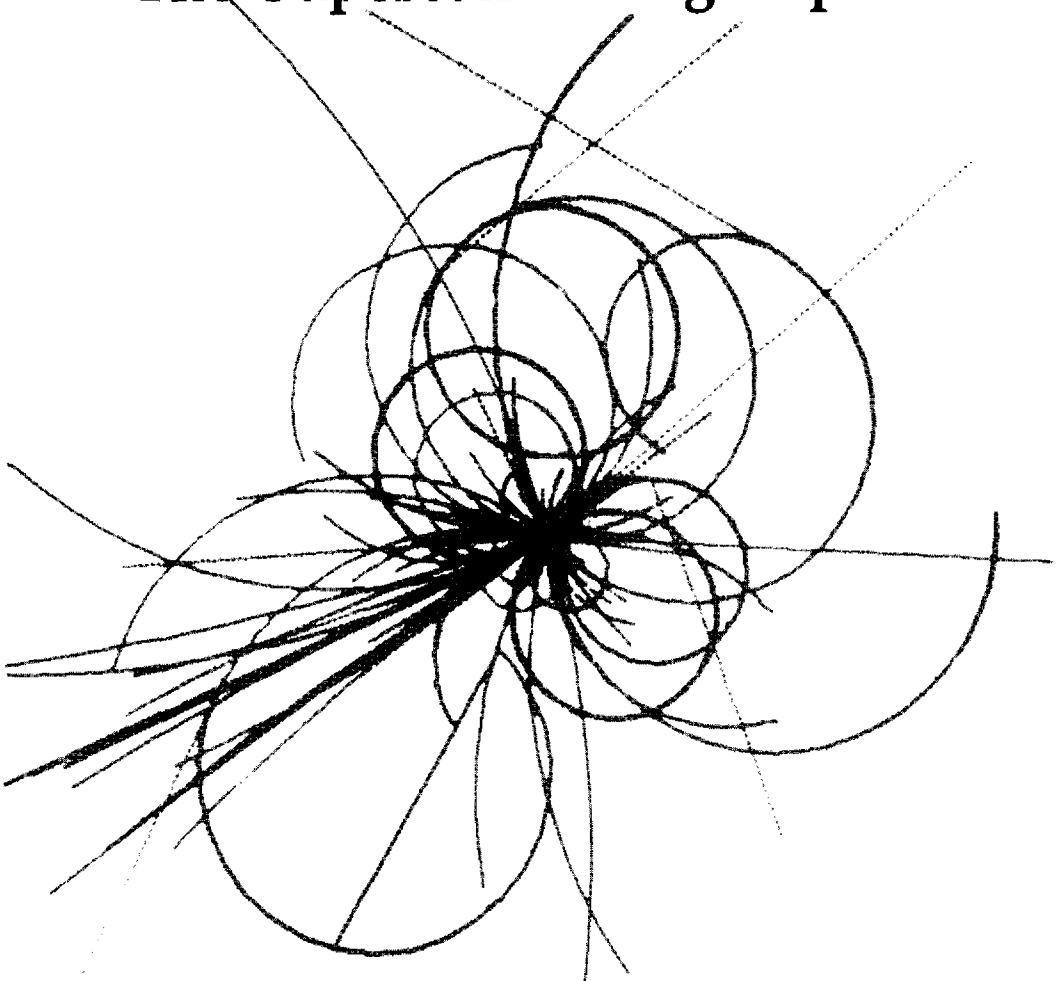
SSC-166

# The Superconducting Super Collider



# Exact Computation of Derivatives with Differential Algebra and Applications to Beam Dynamics

E. Forest, M. Berz, and J. Irwin
SSC Central Design Group

March 1988

SSC-166

# EXACT COMPUTATION OF DERIVATIVES WITH DIFFERENTIAL ALGEBRA AND APPLICATIONS TO BEAM DYNAMICS

Part I. A Survey of Differential Algebra and
Its Use for the Extraction of Maps to Arbitrary Order

M. Berz

Part II. Normal Form Methods for Complicated Periodic Systems: A Complete
Solution Using Differential Algebra and Lie Operators

E. Forest, M. Berz, and J. Irwin
SSC Central Design Group*
c/o Lawrence Berkeley Laboratory, Berkeley, CA 94720

March 1988

*Operated by Universities Research Association for the U. S. Department of Energy

# Part I
# A Survey of Differential Algebra and Its Use for the Extraction of Maps to Arbitrary Order *

M. Berz

SSC Central Design Group
Universities Research Association
c/o Lawrence Berkeley Laboratory
Berkeley, Ca 94720

**Abstract**

The new method of differential algebras for the description of beam dynamics is presented. It allows a very straightforward and elegant computation of transfer maps of accelerator systems which can be analysed for data relevant to accelerators. The order of the procedure is unlimited and it is accurate to machine precision. The theoretical background of the method is presented in detail. It is shown how the method can be used in practice.

## 1 Introduction

The effect of an accelerator section can be described mathematically by a map relating the final coordinates $\vec{z}_f$ of a particle to the initial coordinates $\vec{z}_i$

*Part I submitted to Particle Accelerators under the title "Differential Algebraic Description of Beam Dynamics to Very High Orders"

$$\vec{z}_f = \mathcal{M}(\vec{z}_i, \vec{\delta}) \tag{1}$$

The coordinates $\vec{z}$ contain positions and momenta of the particle. The vector $\vec{\delta}$ contains other parameters that influence the motion like energy, mass or charge of the particle or accelerator parameters like certain multipole strengths. From this map $\mathcal{M}$, quantities of interest for accelerators like tune shifts and chromaticities can be extracted. This is described in detail in a companion paper [16].

Except for the most trivial cases it is impossible to find a closed analytical solution for the map $\mathcal{M}$. However, expanding $\mathcal{M}$ in a power series yields a set of differential equations for the expansion coefficients which in many cases can be solved analytically up to some order. The complexity of the resulting differential equations, however, increases dramatically with the order of the expansion coefficients. So this procedure is limited to low or medium orders. In fact, most widely used codes [1], [2], [3] only have the capability to compute nonlinearities in (1) through third order.

Recently it has been possible to extend this to higher orders using the custom made formula manipulator HAMILTON [4]. This program produces FORTRAN code for formulas of nonlinearities compatible with the program COSY [5]. At the present time, COSY can compute all fifth order nonlinearities of common beam line elements like bending magnets and magnetic and electrostatic multipoles including the dependence on the particle mass.

Since the accuracy of the Taylor series representation of the map in equation (1) increases by using higher orders, it is desirable to know $\mathcal{M}$ to as high an order as possible. This certainly holds for the purpose of using the map for subsequent symplectic tracking of simulation particles; but also for the above mentioned purely analytical purposes like computation of tune shifts with amplitude, chromaticities, invariants etc. based on Hamiltonian perturbation theory [6], [7], [16], this is highly desirable.

In this paper we will present a very straightforward way to compute nonlinearities to arbitrary orders based on differential algebraic techniques. The partial derivatives are computed to machine precision; the whole procedure is completely independent of the order and only limited by the power of the computer.

Besides the elegance in the computation of transfer maps, the method is so versatile that it allows the computation of arbitrary derivatives. Here no analytic formulas for derivatives have to be derived; on the other hand, the method is always accurate to machine precision independent of the order of the derivative which is in sharp contrast to methods of numerical differentiation.

# 2 Differential Algebras

In this section we will provide the mathematical background of the theory of differential algebras required for the advertised study of nonlinearities. Differential algebras are an offspring of the relatively new field of Nonstandard Analysis [8], [9], which allows the introduction of arbitrarily small quantities, "infinitesimals", in a rigorous theory of analysis. There is also some connection to the theories of formal power series [10] and automated differentiation [11]. The use of differential algebras for the field of nonlinear dynamics was first discussed in [12].

For the sake of clarity, we first address the simplest case of differential algebras, the structure ${}_1D_1$.

## 2.1 The Structure ${}_1D_1$

Consider the vector space $R^2$ of ordered pairs $(a_0, a_1)$, $a_0, a_1 \in R$ in which an addition and a scalar multiplication are defined in the usual way:

$$(a_0, a_1) + (b_0, b_1) = (a_0 + b_0, a_1 + b_1) \tag{2}$$

$$t \cdot (a_0, a_1) = (t \cdot a_0, t \cdot a_1) \tag{3}$$

for $a_0, a_1, b_0, b_1 \in R$. Besides the above addition and scalar multiplication a multiplication between vectors is introduced in the following way:

$$(a_0, a_1) \cdot (b_0, b_1) = (a_0 \cdot b_0, a_0 \cdot b_1 + a_1 \cdot b_0) \tag{4}$$

for $a_0, a_1, b_0, b_1 \in R$. With this definition of a vector multiplication the set of ordered pairs becomes an algebra, denoted by ${}_1D_1$.

Note that the multiplication is the same one would obtain by multiplying $(a_0 + a_1 \cdot x)$ and $(b_0 + b_1 \cdot x)$ and keeping terms linear in x.

In the same way than in the case of complex numbers, one can identify $(a_0, 0)$ as the real number $a_0$. Where in the complex numbers, $(0, 1)$ was a root of -1, here it has another interesting property:

$$(0, 1) \cdot (0, 1) = (0, 0) \tag{5}$$

which follows directly from equation (4). So $(0,1)$ is a root of 0. Such a property suggests thinking of $d = (0,1)$ as something infinitely small, small enough that its square vanishes. Because of this we call $d = (0,1)$ the differential unit. The first component of the pair $(a_0, a_1)$ is called the real part, and the second component is called the differential part.

It is easy to verify that $(1,0)$ is a neutral element of multiplication, because according to equation (4)

$$(1,0) \cdot (a_0, a_1) = (a_0, a_1) \cdot (1,0) = (a_0, a_1) \tag{6}$$

It turns out that $(a_0, a_1)$ has a multiplicative inverse if and only if $a_0$ is nonzero; so ${}_1D_1$ is not a field. In case $a_0 \neq 0$ the inverse is

$$(a_0, a_1)^{-1} = (\frac{1}{a_0}, -\frac{a_1}{a_0^2}) \tag{7}$$

Using equations it is easy to check that in fact $(a_0, a_1)^{-1} \cdot (a_0, a_1) = (1,0)$.

The space ${}_1D_1$ is a subspace of the field $R^*$ introduced in Nonstandard Analysis [8],[9]. Besides the usual real numbers, $R^*$ contains a variety of infinitely small and infinitely large quantities. The outstanding result of the theory of Nonstandard Analysis is that differentiation becomes an algebraic problem: a function $f$ is differentiable if and only if for any arbitrarily small quantity $\delta$, the real part of the quotient

$$\frac{f(x+\delta) - f(x)}{\delta} \tag{8}$$

is independent of the choice of the specific $\delta$. Thus, given any differentiable function $f$, we can compute its derivatives by just evaluating the formula for a special choice of $\delta$. We choose $\delta = d = (0,1)$ and thus obtain

$$f'(x) = \mathcal{R}[\frac{f(x+d) - f(x)}{d}] \text{ or } f'(x) = \mathcal{D}[f(x+d) - f(x)] \tag{9}$$

where $\mathcal{R}$ denotes the real part and $\mathcal{D}$ denotes the differential part. Hence differential algebras are useful to compute derivatives directly, without requiring an analytic formula for the derivative and without the inaccuracies of numerical techniques.

The computation of derivatives shall be illustrated in an example using the following function:

$$f(x) = \frac{1}{x + \frac{1}{x}} \tag{10}$$

The derivative of the function is

$$f'(x) = \frac{\frac{1}{x^2} - 1}{(x + \frac{1}{x})^2} \tag{11}$$

Suppose we are interested in the value of the function and its derivative at x=2. We obtain

$$f(2) = \frac{2}{5} \quad f'(2) = -\frac{3}{25} \tag{12}$$

Now take the definition of the function $f$ in equation (10) and evaluate it at $2 + d = (2, 1)$. One obtains:

$$\begin{aligned} f[(2,1)] &= \frac{1}{(2,1) + \frac{1}{(2,1)}} \\ &= \frac{1}{(2,1) + (\frac{1}{2}, -\frac{1}{4})} \\ &= \frac{1}{(\frac{5}{2}, \frac{3}{4})} \\ &= (\frac{2}{5}, -\frac{3}{4}/\frac{25}{4}) \\ &= (\frac{2}{5}, -\frac{3}{25}) \end{aligned} \tag{13}$$

As we can see, after the evaluation of the function the real part of the result is just the value of the function at $x = 2$, whereas the differential part is the derivative of the function at $x = 2$.

