Nuclear Instruments and Methods in Physics Research A298 (1990) 473–479
North-Holland

Computational aspects of optics design and simulation: COSY INFINITY

Martin Berz

Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, USA, and Exploratory Studies Group, Lawrence Berkeley Laboratory, Berkeley, CA 94720, USA

The new differential algebraic (DA) techniques allow very efficient treatment and understanding of nonlinear motion in optical systems as well as circular accelerators. To utilize these techniques in their most general way, a powerful software environment is essential. A language with structure elements similar to Pascal was developed. It has object oriented features to allow for a direct utilization of the elementary operations of the DA package. The compiler of the language is written in Fortran 77 to guarantee wide portability.

The language was used to write a very general beam optics code, COSY INFINITY. At its lowest level, it allows the computation of the maps of standard beam line elements including fringe fields and system parameters to arbitrary order. The power of the DA approach coupled with an adequate language environment reveals itself in the very limited length of COSY INFINITY of only a few hundred lines. Grouping of elements as well as structures for optimization and study are readily available through the features of the language. Because of the openness of the approach, it offers a lot of power for more advanced purposes. For example, it is very easy to construct new particle optical elements. There are also many ways to efficiently manipulate and analyze the maps.

1. Introduction

The quest for aberrations of optical systems is as old as optics itself. In practice, however, it has so far been possible to obtain accurate values for aberrations of particle optical systems only to relatively low orders and only for very special fields. There are several third order codes [1–5] and one fifth order code [6] that allow the computation of aberrations of optical systems. The DA techniques discussed in another paper in these proceedings [7] and in previous papers [8–10] for the first time provide a concise and powerful framework that allows the computation of arbitrary order maps depending on arbitrarily many variables including system parameters. So DA techniques are the tool to a general yet elegant unified theory of optical aberrations.

The usefulness of the concept in practice is intimately related to the need for powerful new software algorithms and strategies. In this paper, we want to discuss some of the application oriented challenges associated with the DA method. In section 2, we shall discuss briefly some efficient algorithms for DA operations, a problem which is at the core of practical usability of DA and requires quite sophisticated programming techniques. Section 3 contains general consideration about efficient use of DA operations and new generation beam dynamics codes. Section 4 discusses the input language of COSY INFINITY, which is so powerful that indeed all the physics of the code was written in it. The last section finally discusses the physics aspects of the implementation.

2. The fundamental DA operations

In this section we will briefly outline the techniques and algorithms used in the DA package. This package is a collection of subroutines performing the elementary DA operations addition, scalar multiplication, multiplication and derivation. It also contains a large selection of intrinsic functions for DA objects, including most intrinsic functions usually available in a Fortran environment. Furthermore, there is a substantial collection of higher level DA tools. Altogether, the whole DA package contains about 8000 lines of Fortran code, and thus represents a major part of the whole investment of writing COSY INFINITY.

2.1. The storage of DA vectors

In this section we will discuss the storage and coding of DA vectors inside the package. The package contains its own memory management, which stores all data in one large common block, and refers to them by pointers. The size of the block can be changed according to the hardware environment, and this limits the amount and size of DA vectors that can be used. Thus the whole problem of the static memory allocation of Fortran is

0168-9002/90/$03.50

removed except for the need to adjust the parameter to the maximum amount of regular or virtual memory available.

In practice it turns out that many DA vectors are almost empty in the sense that many of the derivatives do not occur. For example, a system that has mid-plane symmetry entails that right away half of the possible derivatives do not occur. If in addition there are system parameters, often many intermediate vectors do depend on only one or two of these, further reducing the number of terms.

For this reason, the storage of the DA vectors is done in such a way that only nonzero information is preserved. In the most general case of the algebraically closed version of the DA, vectors are represented by their support points in Q^v (cf. ref. [7]) and the double precision values that they assume there. This information is coded in the following way:

For a given DA vector, first the smallest common denominator i_d of all the rational numbers in its support points in Q^v (cf. ref. [7]) is determined and all the fractions are expressed in terms of this denominator. (Because of the nature of operations, denominators are usually small in practice, for example 2, 3 or 4.) In the second step, the smallest numerator $\imath_n$ is determined. The other numerators are then expressed by their (positive) difference to i_n. In the special case of elements of the non-extended DA ${}_nD_v$ (cf. ref. [7]), we would have $i_d = 1$ and $\imath_n = 0$.

