COSY INFINITY 10.2
Beam Physics Manual

MSU Report MSUHEP20221202

M. Berz and K. Makino
Michigan State University

April 2023

Contents

1 Before Using COSY INFINITY

1.1 User's Agreement

COSY INFINITY can be obtained from MSU under the following conditions.

Permitted Uses: Michigan State University ("MSU") grants you, as "End User," the right to use COSY INFINITY for non-commercial purposes only. Registered users will automatically be given access to updates of the code as they become available. Conversely, we encourage end users to make available tools of sufficient generality they develop for the purpose of inclusion in the master version. Any errors detected in COSY INFINITY should be reported; comments to improve its performance are appreciated. If the code proves useful for work that is being published, a reference is expected.

Prohibited Uses: End User may not make copies of COSY INFINITY available to others, but rather refer them to register for their own license. End User may not distribute, rent, lease, sub-license, decompile, disassemble, or reverse-engineer the COSY INFINITY materials provided by MSU without the prior express written consent of MSU; or remove or obscure the Board of Trustees of MSU copyright notices or those of its licensors. The source files are provided for purposes of compilation only and should not be modified. We advise against modification of the provided COSYScript libraries so as to maintain a clear upgrade path, but rather to maintain derivative code in separate files.

Intellectual Property: COSY INFINITY is a proprietary product of MSU and is protected by copyright laws and international treaty. This Agreement is a legal contract between you, as End User, and the Board of Trustees of MSU governing your use of COSY INFINITY. MSU retains title to COSY INFINITY. You agree to use reasonable efforts to protect the code from unauthorized use, reproduction, distribution, or publication. All rights not specifically granted in this License Agreement are reserved by MSU.

Warranty: MSU MAKES NO WARRANTY, EXPRESS OR IMPLIED, TO END USER OR TO ANY OTHER PERSON OR ENTITY. SPECIFICALLY, MSU MAKES NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OF COSY INFINITY. MSU WILL NOT BE LIABLE FOR SPECIAL, INCIDENTAL, CONSEQUENTIAL, INDIRECT OR OTHER SIMILAR DAMAGES, EVEN IF MSU OR ITS EMPLOYEES HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL MSU LIABILITY FOR ANY DAMAGES TO END USER OR ANY PERSON EVER EXCEED THE FEE PAID FOR THE LICENSE TO USE THE SOFTWARE, REGARDLESS OF THE FORM OF THE CLAIM.

1.2 How to Obtain Help and to Give Feedback

While this manual and the Programmer's Manual [27] are intended to describe the use of the code as completely as possible, there will probably arise questions that this manual cannot answer. Furthermore, we encourage users to contact us with any suggestions, criticism, praise, or other feedback they may have. We also appreciate receiving COSY source code for utilities users have written and find helpful. We can be contacted at support@cosyinfinity.org.

1.3 How to Install and Run the Code

For Windows, Mac, and some Linux environments, one may simply download the self-installing packages, which are almost completely self-explanatory, from the COSY INFINITY web site,

https://cosyinfinity.org

For full details about the installation and the execution, for questions about installation on different platforms, or for questions about performance, implementation and verification of COSY INFINITY, please refer to the COSY INFINITY Programmer's Manual [27].

2 What is COSY INFINITY

The design and analysis of particle optical systems is quite intimately connected with the computer world. There are numerous more or less widespread codes for the simulation of particle optical systems. Generally, these codes fall into two categories. One category includes ray tracing codes which use numerical integrators to determine the trajectories of individual rays through external and possibly internal electromagnetic fields. The core of such a code is quite robust and easy to set up; for many applications, however, certain important information can not be directly extracted from the mere values of ray coordinates. Furthermore, this type of code is often quite slow and does not allow extensive optimization.

The other category of codes are the map codes, which compute Taylor expansions to describe the action of the system on phase space. These codes are usually faster than integration codes, and the expansion coefficients often provide more insight into the system. On the other hand, in the past the orders of the map, which are a measure of the accuracy of the approach, have been limited to third order [33] [80] and fifth order [18] [32]. Furthermore, traditional mapping codes have only very limited libraries for quite standardized external fields and lack the flexibility of the numerical integration techniques. In particular, fringe fields can only be treated approximately.