This is exactly what was to be expected from the theory of Nonstandard Analysis. However, for the sake of not relying on the quite advanced techniques of this relatively new field of mathematics, we also present an elementary but less proof of the result.

By our choice of the starting vector $(2, 1)$, initially the vector contains the value $I(2)$ of the identity function $I : x \to x$ in the first component and the derivative of $I'(2) = 1$ in the second component.

Now assume that in an intermediate step two vectors of value and derivative $(g(2), g'(2))$ and $(h(2), h'(2))$ have to be added. According to (2) one obtains $(g(2) + h(2), g'(2) + h'(2))$. But according to the rule for the differentiation of sums, this is just the value and derivative of the sum function $(g + h)$ at $x = 2$.

The same holds for the multiplication: Suppose that two vectors of value and derivatives $(g(2), g'(2))$ and $(h(2), h'(2))$ have to be multiplied. Then according to (4) one obtains $(g(2) \cdot h(2), g(2) \cdot h'(2) + g'(2) \cdot h(2))$. But according to the product rule, this is just the value and derivative of the product function $(g \cdot h)$ at $x = 2$.

The evaluation of the function $f$ at $(2, 1)$ can now be viewed as successively combining two intermediate functions $g$ and $h$, starting with the identity function and finally arriving at $f$. At each intermediate step the derivative of the intermediate function is automatically obtained as the differential part according to the above reasoning.

An interesting side aspect is that with the search for a multiplicative inverse in equation (7) one has derived a rule to differentiate the function $f(x) = 1/x$ without explicitly using calculus rules.

After discussing the algebra ${}_1D_1$ and its virtues for the computation of derivatives, we now address the most general differential algebra, the structure ${}_nD_v$. It will eventually allow us to arithmetically compute partial derivatives of functions of $v$ variables through order n.

## 2.2 The Structure ${}_nD_v$

First we define $N(n, v)$ to be the number of monomials in $v$ variables through order $n$. It can be shown that $N(n, v) = \frac{(n+v)!}{n!v!} = C(n + v, v)$ where $C(i, j)$ is the familiar binomial coefficient. Now assume that all these $N$ monomials are arranged in a certain manner order by order. For each monomial $M$ we call $I_M$ the position of $M$ according to the ordering. Conversely, with $M_I$ we denote the $I$th monomial of the ordering. Finally, for an I with $M_I = x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}$ we define $F_I = i_1! \cdot \ldots \cdot i_v!$.

We now define an addition, a scalar multiplication and a vector multiplication on $R^N$ in the following way:

$$(a_1, \ldots, a_N) + (b_1, \ldots . b_N) = (a_1 + b_1, \ldots, a_N + b_N) \tag{14}$$

$$t \cdot (a_1, ..., a_N) = (t \cdot a_1, ..., t \cdot a_N) \tag{15}$$

$$(a_1, ..., a_N) \cdot (b_1, ....b_N) = (c_1, ..., c_N) \tag{16}$$

where the coefficients $c_i$ are defined as follows:

$$c_i = F_i \sum_{\substack{0 \leq \nu, \mu \leq N \\ M_\nu \cdot M_\mu = M_i}} \frac{a_\nu \cdot b_\mu}{F_\nu \cdot F_\mu} \tag{17}$$

To help clarify these definitions, let us look at the case of two variables and second order. In this case, we have $n = 2$ and $v = 2$. There are $N = C(2+2, 2) = 6$ monomials in two variables, namely

$$1,\ x,\ y,\ xx,\ xy,\ yy \tag{18}$$

As an example, using the ordering in (18), we have $I_{xy} = 5$ and $M_3 = y$. Using the ordering in (18), we obtain for $c_1$ through $c_6$ in equation (17):

$$\begin{aligned}
c_1 &= a_1 \cdot b_1 \\
c_2 &= a_1 \cdot b_2 + a_2 \cdot b_1 \\
c_3 &= a_1 \cdot b_3 + a_3 \cdot b_1 \\
c_4 &= 2 \cdot (a_1 \cdot b_4/2 + a_2 \cdot b_2 + a_4 \cdot b_1/2) \\
c_5 &= a_1 \cdot b_5 + a_2 \cdot b_3 + a_3 \cdot b_2 + a_5 \cdot b_1 \\
c_6 &= 2 \cdot (a_1 \cdot b_6/2 + a_3 \cdot b_3 + a_6 \cdot b_1/2)
\end{aligned} \tag{19}$$

Where in ${}_1D_1$, $d = (0, 1)$ was an infinitely small quantity, here we have a whole variety of infinitely small quantities that have the property that high enough powers of them vanish. We give special names to the ones in components $I$ belonging to first order monomials, denoting them by $dM_I$. In the example of ${}_2D_2$, we have $dx = (0, 1, 0, 0, 0, 0)$ and $dy = (0, 0, 1, 0, 0, 0)$. It then follows from the theory of Nonstandard Analysis that instead of equation (9) we obtain

$$f(x+dx, y+dy) =$$
$$(f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial x \partial y}, \frac{\partial^2 f}{\partial y^2})(x,y) \tag{20}$$

In the general case of v variables and order n, after evaluating f in the differential algebra one obtains:

$$\frac{\partial^{i_1+i_2+\ldots+i_v} f}{\partial x_1^{i_1} \partial x_2^{i_2} \ldots \partial x_v^{i_v}} = c_{I_{(x_1^{i_1} \cdot \ldots \cdot x_v^{i_v})}} \tag{21}$$

where $I_{(x_1^{i_1} \cdot \ldots \cdot x_v^{i_v})}$ is the index of the monomial $(x_1^{i_1} \cdot \ldots \cdot x_v^{i_v})$, as defined in the beginning of the section.

# 3 Important Functions on Differential Algebras

In this section we will generalize standard functions like exponentials, logarithmic and trigonometric function to differential algebra. As we will see below, virtually all functions existing on a computer can be generalized in a straightforward way.

We start our discussion by noting that for any differential algebra vector of the form $(0, a_1, \ldots, a_N) \in {}_nD_v$, i.e. with a zero in the component belonging to the zeroth order monomial, we have the following property:

$$(0, a_1, \ldots, a_N)^i = (0, 0, \ldots, 0) \quad \text{for } i > n \tag{22}$$

which follows directly from the definition of the multiplication in ${}_nD_v$ defined in equation (16).

Let us begin our discussion of special functions with the exponential function $\exp(x)$. Assume we have to compute the exponential of a differential algebra vector that has already been created by previous operations. First we note that the functional equation $\exp(x+y) = \exp(x) \cdot \exp(y)$ also holds in Nonstandard Analysis. As we will see, this facilitates the computation of the exponential considerably.

$$
\begin{aligned}
\exp[(a_0, a_1, a_2, ..., a_N)] &= \exp(a_0) \cdot \exp[(0, a_1, a_2, ..., a_N)] \\
&= \exp(a_0) \cdot \sum_{i=0}^{\infty} \frac{(0, a_1, a_2, ..., a_N)^i}{i!} \\
&= \exp(a_0) \cdot \sum_{i=0}^{n} \frac{(0, a_1, a_2, ..., a_N)^i}{i!} \qquad (23)
\end{aligned}
$$

In the last step use has been made of equation (22) which entails that the sum has to be taken only through order n and thus allows the computation of the root in finitely many steps. Hence the evaluation of the real number exponential $\exp(a_0)$ which internally on a computer requires a power series summation and hence cannot be done accurately, is more subtle then the rest of the operations in differential algebra.

A logarithm of a differential algebra vector exists if and only if $a_0 > 0$. In this case one obtains

$$
\begin{aligned}
\log[(a_0, a_1, a_2, ..., a_N)] &= \log[a_0 \cdot (1 + (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ... \frac{a_N}{a_0})] \\
&= (\log(a_0), 0, ..., 0) + \sum_{i=1}^{\infty} (-1)^{i+1} \frac{1}{i} (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})^i \\
&= (\log(a_0), 0, ..., 0) + \sum_{i=1}^{n} (-1)^{i+1} \frac{1}{i} (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})^i \\
& \qquad (24)
\end{aligned}
$$

Again use has been made of the fundamental property of the logarithm $\log(x \cdot y) = \log(x) + \log(y)$ which transforms directly into Nonstandard Analysis and leads to simplifications by virtue of equation (22).

As the last example, we will derive a formula for the root function. Even though there is a direct method to compute roots by solving a set of linear equations for the coefficients of the root, we present here a technique based on power series following an approach similar to the exponential and logarithm. The root has the following power series expansion:

$$
\sqrt{1+x} = \sum_{i=0}^{\infty} (-1)^i \frac{1 \cdot 3 \cdot ... \cdot (2i-3)}{2 \cdot 4 \cdot ... \cdot (2i)} \cdot x^i \qquad (25)
$$

Using this formula and the definitions of addition and multiplication (14), (16), one directly obtains for the square root of a differential algebra vector:

$$
\begin{aligned}
& \sqrt{(a_0, a_1, a_2, ..., a_N)} \\
= \; & \sqrt{a_0} \cdot \sqrt{1 + (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})} \\
= \; & \sqrt{a_0} \cdot \sum_{i=0}^{\infty} (-1)^i \frac{1 \cdot 3 \cdot ... \cdot (2i-3)}{2 \cdot 4 \cdot ... \cdot (2i)} \cdot (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})^i \\
= \; & \sqrt{a_0} \cdot \sum_{i=0}^{n} (-1)^i \frac{1 \cdot 3 \cdot ... \cdot (2i-3)}{2 \cdot 4 \cdot ... \cdot (2i)} \cdot (0, \frac{a_1}{a_0}, \frac{a_2}{a_0}, ..., \frac{a_N}{a_0})^i \qquad (26)
\end{aligned}
$$

Using the addition theorems for sine and cosine, one obtains formulas with finite sums in a quite similar way; in general, suppose a function f has an addition theorem of the form

$$f(a + b) = g_a(b) \qquad (27)$$

and $g_a(b)$ can be written in a power series, then by the same reasoning its differential algebraic extension is computable exactly in only finitely many steps. In practice it turns out that this can be done for all commonly supported functions in a FORTRAN computer environment.

# 4 The Implementation of Differential Algebra on a Computer

The arithmetic and the functions of differential algebra can be implemented on a computer for arbitrary order and arbitrary number of variables. As it turns out, this is not easily possible for the field $R^*$ of Nonstandard Analysis; hence we sacrifice the universal existence of multiplicative inverses.

The implementation of the addition and scalar multiplication is trivial. However, the efficient implementation of a multiplication requires some care. First we note that we can increase the speed by defining a slightly different multiplication than in equation (17) with the $c_i$ as

$$c_i = \sum_{\substack{0 \leq \nu, \mu \leq N \\ M_\nu \cdot M_\mu = M_i}} a_\nu \cdot b_\mu \tag{28}$$

This multiplication is the same as in the case of the multiplication of power series. However, in this arithmetic not all power series coefficients of the product are computed; coefficients belonging to terms of orders higher than $n$ are disregarded. For many cases this view of differential algebras as "truncated power series algebras" (see ref. [12]) is sufficient.

The definition of the multiplication (16) requires the knowledge of all possible factorizations of a monomial into two submonomials. The computation of all these factorizations can be quite time consuming. Additionally, in practice it happens frequently that many of the entries in a differential algebra vector are zero.

So it is advantageous to turn the problem inside out so that no factorizations in submonomials are searched, but rather each component of the first vector is multiplied by each component of the second vector and the product is stored at the place where the product monomial belongs.

In order to do that an easy way of finding the address of the product monomial is required. This is done as follows. First, all $N(n, v)$ monomials $M$ are coded with an integer $C$ in the following way: Let $M = x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}$ Then we define $C(M)$ as follows:

$$C(M) = C(x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}) = i_1 \cdot (n+1)^0 + i_2 \cdot (n+1)^1 + \ldots + i_v \cdot (n+1)^{(v-1)} \tag{29}$$

This means that the exponents are just "decimals" in base $(n+1)$. Note that since $i_\nu \leq n$ this representation is injective, i.e. different monomials have different codings. Note also that all codings are always less than $(n+1)^v$, but not all such codings occur.