Given $\imath_d$ and $\imath_n$ for a certain DA vector, the information describing the vector at one support point is thus described by a double precision number and v integers. The v integers are stored in a bit-packed representation and thus compressed into two coding integers.

Altogether, a DA vector is described by $\imath_d$, $\imath_n$, the depth through which support points are known, and the collection of support points, coded by one double precision number and two integers each. The collection of coded support points is stored in a prespecified order that is discussed in connection with the multiplication. All these data are stored dynamically in the large common block and are managed by pointers.

2.2. The elementary operations

The addition of two DA vectors represented by the above coding is a relatively transparent merging process. First, the two vectors are expressed in terms of one common denominator and one common offset. Then, the entries in the vector are run through by two pointers, at each step comparing the corresponding coding integers. If they match, the two double precision values at the respective support point are added. Otherwise, the double precision value of the support point and the two coding integers of the support point coming first in the ordering are copied.

The scalar multiplication is even simpler; it only requires the multiplication of all the double precision values with a constant. The derivations are also relatively simple. They only require a change in the two coding integers; not even a reordering is necessary because of the particular choice of the ordering.

The major challenge regarding the elementary operations was the multiplication, and it required a highly sophisticated algorithm to implement it efficiently. Due to the particular, not very intuitive form of the ordering, the total overhead for bookkeeping is limited to only 30% of the time for the required double precision multiplications. Other less sophisticated algorithms can easily increase the time for a vector multiplication by a factor of 10. For details, we refer to the code description.

2.3. Higher level utilities

After having discussed the four elementary DA operations, we now want to address higher level algorithms. The most important and sophisticated of these operations is the concatenation of transfer maps. In essence, this boils down to an optimum tree transversal problem utilizing rebranching. Together with the multiplication algorithm, this is the most difficult program in the code. We note that using the object oriented features of the COSY language discussed below, the concatenation routine can also be used to push particles through a transfer map in the most efficient way.

Using the concatenator, it is not very difficult to construct routines that allow the partial inversion discussed in section 4.2 of ref. [7]. This readily allows the computation of the various Eikonals, and the implementation of DA optimization algorithms using superconvergent Newton–Raphson methods. It further allows the implementation of various symplectic integration techniques. For details, we refer to ref. [11].

Furthermore, there is a whole Lie algebraic environment in COSY INFINITY. Since any multidimensional DA is a Lie Algebra via a Poisson bracket based on the derivations of the DA, this for the first time allows high order and parameter dependent Lie algebraic operations. Because of the central role played by the Poisson bracket, there is a direct Poisson bracket routine not explicitly based on derivations and multiplications that allows a savings of about 30%. Part of the Lie Algebraic environment are routines that allow the computation of potentials for DA fields to allow the computation of the various Lie operator factorizations discussed in section 4.3 of ref. [7].

Finally, there is a large variety of other support routines, including routines that simplify the interfacing of other DA programs with different representations. There are also routines for the efficient extraction of

individual coefficients from a DA vector, and there are input/output routines and various diagnostic routines.

In this section we have discussed briefly some of the algorithmic problems associated with DA implementations. We have discussed algorithms for the important operations to arbitrary order and in arbitrarily many variables. The next sections will address the efficient utilization of these tools for practical work.

3. The philosophy of COSY INFINITY

A modern tool that allows the design and simulation of optical systems has to fulfil a wide variety of requirements. On the technical side, it should be widely portable. Portability should range from personal computers for simple first order layout to supercomputers for long term tracking in large hadron colliders.

Even though programming languages have come a long way in the past decade, there still is only one numerics oriented language that is truly portable: Fortran. The choice for this language, with all its bad reputation among serious computer scientists, is even more obvious when considering the potential need to interface with a variety of existing programs for various special problems, which historically in our field have all been written in Fortran.

The code itself should be both easy to use and learn, yet allow the knowledgeable expert user utmost flexibility. There are various concepts that have evolved in the past, beginning with TRANSPORT's number tables, continuing with second generation mnemotechnic command languages like in GIOS, MARYLIE and COSY, and finally the third generation unification attempts of MAD.

All of these languages have command structures for various tasks, but are still often not as simple to use and not as flexible as one might desire. However, what all these approaches really boil down to is exactly what computer scientists are attempting to streamline: the interaction with a computer, preferably in a simple yet powerful way. In this view, programming languages are nothing but attempts to provide concepts of thinking that allow a very concise yet powerful phrasing of tasks. So computer science already has the answer to the quest for the optimal command language: take a regular programming language!