2.1 COSY's Algorithms and their Implementation

It is indeed possible to have the best of both worlds: using differential algebraic techniques, any given numerical integration code can be modified such that it allows the computation of Taylor maps for arbitrarily complicated fields and to arbitrary order [14] [6] [9] [3] [2]. An offspring of this approach is the computation of maps for large accelerators where often the system can be described by inexpensive, low order kick integrators.

The speed of this approach is initially determined by the numerical integration process. Using DA techniques, this problem can be overcome too: DA can be used to automatically generate numerical integrators of arbitrary high orders in the time step, yet at the computational expense of only little more than a first order integrator [3] [2]. This technique is very versatile, works for a very large class of fields, and the speeds obtained are similar to those of classical mapping codes.

In order to make efficient use of DA operations in a computer environment, it has to be possible to invoke the DA operations from within the language itself. In traditional languages used for numerical applications like C and Fortran, it is often difficult to introduce new data types and overload the operations on them. Modern object oriented languages like C++ and Fortran 90 on the other hand have the capabilities of conveniently introducing new data types. Consequently, we are providing C++ and Fortran 90 interfaces to COSY INFINITY. Thus the highly optimized COSY tools are widely accessible from nearly every language environment; more details can be found in the Programmer's Manual. Interestingly, the performance of the interfaces is within a factor of two to the regular COSY INFINITY system on most platforms, and significantly outperforms most code natively written in object oriented languages.

However, these languages are not particularly suitable for the formulation of the user's input problem. Indeed, there is the additional difficulty of slow turnaround from completion of input to commencement of execution because of the necessary compiling and liking steps. Thus the user input for COSY INFINITY is written in a simple, high performance scripting environment called COSYScript.

2.2 The COSYScript User Interface

Traditional accelerator codes utilize various kinds of specialized command languages for the description of lattices and beamlines. Various approaches have been used in the past, starting from coding numbers as in the old versions of TRANSPORT [33] over more easily readable command structures like in TRIO [80], GIOS, COSY 5.0 [18] and MARYLIE [39] to the standardized commands of MAD, for which there is a conversion utility to COSY INFINITY (see Section 3.4.1) [58] [59].

COSY INFINITY approaches this problem in a clean way by phrasing the various tasks in terms of a standardized scripting language environment; in fact, the language is so powerful and convenient that all the beam physics modules of COSY INFINITY were written in it.

For ease of use, this language has a deliberately simple syntax. For the user demanding special-purpose features on the other hand, it is rather powerful, much more so than typical lattice description languages. It allows direct and complex interfacing to Fortran routines if needed, and it allows the use of DA and others as built-in types. Finally, it is widely portable, and minimizes turnaround between input completion and commencement of execution, which conventional languages usually do not offer.

For reasons of speed it is helpful to allow the splitting of the program into pieces, one containing the optics program and one the user commands. For this purpose, a complete momentary image of the compilation status is written to a file. When compilation continues with the second portion, this image is read from the file, and compilation continues in exactly the same way as without the splitting.

In Section 7 we provide a one page summary of the scripting language, including an example. The full syntax of COSYScript is described in detail in the Programmer's Manual. However, most of the syntax will become apparent from the detailed examples supplied in the following sections, and experience shows that it is possible to write most COSY INFINITY inputs without explicitly consulting the language reference. In addition, the Programmer's Manual has a handy document, Appendix B "Quick Start Guide for COSY INFINITY", which can be utilized as a quick-start guide.

3 Computing Systems with COSY INFINITY

This section describes some core features of COSY INFINITY's particle optics and accelerator physics environment. This provides the backbone for practical use in particle optics. We assume that the reader has a fundamental knowledge about particle optics, and refer to the literature, for example [92] [54] [36] [60] [88] [49].

3.1 General Properties of the COSYScript Environment

The physics part of COSY INFINITY is written in its own input language. In this context, most commands are just calls to previously defined procedures. If desired, the user can create new commands simply by defining procedures of his own. All commands within COSY INFINITY consist of two or three letters which are abbreviations for two or three words describing the action of the procedure. This idea originated in the GIOS language, and many commands of COSY INFINITY are similar to respective commands in GIOS. All units used in the physics part of COSY INFINITY are SI, except for voltages, which are in kV, and angles, which are in degrees.