Now suppose two monomials $M$ and $N$ have to be multiplied and suppose their product has an order less than or equal to $v$. Since the multiplication corresponds to an addition of the exponents, it follows that

$$C(M \cdot N) = C(M) + C(N) \tag{30}$$

To exploit this for the finding of the desired coordinate position $I_M$ (see section 2) of the product of two monomials, an array $D$ is required that has

the property

$$I_M = D(C(M)) \tag{31}$$

This array can be generated easily by the computer. Since the codings are bounded by $(n+1)^v$, the array has to have at least this length. With 6 variables, this enables orders of 8 or 9 if one wants to stay inside the boundaries of computer storage; with 8 variables the order would decrease to about 4 which is too strict a limitation. To circumvent this, a slight modification of the above coding and decoding will be presented.

Without loss of generality, we assume the number of variables $v$ to be even; if it is not even, increase it by one and ignore the additional variable. We define two coding numbers $C_1$ and $C_2$ for any monomial in the following way:

$$\begin{aligned} C_1(x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}) &= i_1 \cdot (n+1)^0 + i_2 \cdot (n+1)^1 + \ldots + i_{\frac{v}{2}} \cdot (n+1)^{(\frac{v}{2}-1)} \\ C_2(x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}) &= i_{\frac{v}{2}+1} \cdot (n+1)^0 + i_{\frac{v}{2}+2} \cdot (n+1)^1 + \ldots + i_v \cdot (n+1)^{(\frac{v}{2}-1)} \end{aligned} \tag{32}$$

Then we store the $N(n,v)$ monomials in the following way. We start with all monomials that have $C_2(M) = 0$ and group them by order; within one order, the monomials are stored according to ascending values of $C_1(M)$. Then we store all those with $C_2(M) = 1$, again by order, and so forth, going through all possible values of $C_2$. Again we obtain

$$\begin{aligned} C_1(M \cdot N) &= C_1(M) + C_1(N) \\ C_2(M \cdot N) &= C_2(M) + C_2(N) \end{aligned} \tag{33}$$

Finally we introduce some "inverse" arrays $D_1$ and $D_2$ in the following way:

$$\begin{aligned} D_1(c_1) &= (I_M \text{ of first monomial } M \text{ with } C_1(M) = c_1) \\ D_2(c_2) &= (I_M \text{ of first monomial } M \text{ with } C_2(M) = c_2) - 1 \end{aligned} \tag{34}$$

Again the arrays $D_1$ and $D_2$ can be generated by the computer.

Using the definitions of $C_1$, $C_2$, $D_1$ and $D_2$ and using the storage scheme outlined above, it now follows that the address of the product of the monomials $M$ and $N$ can be found directly as

$$I_{M \cdot N} = D_1[C_1(I_M) + C_1(I_N)] + D_2[C_2(I_M) + C_2(I_N)] \tag{35}$$

For the sake of clarity, an example for the arrays $c_1$, $c_2$, $d_1$ and $d_2$ is given in appendix 1 for $n = 3$ and $v = 4$. This example also illustrates equations (32) through (35).

The coding defined in (32) entails that the maximum length of the arrays $D_1$ and $D_2$ can now be chosen much lower, namely as $(n+1)^{\frac{v}{2}}$. Assuming a maximum length of 1 million, this entails limitations on the maximum order given a certain number of variables as listed in table 1.

Table 1: The maximum order for different numbers of variables due to a limitation of the length of the reverse addressing arrays $D_1$, $D_2$

| number of variables | 6 | 8 | 10 | 12 |
|---|---|---|---|---|
| maximum order | 99 | 30 | 14 | 10 |

After addition and multiplication are available, the implementation of differential algebra functions is done quite easily using the formulas discussed in section 3.

For practical purposes it is of importance that in the FORTRAN environment differential algebraic operations can only be utilized by calls to subroutines. For this reason a precompiler [13] was developed that allows the use of a new data type "differential algebra" in regular FORTRAN formulas. The precompiler parses the entire program and transforms formulas containing differential algebraic quantities into subroutine calls.

# 5 The Computation of Transfer Maps

## 5.1 An Illustrating Example

Differential Algebras can be used very efficiently to compute the transfer map of equation (1) of particle optical systems in its Taylor series representation.

To illustrate this, let us start the discussion with a very simple example, the midplane motion in a $90^o$ homogenous bending magnet. Let $x_i$ and $a_i = \sin(\alpha_i)$ denote the initial distance and scaled transverse momentum relative to the reference trajectory (see Fig. 1). Then we are interested in the values $x_f$ and $a_f = \sin(\alpha_f)$. Since the trajectories in the magnet are circles, we can readily read from Fig. 1:

$$\begin{aligned}
A &= R\ \sin(\alpha_i) = R\ a_i \\
B &= R\ (1 - \cos(\alpha_i)) + x_i = R\ (1 - \sqrt{1 - a_i^2}) + x_i \\
a_f &= \sin(\alpha_f) = -\frac{B}{R} \\
x_f &= A - R\ (1 - \cos(\alpha_f)) = A - R\ (1 - \sqrt{1 - a_f^2})
\end{aligned} \tag{36}$$

These equations allow the computation of the final coordinates $x_f, a_f$ in terms of the initial coordinates $x_i, a_i$. However, taking these equations and performing all operations in differential algebra allows us to even obtain all derivatives of $x_f$, $a_f$ with respect to $x_i$, $a_i$. These so obtained derivatives, evaluated at $x_i = 0$, $a_i = 0$, are then the expansion coefficients of the map in equation (1). For the sake of clarity, let us explicitly show how $x_f$ and $a_f$ are computed.

Using the ordering in (18) and identifying the variable $a$ with $y$, we obtain using the arithmetic defined in equations (14), (15) and (16)

$$\begin{aligned}
x_i &= (0, 1, 0, 0, 0, 0) \\
a_i &= (0, 0, 1, 0, 0, 0) \\
A &= (0, 0, R, 0, 0, 0) \\
B &= (0, 1, 0, 0, 0, R)
\end{aligned}$$

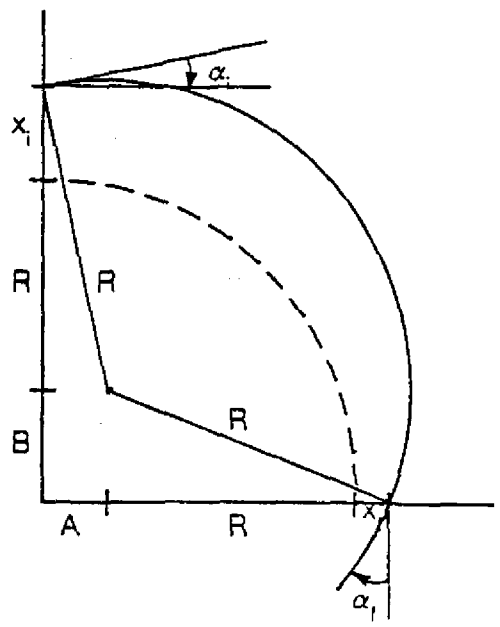



Figure 1: The motion in a 90° homogeneous bending magnet with radius of bend R

$$
\begin{aligned}
a_f &= (0, -\frac{1}{R}, 0, 0, 0, -1) \\
x_f &= (0, 0, R, -\frac{1}{R}, 0, 0)
\end{aligned}
\tag{37}
$$

Comparing the so obtained result with any matrix code [1], [2], [3], [5], we find complete agreement; as an example, the fact that the second component of $x_f$ is zero implies that $\frac{\partial x_f}{\partial x_i} = 0$ and hence (x,x) = 0 (or in TRANSPORT notation $R_{1,1} = 0$) which is a well known property of 90° bends.

In case an additional particle optical element is to follow this bending magnet, one does not have to start all over evaluating this new element at $x_i = (0, 1, 0, 0, 0, 0)$, $a_i = (0, 0, 1, 0, 0, 0)$, but one can start already with $x_f$ and $a_f$ of equation (37). This way one can save the usually quite involved concatenation process and increase performance significantly.

The example discussed in this section has been implemented on the computer. Using the Differential Algebra package, it is easily possible to extract all nonlinearities of this two-dimensional example through order fifty.

## 5.2 Generation of Maps Using Numerical Integration

In this section we will address the general case in which no closed solution of the problem exists. We will see that also in this case we are actually able to compute transfer maps of arbitrary order for arbitrary particle optical elements. Even though we do not have analytical formulas that relate the final coordinates to the initial coordinates, there is still a way to computationally relate the final coordinates to the initial coordinates, by numerical integration of the equations of motion.

In this case, the final coordinates are still computed from the initial coordinates using standard arithmetic and functions, however the relations are more complex than in the case of the homogeneous sector.

Now blindfoldedly performing all these operations in differential algebra automatically leads to all desired derivatives of the transfer function, regardless of the form of the equations of motion.

Differential algebraic techniques have been implemented in the program COSY [5]. They allow the computation of transfer maps of elements with a dependence on the independent variable for which an analytic solution cannot be obtained from HAMILTON [4] like fringing fields. Using an eighth order Runge Kutta integrator, all operations required for a tracking of particles are performed in differential algebra. This allows the computation of arbitrary fringing field effects as soon as the spacial distribution of the electromagnetic fields is known [14].

In many cases, including the proposed SSC [17], the particle optical system can be represented very well by a sequence of kicks as in TEAPOT [15]. The sequence of kicks can be viewed as a symplectic integrator of second order in the independent variable. In this case, the execution of all arithmetic in differential algebra is particularly easy. It may be used for very efficient and accurate computation of tune shifts and chromaticities using normal form theory [6], [7]. This will be discussed in detail in [16].

It is worth mentioning that TEAPOT and COSY are in complete agreement for all nonlinearities through fifth order. In order to show this, we selected a small ring for which bends and quadrupoles of finite length had to be split into several hundred drifts and kicks in order to be computable by TEAPOT. For the actual example and the reason for the ring selection see reference (15).

## 5.3 Hamiltonian Theory

In this subsection we will outline the usefulness of differential algebras for Hamiltonian systems. One of the most fundamental concepts of Hamiltonian theory is the Poisson bracket between two functions of phase space. This requires the differentiation with respect to phase space variables.

Suppose a differential algebra vector is given. Then it can be viewed as a descriptor of a function, giving its value and derivatives at a certain point. In this context the required differentiation is just a "bookkeeping operation" moving derivatives in the vector to different places. Thus all the required elements for a Poisson bracket are available. In fact, a Poisson bracket for arbitrary order and arbitrary number of variables is contained in the differential algebraic package written by the author.

Having the Poisson bracket, Lie operators, "Poisson brackets waiting to happen", can be computed. In fact, as soon as the generator $\vec{v}$ of the Lie operator $:\vec{v}:$ vanishes at the origin and has zero first derivatives, the process is closed in that no feeddown from higher orders occurs.

Using Lie operators, the transfer map or flow of a Hamiltonian system can be computed as

$$\mathcal{M} = \exp(-t : H :) \tag{38}$$

where H is the Hamiltonian of the system. Note that by the proper choice of the coordinates it can always be achieved that $H(\vec{0}) = 0$ and also the first derivatives of H vanish. This entails that each summand in equation (38) can be computed in a closed fashion.

Furthermore, it turns out it suffices to carry only finitely many terms of the sum in (38) if we want the nth order coefficients of the transfer map to a certain accuracy. To see this, we show that the norm of the Lie operator $: -tH :$ is bounded. First note that in order to obtain the transfer map through order $n$, it suffices to know $H$ to order $n+1$.

We define a norm on a Polynomial of phase space variables in the following way

$$|\sum a_{i_1 i_2 \ldots i_v} \cdot x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}| = \sum |a_{i_1 i_2 \ldots i_v}| \tag{39}$$

It is quite easy to show that for arbitrary Polynomials $P_1$, $P_2$ we have

$|P_1 \cdot P_2| \leq |P_1| \cdot |P_2|$. Since a derivative only moves coefficients around in the vector and drops some, it follows for a Polynomial f that

$$|\frac{\partial f}{\partial x_i}| \leq |f| \tag{40}$$

and thus we obtain for the norm of the Lie operator consisting of the two derivatives in each summand:

$$| : -tH : | = sup\frac{| - t : H : f|}{|f|} = | : -tH : | \cdot v \tag{41}$$

This requires that the sum $\sum_{\nu=0}^{\infty} \frac{(-t:H:)^\nu}{\nu!}$ converge (in a differential algebra sense!), and that it is possible to obtain an estimate as to how many terms have to be taken in order to obtain a certain accuracy. In general it turns out that these estimates are even usually quite pessimistic and the terms in the sum become small more quickly.

