DIFFERENTIAL ALGEBRA - A NEW TOOL*

M. Berz
Lawrence Berkeley Laboratory, Berkeley, CA 94720, USA and University of Giessen, West Germany

Abstract

An overview of the theory and implementation of the differential algebraic method is presented. The method allows a very straightforward computation of Taylor maps of beamlines. Contrary to older techniques, the method is not restricted to low order and allows the study or arbitrarily many variables, including system parameters.

The resulting Taylor maps offer a variety of useful applications, including fast short term tracking, the computation of generating functions which can be used for symplectic tracking, as well as the direct computation of tuneshifts, smear, and approximate invariants. References to more detailed literature are given.

Description of Particle Accelerators by Mappings

The motion through a beam line or particle accelerator can be described by a map relating final phase space variables $\vec{z}_f$ to initial phase space variables $\vec{z}_i$ and some machine parameters of interest $\vec{\delta}$:

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

The most direct way to study the map and its behaviour under repetition is its evaluation on specific particle coordinates. This is the well-known tracking method. This method allows a qualitative discussion of stability and gives an estimate of the topology of the occupied phase space area. Its disadvantages are the computational expense and the difficulties in interpreting the results. In particular, many quantities of interest like tune shifts, chromaticities, smear and invariants cannot be extracted directly but only in a time intensive way requiring the tracking of particles for many revolutions.

Because of the computational expense of evaluating the transfer map, it is advantageous to approximate it by a function that is easier to evaluate. In addition, it is advantageous if the approximated function allows some analytical manipulation, for example a direct computation of relevant quantities, search for invariants etc.

A very natural technique to approximate functions that are very costly to evaluate, in particular in the case of functions which are well behaved and smooth, is interpolation; one first evaluates the function on sufficiently many suitably chosen points, and then fits some other function which is easier to evaluate through these points.

To be more specific, one could evaluate the function on a regular grid in phase (and parameter) space and use one of the many existing interpolation schemes. Alternatively, one could expand the action-angle representation of the function in a Fourier series for the angle part and a polynomial for the action part, as it is done in the method by Warnock et al. [1]. Interpolation being one of the best understood and studied subjects of numerical analysis, good results can be achieved by carefully using appropriate techniques.

A significant drawback of this approach becomes noticeable in high dimensions: The required number of interpolation points increases drastically, and so does the amount of data required to describe the map. In the simplest case, the interpolation on a multi-dimensional grid, a total of

$$N = n^v \tag{2}$$

points are required to describe the map, where n is the number of cells per dimension, and v is the dimensionality of the map. After interpolation, the number of required parameters for the description of the map is also in the order of N. For example, in the case of a modest $n = 10$ points per phase space direction, one obtains $N = 10000$ in the case of four dimensions, and $N = 1000000$ in the case of six dimensions, without treating any dependencies on system parameters.

In practical situations, numbers in this magnitude are actually encountered: In the case of the Warnock et al. interpolation method, a total of about $N = 11000$ points are required for an only four dimensional simulation of the SLC North Damping Ring.

Using appropriate techniques designed for speed, for example a well programmed arbitrary dimension local spline approximation, an increase in speed as compared to the direct tracking is very likely to become apparent, in particular for large lattices. Nevertheless, because of the complexity of the interpolation function, not very much more insight has been obtained, and the direct extraction of most quantities of interest is still rather cumbersome.

The major difficulty of the direct interpolation, the very large amount of data required to describe a map, in particular for many variables, can be overcome by using a Taylor series expansion of the map. In the case of our study of accelerator phase space maps, this also seems a quite natural approach because the map is "almost" linear and we are interested in the study of the effects of the "small" nonlinear perturbations. In the case of Taylor maps, use is made of the fact that the operation of taking partial derivatives with respect to different variables commute, i.e. for example

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x} \tag{3}$$

This can be understood as a symmetry of suitable smooth maps and decreases the amount of data required to describe a map significantly. It can be shown [2] that the total number of derivatives required to describe a function in v variables by its Taylor series to order n requires a total of

$$N = \frac{(n+v)!}{n!v!} \tag{4}$$

For example, if we want to describe a four variable function to tenth order, this yields $N = 1001$, and for a six variable function to the same order we obtain $N = 8008$.