Particle optical systems and beamlines are described by a sequence of calls to procedures representing individual elements. The supported particle optical elements can be found in Section 3.3 beginning on page 17; Section 5.7 beginning on page 54 shows how to generate new particle optical elements.

In a similar way, elements can be grouped, which is described in Section 5.3 beginning on page 49. Besides the commands describing particle optical elements, there are commands to instruct the code what to do.

3.2 Control Commands

All user commands for COSY INFINITY are contained in a file which is compiled by FOXY. The first command of the file must be

INCLUDE 'COSY' ;

which makes all the compiled code contained in cosy.fox known to the user input. The user input itself is contained in the procedure RUN. Following the syntax of COSYScript described in the Programmer's Manual [COSYScript, Quick Start Guide for COSY INFINITY], all commands thus have to be included between the statements

PROCEDURE RUN ;

and

ENDPROCEDURE ;

In order to execute the commands, the ENDPROCEDURE statement has to be followed by the call to the procedure,

RUN ;

and the command to complete the COSY INFINITY input file,

END ;

Like any language, the COSYScript environment supports the use of variables and expressions which often simplifies the description of the system. For the declaration of variables, see the Programmer's Manual.

The first command sets up the DA tools and has to be called before any DA operations, including the computation of maps, can be executed. The command has the form

OV <order> <phase space dimension> <number of parameters> ;

and the parameters are the maximum order that is to occur as well as the dimensionality of phase space (1,2 or 3) and the number of system parameters that are requested. If the phase space dimensionality is 1, only the x-a motion is computed; if it is 2, the y-b motion is computed as well, obviously at a slightly higher computation time. If it is 3, the time of flight and chromatic effects are computed also.

The number of parameters is the number of additional quantities besides the phase space variables that the final map shall depend on. This is used in connection with the "maps with knobs" discussed in Section 5.2 on page 48 and to obtain mass and charge dependences if desired, and it is also possible to compute energy dependence without time-of-flight terms at a reduced computational expense.

The order is arbitrary and denotes the maximum order that computations can be performed in. It is possible to change the computation order at run time using the command

CO <order> ;

however, the new order can never exceed the one set in **OV**. Note that the computation time naturally increases drastically for higher orders. Under normal circumstances, orders should not exceed ten very much.

3.2.1 The Coordinates

COSY INFINITY performs all its calculations in the following scaled coordinates:

$$\begin{array}{rclrcl} r_1 & = & x, & r_2 & = & a = p_x/p_0, \\ r_3 & = & y, & r_4 & = & b = p_y/p_0, \\ r_5 & = & l = -(t-t_0)v_0\gamma/(1+\gamma), & r_6 & = & \delta_K = (K-K_0)/K_0, \\ r_7 & = & \delta_m = (m-m_0)/m_0, & r_8 & = & \delta_z = (z-z_0)/z_0. \end{array} \tag{1}$$

The first six variables form three canonically conjugate pairs in which the map is symplectic. The units of the positions x and y are meters. p_0, K_0, v_0, t_0 and γ are the momentum, kinetic energy, velocity, time of flight, and total energy over m_0c^2, respectively. m and z denote mass and charge, and m_0 and z_0 are those of the reference particle.

3.2.2 Defining the Beam

All particle optical coordinates are relative to a reference particle which can be defined with the command

RP <kinetic energy in MeV> <mass in amu> <charge in units> ;

For convenience, there are two procedures that set the reference particle to be protons or electrons:

RPP <kinetic energy in MeV> ;

RPE <kinetic energy in MeV> ;

For the masses of the proton and electron and all other quantities in COSY INFINITY, the values provided by the Berkeley Particle Data Group have been used (**CAUTION:** The data was updated in September 2001 in cosy.fox). Finally, there is a command that allows to set the reference particle from the magnetic rigidity in Tesla meters and the momentum in MeV/c:

RPR <magnetic rigidity in Tm> <mass in amu> <charge in units> ;

RPM <momentum in MeV/c> <mass in amu> <charge in units> ;

Finally it is possible to set the magnetic moments of the particle and activate the computation of spin. This is achieved with the command

RPS < LS > < G > ;

where LS is the spin mode, 1 indicating spin computation and 0 indicating no spin computation. $G = (g-2)/2$ is the anomalous spin factor of the particle under consideration. In case the reference particle has been set to be a proton using **RPP** or an electron using **RPE**, the proper value will be used if G is set to zero.