This method to obtain the Hamiltonian flow in a computer code has the additional advantage that it does not require any composition of the maps of two subsystems; all that is required is to compute H by substituting in the $\vec{z}$ already obtained and then let this differential algebraic vector act on $\vec{z}$.

In practice it turns out that the savings from the avoided compositions is more significant than the additional time required for the iterative evaluation of the exponential sum exp(: $-tH$ :). In fact, a composition routine requires $N(n, v)$ differential algebra multiplications, whereas one term in the sum requires $v$ multiplications per dimension.

In the case the Hamiltonian is time dependent, the above reasoning requires a slight modification. In this case it is in general not possible to go through the time dependent element in one step. However, it is possible to derive an explicit high order numerical integrator in the following way. First we note that for any function of phase space, we have

$$\frac{df}{dt} = [f, H] + \frac{\partial}{\partial t} f = \mathcal{O} f \tag{42}$$

In our particular case, we are especially interested in the cases in which f are the components of $\vec{z}$. Iterating equation (42), we can obtain higher order derivatives of f, say up to order $u$. First note that even for the

f's of interest in, $\frac{\partial}{\partial t} f$ vanishes, but in the higher derivatives such partial derivatives prevail because H is time dependent.

In order to guarantee that all derivatives of the phase space variables can be computed to order $u$, H must be known to order $v + u$. However, terms with a higher power of t than $u$ can be set to zero. In order to compute the u derivatives of the phase space vector, which is needed for the $uth$ order integrator, a total of $6 \cdot u$ Poisson brackets are required. Considering that in many practical cases, an element can be transversed in one or very few steps of an eighth order integrator and that like above we still save the composition process, the effort is still very favorable to the situations in which compositions are required.

**Acknowledgements**

For financial support I want to thank the Deutsche Forschungsgemeinschaft, Dr. Alex Chao of the SSC Central Design Group and Dr. Edward Heighway of Los Alamos National Laboratory. For fruitful discussions I want to thank Dr. Etienne Forest and Professor Dr. Hermann Wollnik.

# 6 References


1. K.L Brown, D. C. Carey, Ch. Iselin and F. Rothacker, SLAC 91 (1973 rev.), NAL 91, and CERN 80-04
2. A.J. Dragt et. al., IEEE Trans. Nucl. Sci., NS-32, 2311 (1985)
3. H. Wollnik, J. Brezina, M. Berz and W. Wendel, Proc. AMCO-7, GSI-Rep. THD-26 (1984) p.679
4. M. Berz and H. Wollnik, Nucl. Instr. Meth. A258 (1987) 364
5. M. Berz, H. Wollnik and H. C. Hoffmann, Nucl. Instr. Meth. A 258 (1987) 402
6. E. Forest, Particle Accelerators 22 (1987) 15
7. E. Forest, SSC Report SSC-111, Berkeley
8. A. Robinson, Proc. Royal Acad. Amsterdam Ser. A, 64, 432-440, 1961
9. D. Laugwitz, JBer. Deutsch. Math. Vereinigung 75 (1973) 66
10. I. Niven, Am. Math. Monthly 76-8 (1969) 871
11. L.B. Rall, Math. Magazine 59 (1986) 275
12. M. Berz, Nucl. Instr. Meth. A258 (1987) 431 ; M. Berz, Los Alamos Accelerator Theory Note AT-6:ATN-86-16 (1986)
13. M. Berz, Los Alamos Report AT-3:TN-87-32 (1987)
14. M. Berz, Dissertation University of Giessen, 1986 (unpublished, in German); B. Hartmann, Diplomarbeit University of Giessen 1987 (unpublished, in German)
15. L. Schachinger and R. Talman, Particle Accelerators 22 (1987) 35
16. E. Forest, M. Berz and J. Irwin, this issue of Particle Accelerators
17. SSC Conceptual Design Report, SSC Note SSC-SR 2020, Berkeley

# 7 Appendix

List of the ordering of the all monomials $M = x_1^{i_1} \cdot \ldots \cdot x_v^{i_v}$ for order $n = 3$ and number of variables $v = 4$. Also shown are the coding integers $C_1$ and $C_2$ (cf section 4)

| $I_M$ | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $C_1$ | $C_2$ |
|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 1 | 0 | 0 | 4 | 0 |
| 4 | 2 | 0 | 0 | 0 | 2 | 0 |
| 5 | 1 | 1 | 0 | 0 | 5 | 0 |
| 6 | 0 | 2 | 0 | 0 | 8 | 0 |
| 7 | 3 | 0 | 0 | 0 | 3 | 0 |
| 8 | 2 | 1 | 0 | 0 | 6 | 0 |
| 9 | 1 | 2 | 0 | 0 | 9 | 0 |
| 10 | 0 | 3 | 0 | 0 | 12 | 0 |
| 11 | 0 | 0 | 1 | 0 | 0 | 1 |
| 12 | 1 | 0 | 1 | 0 | 1 | 1 |
| 13 | 0 | 1 | 1 | 0 | 4 | 1 |
| 14 | 2 | 0 | 1 | 0 | 2 | 1 |
| 15 | 1 | 1 | 1 | 0 | 5 | 1 |
| 16 | 0 | 2 | 1 | 0 | 8 | 1 |
| 17 | 0 | 0 | 0 | 1 | 0 | 4 |
| 18 | 1 | 0 | 0 | 1 | 1 | 4 |
| 19 | 0 | 1 | 0 | 1 | 4 | 4 |
| 20 | 2 | 0 | 0 | 1 | 2 | 4 |
| 21 | 1 | 1 | 0 | 1 | 5 | 4 |
| 22 | 0 | 2 | 0 | 1 | 8 | 4 |
| 23 | 0 | 0 | 2 | 0 | 0 | 2 |
| 24 | 1 | 0 | 2 | 0 | 1 | 2 |
| 25 | 0 | 1 | 2 | 0 | 4 | 2 |
| 26 | 0 | 0 | 1 | 1 | 0 | 5 |
| 27 | 1 | 0 | 1 | 1 | 1 | 5 |
| 28 | 0 | 1 | 1 | 1 | 4 | 5 |
| 29 | 0 | 0 | 0 | 2 | 0 | 8 |
| 30 | 1 | 0 | 0 | 2 | 1 | 8 |
| 31 | 0 | 1 | 0 | 2 | 4 | 8 |
| 32 | 0 | 0 | 3 | 0 | 0 | 3 |
| 33 | 0 | 0 | 2 | 1 | 0 | 6 |
| 34 | 0 | 0 | 1 | 2 | 0 | 9 |
| 35 | 0 | 0 | 0 | 3 | 0 | 12 |

List of the arrays $D_1$ and $D_2$ for order $n = 3$ and number of variables $v = 4$. For all M, one has $I_M = D_1(C_1(M)) + D_2(C_2(M))$ (cf section 4)

| $j$ | $D_1(j)$ | $D_2(j)$ |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 2 | 10 |
| 2 | 4 | 22 |
| 3 | 7 | 31 |
| 4 | 3 | 16 |
| 5 | 5 | 25 |
| 6 | 8 | 32 |
| 7 | 0 | 0 |
| 8 | 6 | 28 |
| 9 | 9 | 33 |
| 10 | 0 | 0 |
| 11 | 0 | 0 |
| 12 | 10 | 34 |

# Part II
# Normal Form Methods For Complicated Periodic Systems: A Complete Solution Using Differential Algebra and Lie Operators

ETIENNE FOREST
Exploratory Studies Group
Accelerator Fusion Research Division
Lawrence Berkeley Laboratory
and
SSC Central Design Group
Universities Research Association
c/o Lawrence Berkeley Laboratory
Berkeley, CA 94720

MARTIN BERZ AND JOHN IRWIN
SSC Central Design Group
Universities Research Association
c/o Lawrence Berkeley Laboratory
Berkeley, CA 94720

March 1988

**Abstract**

We present two types of formal algorithms: an order by order and a so-called "superconvergent" procedure, bringing the one period map into a so-called normal form, which displays the harmonic content of the map. The algorithm is arbitrary in order, in number of parameters and phase space dimensions, and covers the range of signatures of the unperturbed quadratic invariants found in circular machine dynamics. The normal form and the map extraction algorithms have all been implemented using the differential algebra software. In fact, the work of this paper is feasible in toto because of the differential algebra theory and software.

## 1. Introduction

Serious simulations of large and small circular (or periodic) machines require the use of tracking codes. For example, in the study of the proposed Superconducting Supercollider (SSC), these codes must be able to treat misaligned, mispowered and misconstructed magnets with substantial random errors. In addition, the tracking codes must be equipped with closed orbit correction schemes, random and systematic multipole correctors and even sorting algorithms. At the end, the full horrendous lattice is ready for tracking. In the case of the SSC, we are dealing with a lattice file specifying approximately $10^5$ multipoles of different orders and symmetries. But what is the lattice file of a tracking code? In fact, the lattice file specifies the horrible *realistic* Hamiltonian of the SSC.

One would like to find an exact recursive algorithm, implementing canonical transformations, which eliminates non-secular terms in the motion of particles, a scheme that would work for an arbitrarily complex Hamiltonian, and to an order limited only by the power of the present day computers, and would treat problems in a phase space and a parameter space of arbitrary dimension, where again the limits depend only upon the power of our computers.

Our approach rests upon the premise that:

*i*) A process which first extracts the formal power series map for one period from the complex tracking code and then analyses it, is more modular and removes the need for a local $s$-dependent perturbation theory.

*ii*) The production of the map and the analysis of the map should be independent procedures. In particular, the extraction of the map should not have to involve canonical variables.

*iii*) An adequate and desirable analysis of the final map, is best achieved by ex-

pressing it in terms of Lie generators.

For *linear effects*, statements i) to iii) have been common place in accelerator physics since the early work of Courant and Snyder [1]. Indeed, the Twiss parameters and the fractional tunes can be extracted from the one-turn matrix. Similar quantities are available for coupled systems. It is also true that one propagates the Twiss parameters by matrix (i.e., map) manipulations. Incidently, the Courant-Snyder invariant is just a multiple of the Lie generator of the linear map produced by the matrix! Hence it became clear to one of the authors (E.F.) that for maximum efficiency in the treatment of large and messy periodic systems one should extend the matrix manipulations to non-linear maps, and rewrite the whole of perturbation theory in a Hamiltonian-free context [2]. For the relatively simple Hamiltonians used in the design of circular machine, Forest has used the concatenator and the analysis tools of the program MARYLIE [3]to perform exact extraction and analysis of fifth order maps in $(x, p_x, y, p_y, \tau, p_\tau)$. (The tracking codes used were TEAPOT and THINTRAC) [4].

The reader will notice that fifth order and six variables falls short of the goals set forth in this paper. Fortunately, Berz, described a tool in the preceeding paper of this issue, which, in the views of his co-authors will supersede previous techniques of numerical differentiation implemented on computers. While working in the field of particle spectrometers, he realized the importance of the extraction of formal power series for arbitrarily complex optical system (or in other words extract high order aberrations). Numerical differentiation techniques do not allow for high order computations because the precision decays rapidly with the order of the aberrations. This lead Berz to investigate the differential algebra (DA) (an application of non-standard analysis [5, 6]) and produce a DA-software package which allows the user to compute all the partial derivatives of any computed quantity with respect to any

number of variables. As discussed in the preceeding paper, this implies that one can always produce a formal power series map of any system provided a tracking code exists. (Of course, more than one exists!)

In this paper, we will present the tools which are needed to perform the normal form algorithm on the map, using the mathematical operations available in the differential algebra software of Berz. All the operations are formal in the sense that we restrict ourselves to truncated power series i.e. derivatives up to a certain order and do not discuss questions of convergence.

This combination of extraction of mappings and the analysis software provides a definitive tool for perturbation theory on complicated periodic system for it is only limited by the power of the present day computers, not by human lack of ability and stamina.

*A Short Mathematical Survey of Symplectic Maps*

In this short survey we will introduce the maps to be studied in the later sections of this paper. Initially, we take some care to emphasize the difference between a function "f" and the value "f(x)" the function may take once it is evaluated on a given element "x". This is important because the Lie algebraic maps of classical mechanics act on functions, not on the rays.