* This work was supported by the Director, Office of Energy Research, Office of Basic Energy Sciences, Materials Sciences Division, Department of Energy under Contract No. DE-AC03-76SF00098.

If we wanted to interpolate a six dimensional function on a regular grid with a similar amount of data, we could only choose around four or five interpolation points per dimension.

The major characteristic of a high order Taylor approximation is that its accuracy is extremely high close to the expansion point, but then also decreases noticeably when going further and further away. So the accuracy of the approximation is highly weighted, favoring points close to the center, whereas in the case of interpolation the accuracy is typically uniform.

In very unfavorable cases, even for functions whose Taylor series converges to the function (which is the case for our maps), it may be quite cumbersome to increase the order sufficiently to achieve the desired accuracy for very nonlinear maps. In this case, it may be advisable to treat points very far away from the expansion point separately, for example by using additional lower order Taylor series with suitably chosen expansion points.

Taylor series codes have a long history [3,4,5,6,7] and originated in the optical disciplines and the related study of guidance systems and spectrometers. In these cases, a much appreciated feature of the Taylor series method is the ability to relate certain Taylor coefficients to certain classes of beam line elements, which is of utmost importance for the design and correction of these devices.

The major limitation of these codes has been that the orders that could be treated are very low; only recently has it been possible to generate a fifth order code [8,7,9].

In the next section we will introduce a different technique for the computation of Taylor series from maps that allows a rather straightforward computation to arbitrary order.

Differential Algebras

In the traditional Taylor series codes, the total Taylor series of the system is computed in the following way. The codes contain a library of formulas that describe the Taylor series coefficients for each possible element of the beamline, such as drifts and multipoles, as a function of the parameters of the element, such as length and strength. Then, the total Taylor series is computed by combining the individual Taylor maps into a total Taylor map by using the chain rule.

The limitation of this procedure is that the complexity of the Taylor series for the individual elements increases drastically with the order. This suggests to brake down the computation of individual Taylor series into smaller pieces. In the method outlined her, we even go to the most extreme case and represent each individual addition and multiplication separately. For details we refer to [10,11,2,12,13]

Consider the vector space R^2 of ordered pairs (q_0, q_1), $q_0, q_1 \in R$ in which an addition and a scalar multiplication are defined in the usual way:

$$(q_0, q_1) + (r_0, r_1) = (q_0 + r_0, q_1 + r_1) \tag{5}$$

$$t \cdot (q_0, q_1) = (t \cdot q_0, t \cdot q_1) \tag{6}$$

for $q_0, q_1, r_0, r_1, t \in R$. Besides the above addition and scalar multiplication a multiplication between vectors is introduced:

$$(q_0, q_1) \cdot (r_0, r_1) = (q_0 \cdot r_0, q_0 \cdot r_1 + q_1 \cdot r_0) \tag{7}$$

for $q_0, q_1, r_0, r_1 \in R$. With this definition of a vector multiplication, the set of ordered pairs becomes an algebra.

Looking at numbers $(a, 0)$, we see that for addition, multiplication and ordering they behave like real numbers. So we can embed the real numbers into the structure in the same way they could be embedded into the complex numbers.

It is easy to verify that $(1,0)$ is a neutral element of multiplication, because according to equation (7)

$$(1,0) \cdot (q_0, q_1) = (q_0, q_1) \cdot (1,0) = (q_0, q_1) \tag{8}$$

It turns out that (q_0, q_1) has a multiplicative inverse if and only if q_0 is nonzero; so the structure is not a field. In case $q_0 \neq 0$, the inverse is

$$(q_0, q_1)^{-1} = (\frac{1}{q_0}, -\frac{q_1}{q_0^2}) \tag{9}$$

Using Equation (4), it is easy to check that in fact $(q_0, q_1)^{-1} \cdot (q_0, q_1) = (1,0)$.