The command

SB <PX><PA><r12><PY><PB><r34>< PT><PD><r56><PG><PZ> ;

sets half widths of the beam in the x, a, y, b, t, d, g and z directions of phase space as well as the off diagonal terms of the ellipse in TRANSPORT notation r12, r34, and r56. The units are meters for PX and PY, radians for PA and PB, $v_0\gamma/(1+\gamma)$ times time for PT, and $\Delta E/E$ for PD, $\Delta m/m$ for PG, and $\Delta z/z$ for PZ. The command

SP <P1> <P2> <P3> <P4> <P5> <P6> ;

sets the maxima of up to six parameters that can be used as knobs in maps (see Section 5.2 beginning on page 48).

SBE <EX> <EY> <ET> ;

sets the ellipse of the beam to an invariant ellipse of the current map. The emittances in x-a, y-b, and τ-δ space being EX, EY, ET respectively.

3.2.3 The Computation of Maps

COSY INFINITY has a global variable called MAP that contains the accumulated transfer map of the system. Each particle optical element being invoked updates the momentary contents of this global variable.

The following command is used to prepare the computation of maps. It sets the transfer map to the identity. It can also be used again later to re-initialize the map.

UM ;

The command

SM <name> ;

saves the momentary transfer map to the array name, which has to be specified by the user. The array can be specified using the **VARIABLE** command of COSYScript (see the Programmer's Manual [COSYScript]). It could have the form

VARIABLE <name> 1000 8 ;

which declares a one dimensional array with eight entries. Each entry can hold a maximum of 1000 16 byte blocks, which should be enough to store the DA numbers occurring in calculations of at least seventh order. Note that there is a set of convenience functions that allow to easily calculate the required maximal length of objects based on their characteristics; for example, the function LDA determines the length of a DA vector as a function of its order and number of variables.

To copy a map stored in an array name1 to another array name2, use the procedure

SNM <name1> <name2> ;

The command

AM <name> ;

applies the previously saved map <name> to the momentary map. **SM** and **AM** are particularly helpful for the handling of maps of subsystems that are expensive to calculate. In particular in the context of optimization, often substantial amounts of time can be saved by computing certain maps only once and then re-using them during the optimization.

It is also sometimes necessary to compose two individual maps into one map without acting on the current transfer map. This can be achieved with the command

ANM <N> <M> <O> ;

which composes the maps N and M to O=N ∘ M.

The command

PM <unit> ;

prints the momentary transfer map to the output unit. This number can be associated with a file name with the **OPENF** procedure (see index); if **OPENF** is not used, the name associated with the output unit follows the local Fortran conventions. Output unit 6 corresponds to the screen. The different columns of the output belong to the final values of x, a, y, b and t of the map, and different lines describe different coefficients of the expansion in terms of initial values. The coefficients are identified by the last columns which describe the order as well as the exponents of the initial values of the variables. An example of the output of a transfer map can be found in Section 5 on page 47.

The command

PSM <unit> ;

writes the 3×3 spin matrix to the output unit.

Besides the easily legible form of output of a transfer map produced by **PM**, it is also possible to write the map more accurately but less readable with the command

WM <unit> ;

In this case, the transformation of the local coordinate system is also stored and can be reused when read. Maps written by **PM** or **WM** can be read with the command **RM**.

RM <unit> ;

reads a map generated by **PM** or **WM** from the specified unit and applies it to the momentary transfer map. Often a significant amount of computer time can be saved by computing certain submaps ahead of time and storing them either in a variable or a file. In particular this holds for maps which are expensive to compute, for example the ones of electrostatic cylindrical lenses.