Let us assume that we have a 2N-dimensional phase space. A vector $\mathbf{x}$ in this space will be of the form:

$$\mathbf{x} = (q_1, p_1, \ldots, q_N, p_N) \tag{1.1a}$$

where $\mathbf{q}$ are the positions and $\mathbf{p}$ the momenta. Being canonical variables they obey the famous Poisson bracket condition

$$[q_i, p_j] = \delta_{ij} \,. \tag{1.1b}$$

Let us assume that we start with a transformation $\mathcal{G}$ which maps linearly the set $\mathcal{V}$ of functions of phase space and parameter space into itself i.e. it is an endomorphism of $\mathcal{V}$:

$$f \in \mathcal{V} \xrightarrow{\mathcal{G}} \mathcal{G}f \in \mathcal{V}\,;\ \mathcal{G} \in End(\mathcal{V}) \tag{1.2a}$$

$$\mathbf{z} \in \mathbb{R}^{2N} \xrightarrow{f} f(\mathbf{z}) \in \mathbb{R} \tag{1.2b}$$

$$\mathbf{z} = (\mathbf{x}, \boldsymbol{\delta}); \quad \boldsymbol{\delta} = \text{vector of } N_P \text{ parameters}. \tag{1.2c}$$

Let us define 2N projection functions:

$$\mathcal{V} \xrightarrow{\Pi_i} \mathbb{R};\ \ \Pi_i \in \mathcal{V}$$

$$\Pi_i(\mathbf{z}) = z_i\,. \tag{1.2d}$$

Let us assume that $\mathcal{G}$ leaves the Poisson bracket invariant $[f, g]$, i.e. $[f, g] = [\mathcal{G}f, \mathcal{G}g]$. Then we call $\mathcal{G}$ a symplectic map. For a symplectic map such as $\mathcal{G}$, the following can be shown [7]:

$$(\mathcal{G}f)(\mathbf{z}) = f(\tilde{\mathbf{z}})$$

$$\tilde{\mathbf{z}} = ((\mathcal{G}\Pi_1)(\mathbf{z}), ..., (\mathcal{G}\Pi_{2N})(\mathbf{z}))\,. \tag{1.3}$$

Assuming that $\mathcal{G}$ represents the system under study, the function $\mathcal{G}\Pi_i$ ($\in \mathcal{V}$) gives the $i^{th}$ component of the final ray as a function of the initial ray $\mathbf{z}$. To abbreviate the notation, $\tilde{\mathbf{z}}$ will be denoted by $\mathcal{G}\mathbf{z}$ and $\tilde{z}_i$ by $\mathcal{G}z_i$. We consider endomorphisms of $\mathcal{V}$ such as $\mathcal{G}$ because exponentials of Lie operators provide an infinite supply of maps of the type of $\mathcal{G}$. As we will see, Lie operators are essential if symplectic maps are to be put in normal form. Clearly, as we just mentioned, if $\mathcal{G}$ is the map for

one period of the system, $\mathcal{G}\mathbf{z}$ is the ray after one period. This is exactly what the DA-software gives us to any pre-specified order $N_o - 1$:

$$\mathcal{G}\mathbf{z} = (\mathcal{G}\mathbf{x}, \delta) \tag{1.4a}$$

$$\mathcal{G}\mathbf{z} = M\mathbf{z} + \sum_{k=1}^{N_o-1} \mathbf{\Gamma}_k(\mathbf{z}). \tag{1.4b}$$

$$M = \text{Matrix or linear part}.$$

$$\mathbf{\Gamma}_k(\mathbf{z}) = \text{homogeneous polynomial function of order } k.$$

$$N_o - 1 = \text{maximum order of the formal power series considered..} \tag{1.4c}$$

Notice that in general it can be arranged that $\mathcal{G}(\mathbf{0}) = \mathbf{0}$. In the next section, two other properties will be assumed:

1) There exists a DA-function $\mathcal{F}\mathbf{z}$ ( $\mathcal{F} \in End(\mathcal{V})$) which brings the rays to the $\delta$-dependent fixed point of $\mathcal{G}$:

$$\mathcal{G}^f\mathbf{z} = (\mathcal{F}^{-1}\mathbf{z}) \circ (\mathcal{G}\mathbf{z}) \circ (\mathcal{F}\mathbf{z}) \tag{1.5}$$

or mathematically $\mathcal{G}^f$ has the property:

$$\mathcal{G}^f(\mathbf{x}, \delta) = (\mathbf{0}, \delta) + M^f_{2N}(\mathbf{x}, \mathbf{0}) + \sum_{k=1}^{N_o-1} \mathbf{\Gamma}^f_k(\mathbf{x}, \delta) \tag{1.6a}$$

$$\mathcal{G}^f(\mathbf{0}, \delta) = (\mathbf{0}, \delta) \quad \forall \delta. \tag{1.6b}$$

Here "o" is the composition of two functions. It is available in the DA-software package.

2) We will also assume that there exists an endomorphism of $\mathcal{V}$ denoted by $\mathcal{A}_L$ which leaves the sub-space of linear functions globally invariant and brings $M_{2N}^f$ into a very simple form [8]:

$$\mathcal{M}\mathbf{z} = (\mathcal{A}_L^{-1}\mathbf{z}) \circ (\mathcal{G}^f\mathbf{z}) \circ (\mathcal{A}_L\mathbf{z}) \tag{1.7a}$$

$$\mathcal{M}\mathbf{z} = (0, \delta) + R\mathbf{x} + \sum_{k=1}^{N_o-1} \Gamma_k^o(\mathbf{x}, \delta) \tag{1.7b}$$

In Eq. (1.7), every $(q_i, p_i)$ undergoes either stable or "unstable" rotation under the effect of $R$. Using the script letter to represent the symplectic map associated to $R\mathbf{x}$:

$$\mathcal{V} \xrightarrow{\mathcal{R}} \mathcal{V} \tag{1.8a}$$

$$\mathcal{R}\begin{pmatrix} q_i \\ p_i \end{pmatrix} = \begin{pmatrix} \cos\mu_i & -\sin\mu_i \\ \sin\mu_i & \cos\mu_i \end{pmatrix} \begin{pmatrix} q_i \\ p_i \end{pmatrix} \tag{1.8b}$$

or,

$$= \begin{pmatrix} \cosh\mu_i & -\sinh\mu_i \\ -\sinh\mu_i & \cosh\mu_i \end{pmatrix} \begin{pmatrix} q_i \\ p_i \end{pmatrix} \tag{1.8c}$$

The reader can glance at Eq. (3.1b) for the Lie generator equivalent of Eq. (1.8). It is worth pointing out that $\mathcal{F}\mathbf{z}$ will exist if all the $\mu_i$'s are different from zero. Indeed, in this paper we do not consider integer tunes.

The computation of $\mathcal{F}\mathbf{z}$ and the transformation of the matrix $M_{2N}^f$ into $R$ are all operations which have been implemented for an arbitrary phase space dimension in the DA-software.

Therefore, in the body of this paper, we will deal exclusively with the map $\mathcal{M}$ and its DA-representation $\mathcal{M}\mathbf{z}$. It is on this map that the full combined power of the Lie algebraic methods on the symplectic group and the tools of the differential algebra package permit a general solution of the normal form problem.

Before going further, the reader is assumed to have a strong knowledge of Lie operators in classical mechanics. In particular, we will often go back and forth between functions belonging to $\mathcal{V}$ and their associated Lie operators belonging to $End(\mathcal{V})$. The notation of Dragt is used throughout this paper:

$$: f : g = \{f, g\} = \nabla f^{\dagger} \, J \, \nabla g \tag{1.9a}$$

$$J \;=\; \text{symplectic form}. \tag{1.9b}$$

It is worth pointing out that the commutator of two Lie operators :f: and :g: is just the Lie operator of their Poisson bracket :[f,g]:. This is a very important homomorphism of the commutator algebra of Lie operators and the Poisson bracket algebra of functions. It is used implicitly in many manipulations used throughout this paper.

In Section 2, we introduce and justify the existence of a map denoted by $\mathcal{T}$ critical to the normal form algorithm on a factored Lie representation of the map. In Section 3, we introduce a new basis for the functions of phase space which uses the linear eigenfunctions of $\mathcal{T}$ (or $\mathcal{R}$). In Section 4, we discuss the evaluation of $(\mathcal{T}^{-1} f_r)(\mathbf{x})$ in terms of the differential algebra tools.

In Section 5, we extract a Lie generator from a DA-function of the form of (1.7b) again using DA-tools. This is then used in Section 6, were the full recursive algorithm is displayed.

To simplify the notation, the parameter $\delta$ is left out of the subsequent discussion until Appendix A. Hence $\mathbf{z}$ is replaced by $\mathbf{x}$. The vector $\mathbf{z}$ can be put back in by mentally letting all operators leave $\delta$ unchanged.

Finally, the appendices discuss the "coasting" plane normalization, the super-convergent procedure and the normalization of the pseudo-Hamiltonian of the map. (Not a new topic, except for its implementation in the differential algebra context).

## 2. Canonical Transformations on Symplectic Endomorphisms of $\mathcal{V}$

The normal form algorithms of Deprit [9, 10, 11, 12, 13]proceed by successive transformations on the Hamiltonian. The infinitesimal Lie generator of the map between $s$ and $s + ds$ is just :-H: . In our case, we do not work on a map near the identity but on a map far from it since it represents one period of our system.

Let us assume that our map $\mathcal{M}$ is made out of two pieces, an unperturbed linear part $\mathcal{R}$ and a perturbation proportional to a smallness parameter $\alpha$. Following Dragt and Finn [7], it is best to factor the total map into two pieces:

$$\mathcal{M} = \mathcal{R} \exp(: \alpha f :) \tag{2.1}$$

Consider a canonical transformation $\mathcal{A}$ whose purpose is to modify $\mathcal{M}$ into a new factorized representation $\mathcal{N}$ defined to first order in $\alpha$. Using a Lie representation for $\mathcal{A}$, we get for $\mathcal{N}$

$$\begin{aligned}
\mathcal{N} &= \mathcal{A}\,\mathcal{M}\,\mathcal{A}^{-1} \\
&= \exp(: \alpha F :)\mathcal{R} \exp(: \alpha f :) \exp(: -\alpha F :) \\
&= \mathcal{R} \exp(: \alpha \mathcal{R}^{-1} F :) \exp(: \alpha f :) \exp(: -\alpha F :) \\
&= \mathcal{R} \exp\Big( : \alpha\{-(\mathcal{E} - \mathcal{R}^{-1})F + f\} + O(\alpha^2) : \Big) \\
&\qquad (\mathcal{E} = \text{identity map})
\end{aligned} \tag{2.2}$$

Denoting by $\mathcal{T}$ the operator $\mathcal{E} - \mathcal{R}^{-1}$, it is clear from Eq. (2.2) that one must study the range and the kernel of $\mathcal{T}$ in order to specify what possible linear terms in $\alpha$

can remain in Eq. (2.2) [14, 15]. Suppose $f$ is decomposed as follows:

$$f = f_r + f_o \quad ; \quad f_r \perp Ker\,\mathcal{T} \tag{2.3}$$

Then, we can select $\mathcal{A}$ or $F$ such that $\mathcal{N}$ becomes $\exp\Big(: \alpha f_o + O(\alpha^2) :\Big)$. The function $F$ is just given by:

$$F = \mathcal{T}^{-1} f_r \,. \tag{2.4}$$

From this short discussion one sees the central importance of the map $\mathcal{R}$. The eigenvectors of $\mathcal{R}$ of unit eigenvalue will constitute the kernel $Ker\ \mathcal{T}$ so critical to the inversion of $\mathcal{T}$.

In the next section, we examine a suitable eigenbasis for the study of $\mathcal{R}$ in the general case.

## 3. The Eigenfunctions of $\mathcal{R}$ and the Resonance Basis

As discussed earlier we assume that the linear map $\mathcal{R}$ has been brought to the following Lie representation:

$$\mathcal{R} = \exp(: f_2 :)\,, \tag{3.1a}$$

$$f_2 = \sum_{k=1}^{N} -\frac{\mu_k}{2}\,(q_k^2 + (\epsilon_k - \bar{\epsilon}_k)p_k^2) = \sum_{k=1}^{N} f_2^k \tag{3.1b}$$

$$\begin{cases} \epsilon_k = 1, \bar{\epsilon}_k = 0 \quad ; & \text{for stable motion in } k^{th} \text{ plane.} \\ \epsilon_k = 0, \bar{\epsilon}_k = 1 \quad ; & \text{for unstable motion in } k^{th} \text{ plane.} \end{cases} \tag{3.1c}$$

In Eq. (3.1), we purposely neglect the "coasting" plane which is obtained when $\epsilon_k = \bar{\epsilon}_k = 0$. Its inclusion would complicate the discussion, since it is not true anymore that the vector space of polynomial functions is a direct sum of the range $Im\mathcal{T}$ and the kernel $Ker\ \mathcal{T}$ [16](see Appendix A).