Using a regular language, it is very directly possible to express very complicated tasks in a relatively compact way; yet, if the syntax of the language is not too complicated, it is easy to learn. Certain tasks in this language would be expressed by procedure calls. Optical elements themselves would be procedure calls. New optical elements can be written as new procedures.

The language of choice has to be modern and concise, yet portable. Furthermore, it should allow the use of DA in a direct way to allow the transformation of conceptual simplicity of the DA approach into coding simplicity.

4. The object oriented COSY language

Unfortunately, a language that satisfies the above statements does not readilv exist. It would require the portability of Fortran, the compactness and beauty of Pascal, and the object oriented features of C + +. Furthermore, it should not require time consuming linking of the user input to the existing code.

So in order to provide the proper DA environment for COSY INFINITY and the user commands, we designed our own language system. The syntax is very close to Pascal. This allows both a quick understanding of the syntax for the user and limits the complexity of the compiler for the programmer. Many features of Pascal are very useful for the practical programming of COSY INFINITY. In particular, the local and global routines and variables allow a very clean and efficient data management.

For compatibility, the compiler is written in Fortran, and the language is not compiled into machine code but into a metacode that is again interpreted by a Fortran driver. So Fortran serves here merely as a machine independent assembly language. Because of the simplicity of the Pascal syntax, the programming of the compiler was not as difficult as it may appear. The code for the compiler and the interpreter together have a length of about 4000 lines. The code is also very efficient: In a recent benchmark test against a commercially available Pascal compiler, our compiler was only about 30% slower.

The COSY language is object oriented; it is straightforward to add new types to it, and in particular, we have DA now readily available. Also other types like interval arithmetic that will prove valuable in the future are available. Altogether, the COSY language provides a very powerful environment for the handling of DA vectors.

In the COSY language, types of variables are free until runtime. This provides a very simple yet powerful way to call the same procedure once with real arguments and once with DA arguments. How valuable this approach is for COSY becomes apparent in that all of the chromatic effects can now simply be turned on or off by making the rigidities real or DA variables. In a very similar way, the freedom of types automatically allows the computation of maps depending on system parameters.

The procedure structures in the COSY language allow the grouping of elements into beamlines in a rather simple way. A line is nothing but a procedure which contains calls to element procedures or other line

procedures. The logical structures including loops and while and if statements allow the programming of complicated optimization strategies.

The latter is also directly supported by one feature with which the COSY language goes beyond normal programming languages. It contains provisions for optimization that are directly part of the language. There is an optimization structure similar to the loop structure that runs through a block of code over and over again until a target variable has been minimized by varying a set of specified free variables. A variety of optimization algorithms are available, including the very rugged simplex optimizer. The most elegant of these optimizers is probably the superconvergent Newton's method which computes the derivatives of the objective function by replacing the fitting variables with DA.

Using the COSY language combined with the DA methods discussed in ref. [7] and the implementation of the DA package discussed above, the implementation of the arbitrary order code COSY INFINITY with all its flexibility could be achieved with only a few hundred lines of COSY commands.

This section can necessarily only give a limited impression of COSY INFINITY and its power. More detailed descriptions can be found in ref. [12], but the best way is to play with it. The code is distributed by the author, and requests for it should be sent to the above address.

5. The physics in COSY INFINITY

In this section, we want to discuss the physics implemented in COSY INFINITY. Using the DA techniques and the powerful operating environment discussed in the previous sections, it proved rather straightforward to produce a very flexible code that allows map computation as well as particle tracking in a variety of ways. The resulting code is very compact and relatively transparent.

The power of this new approach allows us in particular to avoid many approximations in the physics part. For spectrometers, this means that with limited extra effort, we are able to treat fringing fields exactly and not only in the perturbative fringing field integral technique. It also allows the computation of transfer maps of very complicated field arrangement to very high order.

For the accelerator physicist, it allows the use of the proper Hamiltonian without any approximation of the square root. Tracking through maps, it allows the use of thick elements and even fringe fields instead of kicks. At its simplest level, this can avoid the notorious refitting of the linear tune, but in particular for small machines, it may reveal important physical effects.

In the following subsection we want to discuss the full relativistic equations of motion in the specific coordinates used in COSY INFINITY. These equations of motion are used for the numerical integration of maps as well as particles, for the DA integration of maps, and for the kick implementation.