This structure is very helpful for the computation of derivatives, as shall be illustrated now. Let us consider the following example function:

$$f(x) = x^2 + \frac{1}{x} \tag{10}$$

Differentiating the function yields:

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

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

$$f(2) = \frac{9}{2},\ f'(2) = \frac{15}{4} \tag{12}$$

Now take the definition of the function f in Equation (10), replace all operations occurring in it by the corresponding ones in our algebra, and evaluate it at $(2,1)$. We obtain:

$$\begin{aligned} f[(2,1)] &= (2,1)^2 + (2,1)^{-1} \\ &= (4,4) + (\frac{1}{2}, -\frac{1}{4}) \\ &= (\frac{9}{2}, \frac{15}{4}) \end{aligned} \tag{13}$$

As we can see, after the evaluation of the function the first component of the result is just the value of the function at $x = 2$, whereas the second component is the derivative of the function at $x = 2$.

There are at least two ways to view and understand this phenomenon. We begin with the most down-to-earth and elementary view. 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 (5) 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 (7) 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 (9) one has derived a rule to differentiate the function $f(x) = 1/x$ without explicitly using calculus rules.

It is quite obvious how this method can be generalized to higher derivatives and several variables: one just collects all the derivatives into one vector, introduces the usual componentwise addition and models a vector multiplication using the product rule. Following the same reasoning as above, we can verify that we can compute accurate derivatives.

In this general case of higher derivatives and several variables, it is useful to introduce another operation besides addition, scalar multiplication and vector multiplication. Remembering that all the vectors can be viewed as containing a collection of derivatives of a function f, we can find the vector of derivatives of the derivative function f' by just moving components appropriately. This operation we denote with ∂. It is easy to verify that we then have

$$\partial(\vec{a} \cdot \vec{b}) = (\partial \vec{a}) \cdot \vec{b} + \vec{a} \cdot (\partial \vec{b}) \tag{14}$$

Whenever there is such an operation besides addition and multiplication, the resulting structure is called a differential algebra [14]. We note that if there is more than one such operator (as in the case of partial derivatives of several variables), the operators can be used to introduce a Lie algebraic structure using the usual Poisson bracket.

We note that virtually all standard functions like trigonometric functions, exponentials, roots etc. can be introduced in the new structure. For details we refer to [2].

Besides the hands-on view of differential algebra just presented, there is another more mathematical view of the subject that ultimately leads to very fascinating results. The arithmetic introduced in the beginning of the chapter has been known for a very long time in mathematics, it can be traced back for at least 100 years [15,16]. What captured the interest of the mathematicians in these days was not so much the connection to sum and productrule of calculus, but rather that the structure has some remarkable ordering properties.

Indeed, the set can be ordered in the following way. Let (a, b) and (c, d) be given; we say

$$\begin{aligned} (a,b) &< (c,d) \text{ if } a < c, \text{ or } (a = c \text{ and } b < d) \\ (a,b) &> (c,d) \text{ if } a > c, \text{ or } (a = c \text{ and } b > d) \\ (a,b) &= (c,d) \text{ if } a = c \text{ and } b = d \end{aligned} \tag{15}$$

So in order to determine ordering, we look first only at the first components. If they already differ, they alone determine which of the numbers is larger. Only if they are the same do we compare the second components.

From this definition it is clear that for any (a, b) and (c, d), exactly one of the three properties always holds: $(a, b) = (c, d)$ or $(a, b) < (c, d)$ or $(a, b) > (c, d)$. Furthermore, it follows that if $(a, b) < (c, d)$, then for arbitrary (e, f) we have $(a, b) + (e, f) < (c, d) + (e, f)$, and for $(e, f) > 0$ we have $(a, b) \cdot (e, f) < (c, d) \cdot (e, f)$. Altogether, the algebra assumes the rather special structure of a arithmetically ordered set.

Where in the complex numbers, $(0, 1)$ was a root of -1, here it has another interesting property. Looking at the ordering relations, we infer that for a real number $r > 0$,

$$(0,0) < (0,1) < (r,0) \tag{16}$$

Hence $(0, 1)$ lies "in between" 0 and every real number, i.e. $(0, 1)$ is infinitely small.

Because of this we call $d = (0, 1)$ the differential unit. The first component of the pair (q_0, q_1) is called the real part, and the second component is called the differential part.