The evaluation of $\mathcal{T}^{-1} f_r$ requires a decomposition of $f_r$ in eigenvectors of $: f_2 :$. These eigenvectors are easy to obtain, the answer is given by:

$$: f_2^k : h_k^{\pm} = \mp (i\epsilon_k + \bar{\epsilon}_k)\mu_k \, h_k^{\pm} = \mp \lambda_k h_k^{\pm} \tag{3.2a}$$

$$h_k^{\pm} = q_k \pm (i\epsilon_k + \bar{\epsilon}_k) p_k \tag{3.2b}$$

$$: f_2^k := - \frac{\mu_k}{2} \, h_k^+ h_k^- \, . \tag{3.2c}$$

Using this new basis, we can easily find the kernel $Ker\ \mathcal{T}$. Let us define a new vector as follows:

$$|\mathbf{m}, \mathbf{n}\rangle = (h_1^+)^{m_1} (h_1^-)^{n_1} \ldots (h_N^+)^{m_N} (h_N^-)^{n_N} \, . \tag{3.3}$$

Using the differential property of the operator $: f_2 :$, we can compute the eigenvalue of $|\mathbf{m}, \mathbf{n}\rangle$ [11]:

$$: f_2 : |\mathbf{m}, \mathbf{n}\rangle \ = \ (\mathbf{n} - \mathbf{m}) \cdot \boldsymbol{\lambda} \, |\mathbf{m}, \mathbf{n}\rangle \, . \tag{3.4}$$

Assuming that the $\lambda_k$s are irrational and prime amongst themselves, we conclude that

$$|\mathbf{m}, \mathbf{n}\rangle \ \in \ Ker\, \mathcal{T} \ \Longrightarrow \ \mathbf{n} - \mathbf{m} \ = \ \mathbf{0} \, . \tag{3.5}$$

Providing that one can easily change basis to the $|\mathbf{m}, \mathbf{n}\rangle$, the computation of $\mathcal{T}^{-1} f_r$ is trivial [14, 15]:

$$f_r = \sum_{\mathbf{m},\mathbf{n}} A_{\mathbf{m},\mathbf{n}} \, |\mathbf{m}, \mathbf{n}\rangle \tag{3.6a}$$

$$\mathcal{T}^{-1} f_r = \sum_{\mathbf{m},\mathbf{n}} \frac{A_{\mathbf{m},\mathbf{n}}}{1 - \exp\Big( (\mathbf{m} - \mathbf{n}) \cdot \boldsymbol{\lambda} \Big)} \, |\mathbf{m}, \mathbf{n}\rangle \, . \tag{3.6b}$$

In the next section, we show how to perform the change of basis and the computation of $\mathcal{T}^{-1}$ using the differential algebra composition and the special exponent

dependent scalar multiplication.

## 4. The Evaluation of $\mathcal{T}^{-1} f_r$

The symplectic maps which we study have real Lie generators. However, we have seen that the evaluation of $\mathcal{T}^{-1} f_r$ requires the introduction of complex eigenvectors if one or more planes undergo stable oscillations under the action of $\mathcal{R}$. We will describe here how one uses the composition of two differential algebra maps to avoid dealing with complex vectors.

First, we introduce the special exponent dependent scalar multiplication available to the user of the DA-software. Consider a function $f$ of $\mathbb{R}^{2N}$ into $\mathbb{R}$

$$f = \sum A_{\mathbf{j}}\, x_1^{j_1} \ldots x_{2N}^{j_{2N}} \tag{4.1}$$

and an arbitrary function $g(\mathbf{j})$ from $\mathbb{N}^{2N}$ to $\mathbb{R}$. Using $g(\mathbf{j})$, we built a new function $\phi_g f$ as follows:

$$\mathbb{R}^{2N} \xrightarrow{\phi_g f} \mathbb{R}$$

$$\phi_g f = \sum g(\mathbf{j})\, A_{\mathbf{j}}\, x_1^{j_1} \ldots x_{2N}^{j_{2N}}\,. \tag{4.2}$$

The transformation $\phi_g$, like the Lie map $\mathcal{M}$ acts on the space of phase space functions of $\mathbb{R}^{2N}$ into $\mathbb{R}$.

We now introduce a transformation $\phi_\pi$ by defining $\pi(\mathbf{j})$ to be:

$$\begin{cases} M = \sum_{i=1}^{N} \epsilon_i\, j_{2i} = & \text{number of powers of momenta} \\ & \qquad \text{in the stable planes} \\ \pi(\mathbf{j}) = (-1)^{\frac{M}{2}}; & \qquad \text{even } M \\ \pi(\mathbf{j}) = (-1)^{\frac{M+1}{2}}; & \qquad \text{odd } M \end{cases} \tag{4.3}$$

In the definition of $\pi(\mathbf{j})$, the function $\epsilon_i$ is the stable plane identifier of Eq. (3.1).

Furthermore, we introduce two functions which will serve in the computation of $\mathcal{T}^{-1}$. Using Eq. (3.6b), we define the functions $a(\mathbf{j})$ and $b(\mathbf{j})$ to be respectively the real and imaginary part of the following function:

$$a + ib \;=\; \frac{1}{1 - \exp\Big((\mathbf{m} - \mathbf{n}) \cdot \boldsymbol{\lambda}\Big)} \quad ; \quad \mathbf{j} = (m_1, n_1, \ldots m_N, n_N)\,. \tag{4.4}$$

The explicit formulae for the functions $a$ and $b$ are:

$$a(\mathbf{j}) = \frac{1 - \cos\Delta e^{\overline{\Delta}}}{1 + e^{2\overline{\Delta}} - 2e^{\overline{\Delta}}\cos\Delta} \tag{4.5a}$$

$$b(\mathbf{j}) = \frac{\sin\Delta e^{\overline{\Delta}}}{1 + e^{2\overline{\Delta}} - 2e^{\overline{\Delta}}\cos\Delta} \tag{4.5b}$$

$$\begin{cases} \Delta & = \sum \epsilon_i (m_i - n_i)\mu_i\,, \\ \overline{\Delta} & = \sum \bar{\epsilon}_i (m_i - n_i)\mu_i\,. \end{cases} \tag{4.5c}$$

Clearly, two more exponent dependent scalar multiplications can be defined, namely $\phi_a$ and $\phi_b$.

To complete the battery of tools necessary for the evaluation of $\mathcal{T}^{-1} f_r$, we will define three real change of basis using the composition. Consider two functions $\Re$ and $\Im$ belonging to $\mathcal{V}$ :

$$\begin{aligned} \mathbb{R}^{2N} &\xrightarrow{\Re} \mathbb{R}^{2N}\,, \\ \mathbb{R}^{2N} &\xrightarrow{\Im} \mathbb{R}^{2N} \end{aligned} \tag{4.6}$$

One can always define the function $\Im \circ \Re$ from $\mathbb{R}^{2N}$ to $\mathbb{R}^{2N}$ as follows:

$$\begin{aligned} (\Im \circ \Re)(\mathbf{x}) &= \Im\Big(\Re(\mathbf{x})\Big) \\ \mathbf{x} &\in \mathbb{R}^{2N} \end{aligned} \tag{4.7}$$

The DA-composition allows for operations of the type of (4.7) whenever $\Re$ and $\Im$ are polynomial functions. We now define three transformations from $\mathbb{R}^{2N}$ to $\mathbb{R}^{2N}$ by their effect on the basis components $q_k$ and $p_k$:

$$\begin{cases} \Re_r(\mathbf{x})_{|2k} = \Re_r q_k = \frac{q_k+p_k}{2}\,;\ \mathbf{x} = (q_1, p_1, \ldots, q_N, p_N)\,, \\ \Re_r(\mathbf{x})_{|2k-1} = \Re_r p_k = \frac{q_k-p_k}{2}\,; \end{cases} \tag{4.8a}$$

$$\begin{cases} \Im q_k &= \bar{\epsilon}_k q_k + \epsilon_k p_k \\ \Im p_k &= \bar{\epsilon}_k p_k + \epsilon_k q_k\,, \end{cases} \tag{4.8b}$$

$$\begin{cases} \Re_r^{-1} q_k &= q_k + p_k \\ \Re_r^{-1} p_k &= q_k - p_k\,. \end{cases} \tag{4.8c}$$

Strictly speaking the DA-composition computes $\Im \circ \Re$ in a formal sense only since it truncates at a predetermined order. Here however, since $\Re_r$, $\Im$ and $\Re_r^{-1}$ are linear transformations, the composition will be exactly performed. We are now in a position to compute $\mathcal{T}^{-1} f_r$ in terms of the operations described in this section.

Before giving the result for $\mathcal{T}^{-1} f_r$ in terms of the real number transformations listed in this section, let us write the result for $\mathcal{T}^{-1} f_r$ using complex number transformations. Using Eq. (3.2b), we define the resonance to cartesian transformation $\Re_c$ and its inverse $\Re_c^{-1}$:

$$\begin{aligned} \Re_c q_k &= \frac{q_k + p_k}{2} \\ \Re_c p_k &= \frac{q_k - p_k}{2(i\epsilon_k + \bar{\epsilon}_k)} \end{aligned} \tag{4.9a}$$

$$\begin{aligned} \Re_c^{-1} q_k &= q_k + (i\epsilon_k + \bar{\epsilon}_k) p_k \\ \Re_c^{-1} p_k &= q_k - (i\epsilon_k + \bar{\epsilon}_k) p_k \end{aligned} \tag{4.9b}$$

In terms of the transformation $\Re_c$, $\mathcal{T}^{-1} f_r$ is given by

$$\mathcal{T}^{-1} f_r = \phi_{a+ib}(f_r \circ \Re_c) \circ \Re_c^{-1} . \tag{4.10}$$

Finally, in terms of the real transformation directly available in the DA-software, $\mathcal{T}^{-1} f_r$ can be written as

$$\mathcal{T}^{-1} f_r = \phi_\pi \left\{ \phi_a \left[ (\phi_\pi f_r) \circ \Re_r \right] \circ \Re_r^{-1} \; + \; \phi_b \left[ (\phi_\pi f_r) \circ \Re_r \circ \Im \right] \circ \Re_r^{-1} \right\} . \tag{4.11}$$

The addition in (4.11) is defined in the usual manner:

$$(f+g)(\mathbf{x}) \; = \; f(\mathbf{x}) + g(\mathbf{x}) . \tag{4.12}$$

In the next section, we examine how one produces a first order factorized representation of the type (2.1) from a power series representation of the ray. This will be an essential element of the recursive loop used in the analysis of the one period map $\mathcal{M}$.

## 5. The Extraction of a First-Order Lie Representation

Our ultimate goal is to perform a perturbative computation of the power series representation of the map to arbitrary order. To achieve this we must be able to extract a Lie generator to first order in some smallness parameter $\alpha$.