5.1. The canonical equations of motion

The motion is described relative to the trajectory of a reference particle. In these coordinates, the transfer map is always origin preserving and thus nilpotent (compare ref. [7]). While the reference trajectory is allowed to bend, it is supposed to stay within a common plane. A place in space is described uniquely by an arc length position s, and the function $h(s)$, which describes the relative curvature, i.e. the reciprocal of the radius of curvature $\rho(s)$.

In particular, the total angle by which the trajectory has been bent since $s = s_0$ is given by

$$\alpha = \int_{s_0}^{s} h \, \mathrm{d}s. \tag{1}$$

We now use this formula to determine the equations of motion which in Cartesian coordinates have the relativistic form

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{P} = \boldsymbol{F}, \quad \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r} = \frac{\boldsymbol{P}}{\sqrt{1 + P^2/m^2c^2}}. \tag{2}$$

We transform the equations for the momenta to integral equations by integrating with respect to the independent variable s and obtain

$$\boldsymbol{P}(s) = \boldsymbol{P}(s_0) + \int_{t(s_0)}^{t(s_1)} \boldsymbol{F}(t) \, \mathrm{d}t = \boldsymbol{P}(s_0) + \int_{s_0}^{s} \boldsymbol{F}(s) t' \, \mathrm{d}s, \tag{3}$$

where $t' = \mathrm{d}t/\mathrm{d}s$. In the new local variables determined by the reference trajectory, the momenta thus have the form

$$\boldsymbol{p} = \begin{pmatrix} \cos\left(\int_{s_0}^{s} h \, \mathrm{d}s\right) & 0 & \sin\left(\int_{s_0}^{s} h \, \mathrm{d}s\right) \\ 0 & 1 & 0 \\ -\sin\left(\int_{s_0}^{s} h \, \mathrm{d}s\right) & 0 & \cos\left(\int_{s_0}^{s} h \, \mathrm{d}s\right) \end{pmatrix} \cdot \left(\boldsymbol{P}(s_0) + \int_{s_0}^{s} \boldsymbol{F}(s) t' \, \mathrm{d}s \right). \tag{4}$$

Differentiating the expressions in eq. (4) with respect to s now yields the differential equation for the new momenta:

$$\boldsymbol{p}' = \boldsymbol{F}t' + (0, h, 0) \times \boldsymbol{p}. \tag{5}$$

Here the prime denotes differentiation with respect to the new independent variable, the arc length s.

We note that for a particle at position s with coordinate x, the slope is given by p_x/p_z. Now suppose a particle moves a small Δs. Then because of the curvature of the optical axis, the actual distance it travels parallel to the reference trajectory is $(1+hx)\,\Delta s$ to first order in Δs, and hence its slope is also given by $\Delta x/[(1+hx)\,\Delta s]$, from which we infer

$$x' = (1+hx)\frac{p_x}{p_z}, \qquad y' = (1+hx)\frac{p_y}{p_z}. \tag{6}$$

The time Δt it takes to travel from s to $s+\Delta s$ is given by the distance divided by the velocity, and thus to first order in Δs we obtain the relationship

$$\Delta t = \frac{1}{v}\sqrt{\Delta s^2(1+hx)^2 + \Delta x^2 + \Delta y^2},$$

from which we conclude using eqs. (6):

$$t' = (1+hx)\frac{1}{v}\sqrt{1+\frac{p_x^2+p_y^2}{p_z^2}} = (1+hx)\frac{1}{v}\frac{p}{p_z}. \tag{7}$$

We now will derive the equations of motion for the case of the motion in electromagnetic fields, for which $\boldsymbol{F} = ze(\boldsymbol{E} + \boldsymbol{v}\times\boldsymbol{B})$.

As coordinates describing the motion of a particle we choose the following set:

$$\begin{aligned} r_1 &= x, & r_2 &= a = p_x/p_0,\\ r_3 &= y, & r_4 &= b = p_y/p_0,\\ r_5 &= l = v_0(t-t_0), & r_6 &= \delta_K = (K-K_0)/K_0,\\ r_7 &= \delta_m = (m-m_0)/m_0, & r_8 &= \delta_z = (z-z_0)/z_0, \end{aligned} \tag{8}$$

where x, y, p_x and p_y are the positions and momenta in the moving frame of reference, K is the initial energy, and m and z are the mass and charge of a particle, respectively. The index 0 denotes the respective quantity of the reference particle. Altogether, the coordinates are such that the reference particle is described by the origin, and furthermore the quantities are normalized such that they are usually significantly smaller than 1 in absolute value.