Now consider the relationship

$$f(x + \Delta x) = f(x) + \Delta x \cdot f'(x) + \Delta x^2 \cdot r(x) \tag{17}$$

where $r(x)$ is bounded, which is well known from Calculus. In our new structure we now have infinitely small quantities like $d = (0, 1)$ at our disposal, so one could be tempted to set $\Delta x = d$. Because of $d^2 = 0$, the relationship then reads

$$f(x + d) = f(x) + d \cdot f'(x) \tag{18}$$

It can be shown that the intuitive reasoning presented here is indeed justified, at least for a large class of functions f [17].

Now using the rules for the new arithmetic on ordered pairs, we infer

$$f((x, 1)) = f(x + d) = f(x) + d \cdot f'(x) = (f(x), f'(x)) \tag{19}$$

and this is the same result as above.

We restrict ourselves to these rough sketches. We note that the differential algebra discussed here can be generalized in a very fundamental way to become a field (in which all divisions by nonzero quantities are allowed), and one can introduce calculus on this field in a similar manner as on the original real numbers. However, the field now contains infinitely small and large quantities, and for example allows a direct treatment of delta functions [17]. So there is a similarity to the not so direct methods of nonstandard analysis [18,19].

Conceptually, the computation of Taylor series maps from a tracking code is now a quite straightforward procedure: one just has to replace all operations in the tracking section, including numerical integrators or whatever there might be, by the corresponding ones in differential algebra, and one automatically obtains all the partial derivatives.

The Differential Algebra Package

In practice, the approach suggested here requires an efficient implementation of the differential algebra and related operations. While the implementation of the operations for a fixed low order is rather simple, the implementation to arbitrary order and for an arbitrary number of variables is quite involved and requires sophisticated logistic algorithms.

The operations on the differential algebra discussed in the last section have been programmed in FORTRAN by the author. The package also contains a complete memory management tool to store all differential algebraic quantities. From the user's side, all differential algebraic objects are just pointers to locations within a large memory area.

The most important features of this package being order and variable independence, a rigid storing scheme for the derivatives is prohibited, and so before any of the operations can be performed, the package has to be initialized by telling it a certain maximum order and number of variables.

Another important criterion in the development of the package was speed: using a sophisticated addressing technique logistic overhead is kept to a minimum. To be more specific, typically about 70 percent of the computing time is spent on actual real arithmetic.

In practice it frequently happens that vectors contain many zeros. For example, this is the case if a certain quantity does not depend at all on one or more of the independent variables. To gain speed in these cases, a compression technique was developed which maps away all the zero coefficients, such that actual execution time only depends on the number of nonzero entries.

Because FORTRAN does not allow the introduction of arbitrary data types and operations, all of the fundamental operations are cast into subroutine calls. The author is aware of the advantages other programming languages like C++ offer in this respect; however, virtually all existing beam dynamics codes and tools are written in FORTRAN, so it is not wise to isolate oneself from this world. Furthermore, rumour has it that the new FORTRAN implementation 8x will also allow arbitrary data types.

To facilitate life until these better days, a precompiler [20] has been developed that allows direct declaration of differential algebraic quantities and their use in formulas. It transforms all of the extensions to calls to the appropriate differential algebraic routines and thus generates standard FORTRAN code. We note, however, that the more difficult part, namely the algorithms used for the differential algebra operations themselves, are rather language independent, and other languages have no significant advantage over current FORTRAN.

The Code COSY INFINITY

Despite the fact that the differential algebraic tools described here have been around only for a rather short time, there are already quite a few codes extracting high order maps using the tools. Among older codes, several have been upgraded for map extraction using the differential algebra precompiler. Some of these upgraded codes are TEAPOT [21], THINTRAC [22] and SIXTRAC. Other codes [23,24] were rather specifically designed having differential algebraic map extraction in mind.

The author himself is in the process of completing a generalization of the fifth order code COSY [7,9] to arbitrary order. This very general code allows both thick and thin elements as well as fringe fields. Furthermore, any analytically given or measured field distribution can be used for the extraction of the map. Altogether, the goal was utmost flexibility to allow for the simulation of all conceivable field arrangements.