Consider the map $\mathcal{M}$ of Eq. (2.1), let us assume that we have a formal power series representation of $\mathcal{R}^{-1}\mathcal{M}$(*i.e.* $\mathcal{R}^{-1}\mathcal{M}\mathbf{x} = \; (\mathcal{M}\mathbf{x}) \circ (\mathcal{R}^{-1}\mathbf{x})$):

$$\mathcal{R}^{-1}\mathcal{M}\,\mathbf{x} \; = \; \mathbf{x} + \alpha \sum_{k=2}^{N_o-1} \; \mathbf{G}_k(\mathbf{x};\alpha) = \mathbf{x} + \alpha \mathbf{G}(\mathbf{x};\alpha) . \tag{5.1}$$

In Eq. (5.1), the functions $\mathbf{G}_k(\mathbf{x})$ are homogenous polynomials in $\mathbf{x}$ and $N_o - 1$ is the highest power in $\mathbf{x}$. Assuming a Lie representation for $\mathcal{R}^{-1}\mathcal{M}$ as done in (2.1),

we can evaluate its effect on $\mathbf{x}$:

$$\mathcal{R}^{-1}\mathcal{M}\mathbf{x} = \exp\Big(:\alpha f(\mathbf{x};\alpha):\Big)\mathbf{x} = \mathbf{x} + \alpha\,[f(\mathbf{x};0),\mathbf{x}] + O(\alpha^2)\ldots \tag{5.2}$$

Comparing Eqs. (5.1) and (5.2), we get equation for $f(\mathbf{x};0)$ [7, 17]:

$$\Big[f(\mathbf{x};0),\mathbf{x}\Big] = \mathbf{G}(\mathbf{x};0)$$

$$-J\nabla f(\mathbf{x};0) = \mathbf{G}(\mathbf{x};0)$$

$$\Longrightarrow \quad f \;=\; \int_0^{\mathbf{x}} J\mathbf{G}(\mathbf{x}';0)d\mathbf{x}'\,. \tag{5.3}$$

over an arbitrary path

In the formula for $f$, the matrix $J$ is the symplectic form used to define the Poisson bracket. The independence of $f$ from the integration path is a consequence of the symplectic nature of $\mathcal{R}^{-1}\mathcal{M}$ [17]. To perform this integration, we choose a path along the "diagonal":

$$\mathbf{x}' = \eta\,\mathbf{x}\,,$$

$$d\mathbf{x}' = d\eta\,\mathbf{x}\,. \tag{5.4}$$

We can write the resulting $f$ with the help of the special exponent dependent transformation $\phi$. We first define a function $\sigma(\mathbf{j})$ which simply counts the powers of $\mathbf{x}$ in each monomial of $\mathbf{G}$:

$$\sigma(\mathbf{j}) = \frac{1}{\left(\sum\limits_{k=1}^{2N} j_k\right)+1}\,. \tag{5.5}$$

The integral for $f$ is then given by the DA-software operation:

$$f(\mathbf{x};0) = \mathbf{x}^\dagger J\phi_\sigma\mathbf{G}(\mathbf{x};0) = \sum_{k=1}^{N}\Big\{x_{2k}(\phi_\sigma\mathbf{G})_{2k-1} - x_{2k-1}(\phi_\sigma\mathbf{G})_{2k}\Big\} \tag{5.6}$$

Finally, we must point out that it is possible to compute the DA-function asso-

ciated with this first order approximation of the map $\mathcal{R}^{-1}\mathcal{M}$ because of our ability to perform Poisson bracket on DA-functions:

$$\exp\Big(:\alpha f(\mathbf{x};0):\Big)\mathbf{x} = \mathbf{x} + \sum_{k=2}^{N_o-1} \frac{\alpha^{(k-1)}}{(k-1)!} \; : f(\mathbf{x},0) :^{k-1} \mathbf{x}\,. \tag{5.7}$$

Needless to say that this function or its inverse ($\alpha \to -\alpha$) can be composed with another DA-function exactly in a formal power series sense thanks to the DA-software. This is how we can bypass the Campbell-Baker-Hausdorff formula.

In section 6, we will reach our final goal, that is to write a recursive procedure brings a map into a normal form.

## 6. Recursive Loop for the Order by Order Normalization of the Map $\mathcal{M}$

Let us assume that we have a map $\mathcal{M}$ factored into the linear part $\mathcal{R}$ of Eq. (3.1) and a non-linear map $\mathcal{N}_0$:

$$\begin{aligned} \mathcal{M} &= \mathcal{R}\mathcal{N}_0\,, \\ \mathcal{N}_0\mathbf{x} &= \mathbf{x} + \sum_{k=2}^{N_o-1} \mathbf{G}_k^o\,. \end{aligned} \tag{6.1}$$

Our final goal is to express $\mathcal{M}$ as follows

$$\mathcal{M} = \mathcal{A}^{-1}\mathcal{R}\,\mathcal{N}_\Omega\,\mathcal{A}\,, \tag{6.2a}$$

$$\mathcal{A} = \exp(: F_\Omega :)\ldots\exp(: F_\omega)\ldots\exp(: F_1 :)\,, \tag{6.2b}$$

$$\mathcal{N}_\Omega = \exp(: T_1 :)\ldots\exp(: T_\Omega :)\,. \tag{6.2c}$$

In general, as we will see, the transformation $\phi_a$ and $\phi_b$ introduced in Eqs. (4.4) and (4.5) will determine the target map $\mathcal{N}_\Omega$. For example, if all the $T_\omega$'s belong to

$Ker\ \mathcal{T}$, we simply define $a$ and $b$ as follows:

$$\text{if} \qquad \sum_{k=1}^{N} |j_{2k} - j_{2k-1}| = 0 \quad \text{then } a = b = 0 \ , \tag{6.3}$$

Otherwise, $a(\mathbf{j})$ and $b(\mathbf{j})$ are given by (4.5).

Sometimes, one would like to leave one or many resonances not belonging to $Ker\,\mathcal{T}$ in the $T_\omega$s. This is achieved by letting $a$ and $b$ be zero for the particular $\mathbf{j}$ vectors belonging to these resonances. Finally, in the case of total removal of the resonances (Eq. (6.3)), the map $\mathcal{RN}_\Omega$ can be written as a single exponent:

$$\mathcal{RN}_\Omega = \exp\left( : f_2 + \sum_{\omega=1}^{\Omega} T_\omega : \right) . \tag{6.4}$$

In this particular case, we can even compute the formal pseudo-Hamiltonian at that particular surface of section:

$$\begin{aligned} \mathcal{M} &= \exp(: \widetilde{H} :) \\ \widetilde{H} &= \mathcal{A}^{-1}\left( f_2 + \sum_{\omega=1}^{\Omega} T_\omega \right) . \end{aligned} \tag{6.5}$$

Again, we emphasize that all the operations given here are within the power of the DA-theory and *software*.

We now describe the recursive process leading from the factored representation of (6.1) to the final result of Eq. (6.2). This will be achieved in $\Omega$ steps where $\Omega$ is just $N_o - 2$ for an order by order perturbative calculation. For the superconvergent process of appendix C, $\Omega$ is just the integer part of $1 + \log(N_o - 2)/\log 2$. We will see that the nature of the process is partly determined by the function $\sigma(\mathbf{j})$ introduced in Eq. (5.5).

Let us look at the DA-operations leading from the $\omega^{th}$ step to the $(\omega+1)^{th}$ step. To start we assume that the following maps and DA-functions have been found:

$$\mathcal{A}_\omega \mathbf{x} = \exp(: F_\omega :)\mathbf{x} = \mathbf{x} + \mathbf{F}^\omega \tag{6.6a}$$

$$\mathcal{N}_\omega \mathbf{x} = \mathbf{x} + \sum_{k=2}^{N_o-1} \mathbf{G}_k^\omega = \mathbf{x} + \mathbf{G}^\omega \tag{6.6b}$$

$$\begin{aligned} \mathcal{N}_\omega^t \mathbf{x} &= \exp(: -T_\omega :) \ldots \exp(: -T_1 :)\mathbf{x} \\ &= \mathbf{x} + \mathbf{T}^\omega \end{aligned} \tag{6.6c}$$

We first compute $T_{\omega+1}$ and $F_{\omega+1}$:

$$F_{\omega+1} = \mathcal{T}^{-1}\mathbf{x}^\dagger J \phi_\sigma \Big( (\mathcal{N}_\omega \mathbf{x}) \circ (\mathcal{N}_\omega^t \mathbf{x}) - \mathbf{x} \Big) . \tag{6.7}$$

For a superconvergent process $\sigma(\mathbf{j})$ is exactly given by Eq. (5.5). However for the normal order by order process described in this section, $\sigma(\mathbf{j};\omega)$ is $\omega$-dependent:

$$\begin{cases} \sigma(\mathbf{j};\omega) = \frac{1}{\omega+3} & \text{if } \sum_{k=1}^{2N} j_k = \omega + 2 , \\ \sigma(\mathbf{j};\omega) = 0 & \text{otherwise} \end{cases} \tag{6.8}$$

Of course, $T_{\omega+1}$ is just the left over part which is saved along during the evaluation of $\mathcal{T}^{-1}$. More precisely, the terms which are zeroed by $\phi_a$ and $\phi_b$ are kept and transformed along with those leading to $F_{\omega+1}$.

To close the recursive loop, we must compute $\mathcal{A}_{\omega+1}\mathbf{x}$. This is of course a DA-operation since it involves a finite number of Poisson brackets, namely at most $(N_o - 2)/(\omega + 1)$ terms of the Lie exponential map. The same can be said about

$\mathcal{A}_{\omega+1}^{-1}\mathbf{x}$. Hence we obtain the DA-map $\mathcal{N}_{\omega+1}$:

$$\mathcal{N}_{\omega+1}\mathbf{x} = (\mathcal{A}_{\omega+1}\mathbf{x}) \circ (\mathcal{N}_{\omega}\mathbf{x}) \circ (\mathcal{A}_{\omega+1}^{-1}\mathbf{x}) . \tag{6.9}$$

By exponentiation of the Poisson bracket or composition, the map $\mathcal{N}_{\omega+1}^{t}$ is also computed:

$$\mathcal{N}_{\omega+1}^{t}\mathbf{x} = (\mathcal{N}_{\omega}^{t}\mathbf{x}) \circ \Big(\exp(: -T_{\omega+1} :)\mathbf{x}\Big) \tag{6.10}$$

This last operation brings us back to the initial point of Eq.(6.6) with $\omega$ raised to $\omega + 1$. The similar process of normalization can be now applied on the pseudo-Hamiltonian $\widetilde{H}$. This will involve a few changes in the recursive process and a redefinition of the function $a$ and $b$. We describe this in Appendix B.

## Conclusion

We have described in this paper the algorithm necessary for "normalizing" a map using its DA-representation. This algorithm has been written and tested against the less powerful tools installed in MARYLIE 5.0 by Neri and Dragt [3]. The algorithm itself depends entirely on the ability to partially factor a map in terms of Lie operators. Its implementation to arbitrary order, arbitrary number of parameters and arbitrary signature of the quadratic invariants is made possible only by the awesome power of the differential algebra package of Berz. For example, at this moment, we estimate that it is feasible to produce a ninth order map (with five variables) for the realistic lattice of the SSC. For smaller machines, we can produce maps of even higher order. This will be of great use in the investigation of undulators and their effects on circular light source devices.

The maps we produce can be exactly analyzed with our algorithm and daredevils can use them for tracking. Incidentally, the partial map inverter of the DA-package, in addition of computing the fixed point map $\mathcal{F}\mathbf{z}$ mentioned in the introduction, permits *several* "symplectification" schemes of any DA-function used in a tracking simulation.

Along the lines of perturbation theory, it is clear that the DA-software can be useful in manners not described or even foreseen by the authors. For example, the symplectic-SU(2) maps needed in Yokoya's algorithm [15] as well as the algorithm itself can be written easily using the DA-software. This is especially true since the coupling from the orbital on the spin motion requires a high order expansion in the spin directions.

We hope that the reader will derive two conclusions from this paper: firstly, that formal perturbation theory on the map and a Hamiltonian-free [2]approach to single particle dynamics in complicated periodic systems is much more powerful than a straightforward application of the Birkhoff or Deprit normal form algorithm directly on the messy Hamiltonian [13], and secondly that the differential algebra method supersedes all numerical differentiation methods and map manipulators by its generality and precision. For this reason, we restrained ourselves from giving simple examples, for it would give a false impression to the reader. (See Reference [14], Appendix B for an analysis of a trivial linear map.)

## Acknowledgments

We would like to thank Dr. Alexander Chao of the SSC Central Design Group for financially permitting Martin Berz and the two other authors to work together on the normal form problem. In addition, we are grateful to Dr. Filippo Neri and

Prof. Alex Dragt of the University of Maryland for providing us with a version of MARYLIE which can analyze stable fifth order maps. It was used to benchmark our calculations and test our algorithms. Finally, we would like to express our gratitude to Miss Valerie Kelly for her prompt and skillful scientific typing of the manuscript.