Besides storing maps of an element or system with one specific setting of parameters, using the technique of symplectic scaling it is possible to save maps with a certain setting of field strengths and lengths and later re–use them for different settings of lengths or strengths. This is particularly useful for elements that require a lot of calculation time, including fringe fields and solenoids. A representation of the map of an element with typical dimensions and field strength for a typical beam is saved using

WSM < unit> <L> <B> <D> ;

This map has to be calculated either in three dimensions (OV order 3 0 ;) or with the energy as a parameter (OV order 2 1 ;). The parameters are the output unit, length, pole–tip field, and aperture of the element that created the momentary map. The map of the motion of a different type of beam through any similar element that differs in scale or field strength can be approximated quickly by

RSM <unit> <L> <B> <D> ;

It is also possible to extract individual elements of transfer maps. This is achieved with the COSY function

ME(<phase space variable>,<element identifier>)

The element identifier follows TRANSPORT notation; for example, ME(1,122) returns the momentary value of the element (x, xaa).

The beam's current sigma matrix is computed from the ellipse data previously set with SB by the function

SIGMA(<I>,<J>)

Sometimes it is necessary to determine the map of the reversed system, i.e. the system transversed backwards. In case M is the map of the system, the map MR of the corresponding reversed system can be computed with the command

MR <M> <MR> ;

Note again that the current transfer map is stored in the global variable MAP. Similarly, it is sometimes necessary to determine the map of the system in which the coordinates are twisted by a certain angle. For example, if the direction of bending of all magnets is exchanged, this corresponds to a rotation by 180 degrees. In case M is the map of the system, the map MT of the system twisted by angle can be computed with the command

MT <M> <MT> <angle> ;

3.2.4 The Computation of Trajectories

Besides the computation of maps, COSY INFINITY can also trace rays through the system. The trajectories of these rays can be plotted or their coordinates printed. If rays are selected, they are pushed through every new particle element that is invoked. Note that COSY INFINITY can also push rays through maps repetitively and display phase space plots. This uses different methods and is discussed in Section 4.4 beginning on page 41.

The following command sets a ray that is to be traced through the system. The parameters are the values of the eight particle optical coordinates that are defined in eqs. (1) on page 9.

SR $< x > < a > < y > < b >$ <T> <D> <G> <Z> <color> ;

Here x and y are the positions of the ray in meters, a and b are the angles in radians, T is l, the time of flight multiplied by $v_0\gamma/(1+\gamma)$, in meters. D, G and Z are δ_K, δ_m, δ_z, i.e., the energy, mass and charge deviations. For graphics purposes, it is also possible to assign a color. Different colors are represented by numbers as follows. 1: black, 2: blue, 3: red, 4: yellow, 5: green, 6: yellowish green, 7: cyan, 8: magenta, 9: navy, 10: white/background. The command

SSR <X> <Y> <Z> ;

sets the spin coordinates of the particle. Note that command has to be used immediately following the setting of the coordinates of the particle with **SR**. Regardless the input values for X, Y, and Z, the normalized vector of (X,Y,Z) is stored in the system as a spin vector. (0,0,0) is not accepted.

It is also possible to automatically set an ensemble of rays. This can be achieved with the command

ER <NX> <NA> <NY> <NB> <NT> <ND> <NG> <NZ> ;

Here NX, NA ... denote the number of rays in the respective phase space dimension and have to be greater than or equal to 1. The ray coordinates are equally spaced according to the values set with the command **SB**, which has to be called before **ER**. In case any of the N's is 1, only rays with the respective variable equal to 0 will be shown. Note that the total number of rays is given by NX $\cdot$ NA $\cdot \cdots \cdot$ NZ, which should not exceed 200. Note that this command is incompatible with the setting of spin coordinates with **SSR** as described above. The command

SCDE ;

sets sine like and cosine like rays as well as the dispersive ray and the beam envelope in accordance with the data provided by SB or SBE. After the envelope has been set by SCDE it can be displayed alone as it varies along the system with PGE, or together with the other trajectories with PG. If only the envelope should be evaluated,

ENVEL ;

should be used. The closed orbit for an off energy particle, often called the η function, is produced by

ENCL <D> ;

The periodic orbit for an off energy particle with the dispersion D is computed from the one turn map. Therefore a current map has to be produced before calling **ENCL**. This is equivalent to the requirement of computing a current map before calling **SBE**.