We note that the momentary kinetic energy K is given by

$$K = K_0(1+\delta_k) - z_0(1+\delta_z)eV(x, y, s), \tag{9}$$

where V is the electrostatic potential. It turns out to be particularly advantageous to introduce the following measure of relativity:

$$\eta = \frac{K}{mc^2}. \tag{10}$$

Because of $K = mc^2[(1-v^2/c^2)^{-1/2} - 1]$, we obtain

$$(1-v^2/c^2)^{-1/2} = 1+\eta,$$

and from this

$$\frac{v}{c} = \frac{\sqrt{\eta(2+\eta)}}{1+\eta}. \tag{11}$$

From eq. (11) and $\boldsymbol{p} = m\boldsymbol{v}/(1-v^2/c^2)^{1/2} = m\boldsymbol{v}(1+\eta)$, we infer

$$\frac{p}{mc} = \sqrt{\eta(2+\eta)}. \tag{12}$$

Eqs. (11) and (12) allow a direct determination of velocity and momentum from the quantity η. Note that contrary to many other relativistic relationships, the evaluation of these formulas is always numerically stable since no differences of small quantities occur in the limits $\eta \to 0$ or $\eta \to \infty$.

From eqs. (11) and (12), we also obtain

$$\frac{p}{v} = m(1+\eta). \tag{13}$$

We begin the derivation of the differential equations in the new coordinates with the equation for l'. We obtain from eq. (7) using eq. (13):

$$\begin{aligned} l' = v_0 t' &= (1+hx)\frac{p/v}{p_0/v_0}\frac{p_0}{p_z}\\ &= (1+\delta_m)(1+hx)\frac{1+\eta}{1+\eta_0}\frac{p_0}{p_z}. \end{aligned} \tag{14}$$

Note that the coordinates in eq. (8) do not contain the momentum in the z direction, but instead the energy spread d. Using eq. (12), we determine p_z/p_0 from the other coordinates as

$$\begin{aligned} \frac{p_z}{p_0} &= \sqrt{\frac{p^2}{p_0^2} - a^2 - b^2}\\ &= \sqrt{(1+\delta_m)^2\frac{\eta(2+\eta)}{\eta_0(2+\eta_0)} - a^2 - b^2}. \end{aligned} \tag{15}$$

Using $\boldsymbol{v}/v = \boldsymbol{p}/p$ (which merely says that velocity and momentum are parallel) and eqs. (7) and (14), we obtain from eq. (5):

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}s}\left(\frac{p_x}{p_0}, \frac{p_y}{p_0}, \frac{p_z}{p_0}\right)\\ &\quad = ze\boldsymbol{E}\frac{t'}{p_0} + ze\frac{\boldsymbol{v}}{vp_0}\times\boldsymbol{B}(1+hx)\frac{p}{p_z} + h\left(\frac{p_z}{p_0}, 0, -\frac{p_x}{p_0}\right)\\ &\quad = \left(\frac{\boldsymbol{E}}{\chi_{E_0}}l' + \frac{\boldsymbol{p}}{p_0}\times\frac{\boldsymbol{B}}{\chi_{M_0}}(1+hx)\frac{p_0}{p_z}\right)(1+\delta_z)\\ &\qquad + h\left(\frac{p_z}{p_0}, 0, -\frac{p_x}{p_0}\right)\\ &\quad = \left((1+\delta_m)\frac{1+\eta}{1+\eta_0}\frac{p_0}{p_z}\frac{\boldsymbol{E}}{\chi_{E_0}}\right. \end{aligned}$$

$$+\left(a\frac{p_0}{p_z},\, b\frac{p_0}{p_z},\, 1\right)\frac{\boldsymbol{B}}{\chi_{M_0}}\Bigg)(1+\delta_z)(1+hx)$$
$$+h\left(\frac{p_z}{p_0},\, 0,\, -\frac{p_x}{p_0}\right). \tag{16}$$

Here the quantities

$$\chi_{E_0}=\frac{p_0 v_0}{z_o e} \quad \text{and} \quad \chi_{M_0}=\frac{p_0}{z_0 e} \tag{17}$$

have been used. These quantities depend on the reference particle and present scaling factors that determine how strongly the electric and magnetic fields affect the motion. They are usually referred to as the electric and magnetic rigidities.