## Appendix A. The Coasting Plane Normalization Procedure

Let us assume that some part of the $f_2$ are "drift" like in nature. Then $f_2$ can be reduced and broken into two parts:

$$f_2 = f_{2H} + f_{2D}\,, \tag{A.1a}$$

$$= \sum_{k=1}^{N_H} -\frac{\mu_k}{2}\left(q_k^2 + (\epsilon_k - \bar{\epsilon}_k)p_k^2\right) + \sum_{k=N_H+1}^{N_H+N_D} -\frac{p_k^2}{2}\,. \tag{A.1b}$$

If we further assume that all the $p_k$'s ($k > N_H$) are constants of the motion of the full non-linear map, the procedures outlines in the body of this paper follow immediately by simply considering the $p_k(k > N_H)$ as parameters instead of canonical variables. The basis vectors are just obtained from $|\mathbf{m},\mathbf{n}\rangle$ of Eq. (3.3):

$$|\mathbf{m},\mathbf{n};\boldsymbol{\nu}\rangle = |\mathbf{m},\mathbf{n}\rangle\, \delta_1^{\nu_1} \ldots \delta_{N_P+N_H}^{\nu_{N_P+N_H}}\,, \tag{A.2a}$$

$$\delta = \left(\delta_1, \ldots, \delta_{N_P}, p_{N_{H+1}}, \ldots, p_{N_H+N_D}\right)\,. \tag{A.2b}$$

Here $\delta$ contains the non-canonical parameters as well as the drift-like momenta. Essentially, the absence of the $q_k(k > N_H)$ from the Lie generators of the map, allows for a full diagonalization of $\mathcal{T}$ in terms of the $|\mathbf{m},\mathbf{n};\boldsymbol{\nu}\rangle$ eigenfunctions.

Physically, the longitudinal phase space is an example of a "coasting" plane whenever RF-cavities are turned off in a circular ring.

When the $p_k$'s ($k > N_H$) are not constant, the transformation $\mathcal{T}$ cannot be diagonalized. At best $\mathcal{T}$ can be put in Jordan normal form. For our resonance basis, we will again factor the harmonic part from the "drifting" planes:

$$|\mathbf{m},\mathbf{n},\mathbf{k},\mathbf{l},\boldsymbol{\nu}\rangle = |\mathbf{m},\mathbf{n}\rangle\, |\mathbf{k},\mathbf{l}\rangle_D\, |\boldsymbol{\nu}\rangle_P \tag{A.3a}$$

$$|\mathbf{k},\mathbf{l}\rangle = q_{N_{H+1}}^{k_1} \cdots q_{N_H+N_D}^{k_{N_D}}\, p_{N_{H+1}}^{k_1} \cdots p_{N_H+N_D}^{k_{N_D}} \tag{A.3b}$$

$$|\nu\rangle_p = \delta_1^{\nu_1} \dots \delta_{N_P}^{\nu_{N_P}} . \tag{A.3c}$$

We also factor the map $\mathcal{R}$ into the harmonic and the drifting part:

$$\mathcal{R} = \mathcal{R}_H \mathcal{R}_D = \mathcal{R}_D \mathcal{R}_H = \exp(: f_{2H} :) \exp(: f_{2D} :) \tag{A.4}$$

Finally, we rewrite the operator $\mathcal{T}$ as a product:

$$\mathcal{T} = \left( \mathcal{E} - \mathcal{R}_H^{-1} \mathcal{R}_D^{-1} \right) = \mathcal{T}_H \left( \mathcal{E} - \mathcal{R}_H^{-1} \mathcal{T}_H^{-1} \mathcal{D} \right) , \tag{A.5a}$$

$$\mathcal{D} = \mathcal{R}_D^{-1} - \mathcal{E} = \sum_{k=1}^{\infty} \frac{: -f_{2D} :^k}{k!} \tag{A.5b}$$

$$\mathcal{T}_H = \mathcal{E} - \mathcal{R}_H \tag{A.5c}$$

Due to the special nature of $f_{2D}$, the operator $: f_{2D} :$ is nilpotent in a formal power series sense. In fact, if our maximum order is $N_o$, we must have:

$$\mathcal{T}^{-1} = \mathcal{T}_H^{-1} \sum_{k=0}^{N_o} (\mathcal{R}_H^{-1} \mathcal{T}_H^{-1})^k \, \mathcal{D}^k . \tag{A.6}$$

The form of $\mathcal{T}^{-1}$ will complicate the process outlined in Section 4. Physically it corresponds to a particle undergoing oscillations in an RF-bucket with no linear part. Because of the rarity of this problem, we have not yet implemented the map $\mathcal{T}$ of (A.6) in our normal form software. It should not pose any problem to the interested user.

Mathematically, the harmonic part $\mathcal{R}_H$ has a semisimple Lie generator and, as a result, the space of polynomial functions is a direct sum of the range and the kernel of $\mathcal{R}_H$. In the coasting case, $\mathcal{R}$ can be factored into a semisimple part $\mathcal{R}_H$ and a nilpotent part $\mathcal{R}_D$ substantially complicating the process [16].

## Appendix B: Order by Order Normalization of the Pseudo-Hamiltonian $\widetilde{H}$

The pseudo-Hamiltonian $\widetilde{H}$ can be written as a sum of a quadratic part $f_2$ given by (3.1b) and a non-linear part $\alpha V$:

$$\widetilde{H} = f_2 + \alpha V \,. \tag{B.1}$$

Following Dragt and Finn[11], we transform the map $\exp(: \widetilde{H} :)$ by a similarity transformation $\mathcal{A}$ as done in Section 2. One gets the result:

$$\begin{aligned} \mathcal{N} &= \mathcal{A} \exp(: \widetilde{H} :) \mathcal{A}^{-1} \\ &= \exp(: \mathcal{A}\widetilde{H} :) \\ &= \exp\left(: f_2 : + : -f_2 : F + \alpha V + O(\alpha^2) \ldots :\right) \end{aligned} \tag{B.2}$$

From (B1), we define again a map $\mathcal{T}$:

$$\mathcal{T} =: f_2 : \,. \tag{B.3}$$

Letting $: f_2 :^{-1}$ act on $|\mathbf{m}, \mathbf{n}\rangle$ provides the new functions $a(\mathbf{j})$ and $b(\mathbf{j})$ which are to be used to produce the transformations $\phi_a$ and $\phi_b$. This was already provided by Eq. (3.4):

$$\begin{aligned} : f_2 :^{-1} \; |\mathbf{n} - \mathbf{m}\rangle &= \Big[(\mathbf{n} - \mathbf{m}) \cdot \boldsymbol{\lambda}\Big]^{-1} |\mathbf{m}, \mathbf{n}\rangle \\ &= \frac{1}{(i\Delta + \overline{\Delta})} \; |\mathbf{m}, \mathbf{n}\rangle \\ &= \frac{\overline{\Delta} - i\Delta}{\Delta^2 + \overline{\Delta}^2} \; |\mathbf{m}, \mathbf{n}\rangle \,. \end{aligned} \tag{B.4}$$

Consequently, we get for $a$ and $b$

$$a(\mathbf{j}) = \frac{\overline{\Delta}}{\Delta^2 + \overline{\Delta}^2}\,,$$
$$b(\mathbf{j}) = \frac{-\Delta}{\Delta^2 + \overline{\Delta}^2}\,. \qquad (B.5)$$

Let us start again the recursive process as was done in Section 6, by postulating the existence of a few maps:

$$\widetilde{H}_\omega = f_2 + V_\omega$$

$$\mathcal{A}_\omega \mathbf{x} = \exp(: F_\omega :)\mathbf{x} = \mathbf{x} + \mathbf{F}^\omega \qquad (B.6)$$

We first compute $F_{\omega+1}$ again using $\mathcal{T}^{-1}$:

$$F_{\omega+1} = \mathcal{T}^{-1}\mathcal{P}_{\omega+2}V_\omega \qquad (B.7)$$

Here $\mathcal{P}_{\omega+2}$ projects the part of $V_\omega$ which is of degree $\omega + 2$.

Then the DA-function $\mathcal{A}_{\omega+1}\mathbf{x}$ is computed with a finite expansion of the Lie exponentiation as we did in Section 6. Finally, $\widetilde{H}_{\omega+1}$ is simply obtained by composition:

$$\widetilde{H}_{\omega+1} = \widetilde{H}_\omega \circ (\mathcal{A}_{\omega+1}\mathbf{x}) = f_2 + V_{\omega+1}\,. \qquad (B.8)$$

This final step closes the recursive loop.

Again any resonance can be left in by a suitable modification of $a(\mathbf{j})$ and $b(\mathbf{j})$. One bizarre application of this method in connection with the full DA-software package, is the ability to compute the full $(N_o - 1)^{th}$ order matrix of any ideal electro-magnetic element exactly. This is true because the formal solution of an

ideal element can be written as:

$$\mathcal{M} = \exp(: f_2 + V :) \tag{B.9}$$

Then one can follow our procedure and get for $\mathcal{M}$:

$$\mathcal{M} = \mathcal{A}^{-1} \exp(: f_2 + V_\Omega :)\mathcal{A}\,. \tag{B.10}$$

However, since $f_2 + V_\Omega$ is a function of the $f_2^k$'s only (see Eq. (3.1b)), its motion can be computed exactly in terms of the sine, the cosine and their hyperbolic counterparts.

Having the exact solution in terms of elementary functions allows us to compute the DA-representation of the map (see preceding paper of this issue). Therefore, we can get $\mathcal{M}\mathbf{x}$:

$$\mathcal{M}\mathbf{x} = (\mathcal{A}\mathbf{x}) \circ (\exp(: f_2 + V_\Omega :)\mathbf{x}) \circ (\mathcal{A}^{-1}\mathbf{x})\,. \tag{B.11}$$

It is not clear whether this surprising result has a real practical application in the field of power series codes. (See again the preceeding paper)

## Appendix C: The Superconvergent Algorithm on the Map

A superconvergent algorithm can be constructed by restoring the function $\sigma$ used in the integration to its general form given by Eq. (5.5) and defining an $\omega$ dependent $\mathcal{T}$ map. To achieve superconvergence in the algorithm of section 6, we define $\mathcal{T}$ as follows:

$$\mathcal{T}(\omega) = \mathcal{E} - \mathcal{R}^{-1}\mathcal{N}_\omega^t \tag{C.1a}$$

$$\mathcal{N}_\omega^t = \text{tune shift map defined by Eq. (6.6c)} \tag{C.1b}$$

Because $\mathcal{N}_\omega^t - \mathcal{E}$ is nilpotent over the space of truncated polynomials , we can use the trick of appendix A to invert the map $\mathcal{T}(\omega)$:

$$\mathcal{T}^{-1}(\omega) = \mathcal{T}_0^{-1} \sum_{k=0}^{N_o-3} (\mathcal{R}^{-1}\mathcal{T}_0^{-1})^k \, \mathcal{D}_\omega^k \,, \tag{C.2a}$$

$$\mathcal{T}_0 = \mathcal{E} - \mathcal{R}^{-1} \,, \tag{C.2b}$$

$$\mathcal{D}_\omega = \mathcal{N}_\omega^t - \mathcal{E} \,, \tag{C.2c}$$

The superconvergent algorithm is faster than the order by order procedure for very high order calculations. On a simple example, it was slower by a factor of two at the sixth order but about 80 percent faster at the sixteenth order despite no special attempts to optimize the superconvergent algorithm. We leave it to the reader to construct a superconvergent process reproducing the results of appendix B.

## References


1. E.D. Courant and H.S. Snyder, Ann. Phys. **3** (1958) 1.
2. E. Forest, SSC Report 111 (1987).
3. A.J. Dragt et al., IEEE Trans. Nucl. Sci., NS-32 (1985) 2311.
4. Done to $3^{rd}$ order in TEAPOT and $5^{th}$ order in THINTRAC. Later done to $5^{th}$ order in TEAPOT with the DA-software.
5. A. Robinson, Proc. Royal Acad. Amsterdam Ser. A, **64**, 432.
6. D. Laugwitz, J. Ber. Deutsch. Math. Vereinigung **75** (1973) 66.
7. A.J. Dragt and J.M. Finn, J. Math. Phys. **20** (1979) 2649.
8. E. Forest, SSC Report 29 (1985).
9. A. Deprit, Cel. Mech. **1** (1969) 12.
10. J.R. Cary, Phys. Rep. **79** (1981) 129–159.
11. A.J. Dragt and J.M. Finn, J. Math. Phys. **17** (1976) 2215.
12. G. Hori, Pub. Astron. Soc. Jap. **18** (1966) 287.
13. L. Michelotti, Part. Acc. **16** (1985) 233.
14. E. Forest, Part. Acc. **22** (1987) 15.
15. K. Yokoya, Nucl. Instr. and Meth. A258 (1987) 149.
16. R. Cushman, A. Deprit, and R. Mosak, J. Math. Phys. **24** (1983) 2102.
17. E. Forest, SSC Report 138 (1987).