The command

CR ;

clears all the rays previously set.

There are a variety of utility commands to handle the coordinates of the rays as well as the spin coordinates associated to them, and they are analogous to those utility commands for handling maps such as **PM**, **WM**, **RM** and so on (see page 11).

The command

PRAY <unit> ;

prints the momentary coordinates of the rays to the specified output unit. Each ray is output in one line, and the first ray in the output is the reference ray. The command

WRAY <unit> ;

prints the momentary coordinates of the rays to the specified output unit. **WRAY** prints all the coordinate values in the full digit length of double precision number expression. First, the x coordinate values of all the rays are output, where the first component corresponds to the reference ray. Next, the a coordinate values are output, followed by y, b,..., δ_z.

Further, the particle survival information through aperture cuts is output by **PRAY** and **WRAY**; see page 42. The "REMAIN" column denotes if the particle has survived (1) or removed (0), and the "REMOVE" column denotes the iteration number when the particle disappeared if removed, otherwise 0.

The command

PR <unit> ;

prints the momentary coordinates of the rays to the specified output unit using the standard output format of the **VE**ctor data type of COSYScript (see the Programmer's Manual [COSY Types]). First, the x coordinate values of all the rays are output, where the first component corresponds to the reference ray. Next, the a coordinate values are output, followed by y, b,..., δ_z. Note that using the **WRITE** command of COSYScript, it is also possible to print any other quantity of interest either to the screen or to a file.

The rays output by **PRAY** or **WRAY** can be read from the specified unit by the command

RRAY <unit> ;

However **RRAY** does not read the ray output by **PR**. Before calling **RRAY**, the command **CR** has to be called. Instead of writing and reading the rays, it is also possible to copy the momentary rays to a specified array. The command

SRAY <name> ;

copies the momentary rays to an array <name>, which has to be specified by the user. The array can be specified using the **VARIABLE** command of COSYScript (see the Programmer's Manual [COSYScript]). It could have the form

VARIABLE <name> 201 8 ;

which declares a one dimensional array with eight entries. Each entry can hold a maximum of 201 16 byte blocks, which can store up to 200 rays. The additional one ray is to be reserved for the reference ray. The opposite operation can be performed by the command

LRAY <name> ;

where the rays stored in the array <name> are copied into the system, and the functionality is analogous to that of **RRAY** for **PRAY** and **WRAY**. Before calling **LRAY**, the command **CR** has to be called.

To the current set of momentary coordinates of the rays in the system, the command

ARAY <unit> ;

appends another set stored in a file produced by **PRAY** or **WRAY** via the specified unit. Note that currently the resulting set begins with the additional set and the pre-existed set is shifted toward the end. Unlike **RRAY** and **LRAY**, the command **CR** must not be called before calling **ARAY**. The command

ADDRAYS <file1> <file2> <file> ;

adds two sets of rays that are produced by **PRAY** or **WRAY** and stored in file1 and file2, and stores the resulting rays in a file by **PRAY** in the sequence of file1 and file2. Provide file names. The process of

ADDRAYS internally calls **CR**, so **ADDRAYS** has to be called independently from the current setup of rays to avoid any unwanted overwriting of rays.

To handle the spin coordinates associated to the rays, there are a similar set of commands. The command

PSPI <unit> ;

prints the momentary spin vector coordinates corresponding to the rays to the specified output unit. Each spin vector is output in one line, and the first vector in the output corresponds to the reference ray. The command

WSPI <unit> ;

prints the momentary spin vector coordinates corresponding to the rays to the specified output unit. **WSPI** prints all the coordinate values in the full digit length of double precision number expression. First, the s_x coordinate values of all the spin vectors are output, where the first component corresponds to the reference ray. Next, the s_y coordinate values are output, followed by s_z.