The code can be used both for tracking and for map extraction with subsequent map manipulations including closed orbit correction, computation of tune shifts and invariants and other quantities of interest.

The input of the code is itself a rather powerful language with many dedicated features for specific purposes, in particular with all differential algebraic operations built in. The syntax of the language is open to allow for easy future extensions.

Acknowledgements

For discussions, encouragement and support, I want to thank Alex Chao, Swapan Chattopadhyay, Etienne Forest, Ed Heighway, Walter Lysenko, and Hermann Wollnik.

References

[1] R. L. Warnock, R. D. Ruth, W. Gabella, and K. Ecklund. Methods of stability analysis in nonlinear mechanics. In *1987 Accelerator Physics Summer School, in print and SLAC Pub 4846, 1989*, AIP Conference Proceedings, 1988.

[2] M. Berz. Differential Algebraic description of beam dynamics to very high orders. *Particle Accelerators*, in print, 1988.

[3] K. L. Brown. *The Ion Optical Program TRANSPORT*. Technical Report 91, SLAC, 1979.

[4] T. Matsuo and H. Matsuda. Computer program TRIO for third order calculations of ion trajectories. *Mass Spectrometry*, 24, 1976.

[5] H. Wollnik, J. Brezina, and M. Berz. GIOS-BEAMTRACE, a program for the design of high resolution mass spectrometers. In *Proceedings AMCO-7*, Darmstadt, 1984.

[6] A. J. Dragt, L. M. Healy, F. Neri, and R. Ryne. MARYLIE 3.0 - a program for nonlinear analysis of accelerators and beamlines. *IEEE Transactions on Nuclear Science*, Ns-3,5:2311, 1985.

[7] M. Berz, H. C. Hofmann, and H. Wollnik. COSY 5.0, the fifth order code for corpuscular optical systems. *Nuclear Instruments and Methods*, A258:402, 1987.

[8] 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.

[9] H. Wollnik, B. Hartmann, and M. Berz. Principles behind GIOS and COSY. *AIP Conference Proceedings*, in print, 1988.

[10] M. Berz. The description of particle accelerators using high order perturbation theory on maps. In *1987 Accelerator Physics Summer School, in print*, AIP Conference Proceedings, 1988.

[11] M. Berz. Differential Algebraic treatment of beam dynamics to very high orders including applications to spacecharge. *AIP Conference Proceedings*, in print, 1988.

[12] M. Berz. Differential Algebraic description and analysis of trajectories in vacuum electronic devices including spacecharge effects. *IEEE Transactions on Electron Devices*, 35-11, 1988.

[13] M. Berz. The method of power series tracking for the mathematical description of beam dynamics. *Nuclear Instruments and Methods*, A258:431, 1987.

[14] J. F. Ritt. *Differential Algebra.* American Mathematical Society, Washington, D.C., 1950.

[15] P. Du Bois-Reymond. *Mathematische Annalen*, 11:, 1877.

[16] O. Stolz. *Mathematische Annalen*, 39:, 1891.

[17] M. Berz. *Analysis auf einer nichtarchimedischen Erweiterung der reellen Zahlen.* Report (in German), Universität Gießen, 1988.

[18] D. Laugwitz. Ein Weg zur Nonstandard-Analysis. *Jahresberichte der Deutschen Mathematischen Vereinigung*, 75:66, 1973.

[19] A. Robinson. Non-standard analysis. In *Proceedings Royal Academy Amsterdam, Series A*, page 432, 1961.

[20] M. Berz. *The Differential Algebra FORTRAN precompiler DAFOR.* Technical Report AT-3:TN-87-32, Los Alamos National Laboratory, 1987.

[21] L. Schachinger and R. Talman. TEAPOT, a thin element program for optics and tracking. *Particle Accelerators*, 22:35, 1987.

[22] B. T. Leeman and E. Forest. *Brief Description of the tracking codes THINTRACK and FASTRACK.* Technical Report SSC-133, SSC Central Design Group, Berkeley, Ca, 1988.

[23] E. Forest and H. Nishimura. these proceedings.

[24] S. Peggs and J. Irwin. these proceedings.