Thus we finally obtain the total set of equations of motion:

$$x'=a(1+hx)\frac{p_0}{p_z},$$

$$y'=b(1+hx)\frac{p_0}{p_z},$$

$$l'=(1+\delta_m)(1+hx)\frac{1+\eta}{1+\eta_0}\frac{p_0}{p_z},$$

$$a'=\left((1+\delta_m)\frac{1+\eta}{1+\eta_0}\frac{p_0}{p_z}\frac{E_x}{\chi_{E_0}}-\frac{B_y}{\chi_{M_0}}+b\frac{p_0}{p_z}\frac{B_z}{\chi_{M_0}}\right)$$
$$\times(1+hx)(1+\delta_z)+h\frac{p_z}{p_0},$$

$$b'=\left((1+\delta_m)\frac{1+\eta}{1+\eta_0}\frac{p_0}{p_z}\frac{E_y}{\chi_{E_0}}+\frac{B_x}{\chi_{M_0}}-a\frac{p_0}{p_z}\frac{B_z}{\chi_{M_0}}\right)$$
$$\times(1+hx)(1+\delta_z), \tag{18}$$

where χ_{M_0}, and χ_{E_0} are the rigidities defined in eqs. (17), and the following expressions have been used:

$$\eta=\left(\frac{K_0(1+\delta_k)-z_0 e(1+\delta_z)V(x,\,y,\,s)}{m_0c^2(1+\delta_m)}\right), \tag{19}$$

$$\frac{p_z}{p_0}=\sqrt{(1+\delta_m)^2\frac{\eta(2+\eta)}{\eta_0(2+\eta_0)}-a^2-b^2}\,. \tag{20}$$

These equations of motion describe the pure Hamiltonian motion in the electromagnetic field. There are two effects which are often relevant for particle accelerators not included in these equations.

The first effect is synchrotron radiation which leads to a gradual loss in particle energy. Even though radiation is a quantum effect occurring stochastically, often it suffices to only study the average radiation loss which is simply proportional to the momentary curvature of the trajectory.

The second effect that is relevant is the spin. While the forces exerted on the particle due to the interaction of the magnetic moment with a field gradient are usually small and can be neglected in a very good approximation, the dynamics of the spin itself, which is of importance for the study of polarization, can be treated in the same way as the other quantities.

It is probably one of the strongest points of the DA approach that the whole computational effort to obtain maps to arbitrary order is only limited by the programming of the equations of motion. While we do not have spin and radiation treatment in COSY INFINITY now, it should be apparent that their inclusion is rather straightforward.

5.2. The description of the fields

In order to utilize the equations of motion in particle optical coordinates, we need to know the specific form of the electric and magnetic fields in the curvilinear coordinates. Since both the electric and magnetic fields are rotation-free, there are scalar potentials V for the fields. We write V as a power series expansion in the curvilinear coordinates:

$$V(x,\,y,\,s)=\sum_{\iota=0}^{\infty}\sum_{\jmath=0}^{\infty}a_{\iota,\jmath}(s)\frac{x^\iota y^\jmath}{\iota!\jmath!}. \tag{21}$$

Since the electric and magnetic fields are divergence free, we can conclude $\nabla^2 V=0$. In the curvilinear coordinates, this condition assumes the well-known form [1,13,14]:

$$\nabla^2 V_B=\frac{1}{1+hx}\frac{\partial}{\partial x}\left((1+hx)\frac{\partial V_B}{\partial x}\right)+\frac{\partial^2 V_B}{\partial y^2}$$
$$+\frac{1}{1+hx}\frac{\partial}{\partial z}\left(\frac{1}{1+hx}\frac{\partial V_B}{\partial z}\right)=0. \tag{22}$$

Inserting eq. (21) into eq. (22), one obtains the following recursion relation:

$$a_{\iota,\jmath+2}=-a''_{\iota,\jmath}-iha''_{\iota-1,\jmath}+\iota h'a'_{\iota-1,\jmath}-a_{\iota+2,\jmath}$$
$$-(3\iota+1)ha_{\iota+1,\jmath}-3\iota ha_{\iota-1,\jmath+2}$$
$$-\iota(3\iota-1)h^2a_{\iota,\jmath}-3i(\iota-1)h^2a_{\iota-2,\jmath+2}$$
$$-i(\iota-1)^2h^3a_{\iota+1,\jmath}$$
$$-i(i-1)(i-2)h^3a_{\iota-3,\jmath+2}, \tag{23}$$

where primes denote derivatives with respect to z, and it is understood that all coefficients with negative indices are zero.