The spin vectors output by **PSPI** or **WSPI** can be read from the specified unit by the command

RSPI <unit> ;

Instead of writing and reading the spin vectors, it is also possible to copy the momentary spin vectors to a specified array. The command

SSPI <name> ;

copies the momentary spin vectors to an array <name>, which has to be specified by the user. The array can be specified using the **VARIABLE** command of COSYScript (see the Programmer's Manual [COSYScript]). It could have the form

VARIABLE <name> 201 3 ;

which declares a one dimensional array with three entries. Each entry can hold a maximum of 201 16 byte blocks, which can store up to 200 spin vectors. The additional one vector is to be reserved for the reference ray. The opposite operation can be performed by the command

LSPI <name> ;

where the spin vectors stored in the array <name> are copied into the system, and the functionality is analogous to that of **RSPI** for **PSPI** and **WSPI**.

To the current set of momentary spin vector coordinates corresponding to the rays in the system, the command

ASPI <unit> ;

appends another set stored in a file produced by **PSPI** or **WSPI** via the specified unit; see the description for **ARAY** above. The command

ADDSPIS <file1> <file2> <file> ;

adds two sets of spin vectors that are produced by **PSPI** or **WSPI** and stored in file1 and file2, and stores the resulting spin vectors in a file by **PSPI** in the sequence of file1 and file2. Provide file names; see the description for **ADDRAYS** above.

3.2.5 Plotting System and Trajectories

Besides computing transfer maps and rays, COSY INFINITY also allows to plot the system or any part of it and the rays going through it. The command

PTY < scale > ;

selects the type of system plot. If scale is zero, the reference trajectory will be plotted as a straight line; this is also the default if **PTY** is not called. If scale is nonzero, all rays including the reference trajectory are displayed in laboratory coordinates. To account for the fact that in such a view rays are rather close to the reference trajectory and hence may be hard to distinguish, the coordinates transverse to the optic axis will be magnified by the value of scale.

BP ;

defines the beginning of a section of the system that is to be plotted, and the command

EP ;

defines the end of the section. The command

PP <unit> <phi> <theta> ;

plots the system to the graphics output unit. Following the convention of printing graphics objects discussed in the Programmer's Manual [Graphics], positive units produce a low-resolution ASCII plot of 80 columns by 24 lines, which does not require any graphics packages. Negative units correspond to various graphics standards.

The picture of the trajectories and elements is fully three dimensional and can be viewed from different angles. Phi=0 and Theta=0 correspond to the standard x projection; Phi=0 and Theta=90 correspond to the y projection; and Phi=90 and Theta=0 correspond to viewing the rays along the beam.

There is an abbreviated way to produce both an x projection and a y projection simultaneously. The command

PG <Unit1> <Unit2> ;

produces both x and y pictures, including length (lower right), height (upper left) and depth (lower left) of the system with all selected rays and the envelope if selected. Unit1 and Unit2 denote the graphics output units (see the Programmer's Manual [Graphics]). The command

PGE <Unit1> <Unit2> ;

produces both x and y pictures, including length (lower right), height (upper left) and depth (lower left) of the system and the beam envelope. Unit1 and Unit2 denote the graphics output units (see the Programmer's Manual [Graphics]).

In a picture, it is sometimes advantageous to identify a particular location on the reference trajectory, for example to identify a focal plane or a plane of interest in a ring. This can be achieved with the command

PS <d> ;

which draws a Poincare section plane with width d at the momentary position of the reference trajectory.

There are several parameters which control the graphics output of a system. Such a graphics displays the central trajectory along with all rays and the envelope, the optical elements, two letters below each element indicating its type and three numbers indicating the height, width, and depth of the system. Before the system is computed, this default can be changed by

- LSYS = 0 ; (Suppresses the beamline elements)
- LCE = 0 ; (Suppresses the types of the elements)
- LAX = 0 ; (Suppresses the numbers describing the size of the system)

These options can become important when graphics output of huge machines is desired. These choices can then avoid memory overflow and incomprehensible picture.

3.3 Supported Elements

In this section we present a list of all elements available in COSY INFINITY. They range from standard multipoles and sectors over glass lenses and electromagnetic cylindrical lenses to a general element, which allows the computation of the map of any element from measured field data. The maps of all elements can be computed to arbitrary order and with arbitrarily many parameters. No approximation to the symplectic equations of motion [14] are being made. In particular, there is no expansions of square roots that is frequently performed [69] [17] [81].