This recursion relation allows us to compute all $a_{\iota,\jmath}$ from the ones with $\jmath=0$ and $\jmath=1$. So the potentials are determined by their behavior in the plane of deflection and the first derivative with respect to y. In the electric case, we obtain $a_{\iota,1}=0$, so the $a_{\iota,0}$ and thus the knowledge of E_x in the midplane provide complete information. In the magnetic case, we have $a_{\iota,0}-0$, such that the $a_{\iota-1}$ and hence the knowledge of B_y in the midplane provide complete information.

In case of s independence, the relations considerably simplify to

$$a_{i,j+2} = -a_{i+2,j} - ia_{i-1,j} - (i+1)a_{i+1,j}. \quad (24)$$

In the equations of motion, these fields have to be evaluated at (x, y, s). Because the transfer map in the curvilinear coordinates preserves the origin, we obtain that x and y are always infinitesimal (cf. ref. [7]), and thus powers higher than v vanish. This entails that in order to compute the map to order v, it is sufficient to know the power series expansion of the fields to order v only.

5.3. Numerical integration, DA integration and kicks

Using the differential equations and the fields given in the last two sections, it is now possible to compute transfer maps to arbitrary order. To this end, we programmed a seventh order Runge Kutta algorithm with automatic step size control [15] in the COSY language. This integrator is mainly used for special elements like round lenses and for checking purposes.

There is also a DA integrator [7] for more standard fields, including the main fields of standard beamline elements and most fringe fields. This integrator uses a fixed step size and adjusts the order to obtain the requested precision. A typical element is transversed in one step, using orders of about 25, which gives an accuracy close to machine precision. Since this integrator requires only one DA evaluation of the fields per step, it is more than one order of magnitude faster than the usual integration. In this case, the time it takes to compute a map to third order is very similar to the time required by TRANSPORT [1] or any other code.

The equations of motion are also used for a kick environment, which is essentially a second order integrator. Even though the accuracy of this technique is not very high, this approach has been used extensively in the simulation of particle accelerators.

Because of the freedom of types in the COSY environment, the code for the integrators can be used both for the computation of arbitrary order maps and the numerical integration of trajectories. Using the special vector data type, many particles can be tracked simultaneously, which entails perfect vectorization and is attractive for long term tracking in hadron accelerators.

References

[1] K.L. Brown, The ion optical program TRANSPORT, Technical Report 91, SLAC (1979).

[2] T. Matsuo and H. Matsuda, Mass Spectrom. 24 (1976).

[3] H. Wollnik, J. Brezina and M. Berz, Proc. AMCO-7, Darmstadt, 1984, p. 679.

[4] H. Wollnik, B. Hartmann and M. Berz, AIP Conf. Proc. 177 (1988) p. 74.

[5] A.J. Dragt, L.M. Healy, F. Neri and R. Ryne, IEEE Trans. Nucl. Sci. NS-3 (5) (1985) 2311.

[6] M. Berz, H.C. Hofmann and H. Wollnik, Nucl. Instr. and Meth. A258 (1987) 402.

[7] M. Berz, these Proceedings (3rd Int. Conf. on Charged Particle Optics, Toulouse, France, 1990) Nucl. Instr. and Meth. A298 (1990) 426.

[8] M. Berz. Part. Accel. 24 (1989) 109.

[9] M. Berz, IEEE Trans. Electron Devices ED-35 (11) (1988) 2002.

[10] M.Berz, AIP Conf. Proc. 177 (1988) p. 275.

[11] M. Berz, in: Nonlinear Problems in Future Accelerators (World Scientific, 1990) in press.

[12] M. Berz, COSY INFINITY, a reference manual. Technical Report 28881, Lawrence Berkeley Laboratory, Berkeley, CA (1990).

[13] L.C. Teng, Expanded form of magnetic field with midplane symmetry, Technical Report ANL-LCT-28, ANL (1962).

[14] M. Berz, Verallgemeinerung der nichtlinearen Teilchenoptik auf Bildfehler bis fünfter Ordnung, PhD thesis, Justus Liebig Universität Giessen, 6300 Giessen, FRG (1986).

[15] Ingolf Kübler, Master's thesis, Justus Liebig Universität Giessen, 6300 Giessen, FRG (1987).