Elements based on strong focusing devices such as multipoles and bending magnets can be computed with their fringe fields or without, which is the default. Section 3.3.7 beginning on page 24 describes various fringe field computation modes available.

The simplest particle optical element, the field- and material free drift, can be applied to the map with the command

DL <length> ;

The element

CB ;

changes the bending direction of bending magnets and deflectors. Initially, the bending direction is clockwise. The procedure **CB** changes it to counterclockwise, i.e. a left handed coordinate system, and each additional **CB** switches it to the other direction. Note that at the location of output with **PM** etc., it is important that the coordinate system is again right handed, so an even number of **CB** calls is necessary. It is also possible to change the bending direction of all the elements in an already computed map using the command **MT** (see index).

COSY INFINITY supports a large ensemble of other particle optical elements, and it is very simple to add more elements. The following subsections contain a list of momentarily available elements.

3.3.1 Multipoles

COSY INFINITY supports magnetic and electric multipoles in a variety of ways. There are the following magnetic multipoles:

MQ <length> <flux density at pole tip> <aperture> ;

MH <length> <flux density at pole tip> <aperture> ;

MO <length> <flux density at pole tip> <aperture> ;

MD <length> <flux density at pole tip> <aperture> ;

MZ <length> <flux density at pole tip> <aperture> ;

which let a magnetic quadrupole, sextupole, octupole, decapole or duodecapole act on the map. The aperture is the distance from reference trajectory to pole tip. For the sake of speed, direct formulas for the aberrations are used for orders up to two. There is also a superimposed multipole for multipole strengths up to order five:

M5 <length> <BQ >< BH >< BO >< BD >< BZ> <aperture> ;

And finally, there is a general superimposed magnetic multipole with arbitrary order multipoles:

MM <length> <MA> <NMA> <aperture> ;

Contrary to the previous procedure, the arguments now are the array MA and the number NMA of supplied multipole terms. Besides the magnetic multipole just introduced, which satisfies midplane symmetry, there is also a routine that allows the computation of skew multipoles. The routine

MMS <length> <MA> <MS> <NMA> <aperture> ;

lets a superposition of midplane symmetric and skew multipoles act on the map. The array MA contains the strengths of the midplane symmetric multipoles in the same units as above. The array MS contains the strengths of the skew multipoles; the units are such that a pure skew 2n pole corresponds to the midplane symmetric multipole with the same strength rotated by an angle of $\pi/2n$.

Similar procedures are available for electrostatic multipoles

EQ <length> <voltage at pole tip> <aperture> ;

EH <length> <voltage at pole tip> <aperture> ;

EO <length> <voltage at pole tip> <aperture> ;

ED <length> <voltage at pole tip> <aperture> ;

EZ <length> <voltage at pole tip> <aperture> ;

which let an electric quadrupole, sextupole, octupole, decapole or duodecapole act on the map. The strengths of the multipoles are described by their voltage in kV. There is an electric multipole

E5 <length >< EQ >< EH >< EO >< ED >< EZ> <aperture> ;

which lets a superimposed electric multipole with components EQ through EZ act on the map, and there is the procedure

EM <length> <EA> <NEA> <aperture> ;

which lets a general electrostatic multipole with arbitrary order multipoles act on the map. Similar to the magnetic case, there are also electric skew multipoles. The routine

EMS <length> <EA> <ES> <NEA> <aperture> ;

lets a superposition of midplane symmetric and skew multipoles act on the map. The array EA contains the strengths of the midplane symmetric multipoles in the same units as above. The array ES contains the strengths of the skew multipoles; like in the magnetic case, the units are such that a pure skew 2n pole corresponds to the midplane symmetric multipole with the same strength rotated by an angle of $\pi/2n$.

3.3.2 Bending Elements

COSY INFINITY supports both magnetic and electrostatic elements including so called combined function elements with superimposed multipoles. In the case of magnetic elements, edge focusing and higher order edge effects are also supported. By default, all bending elements bend the reference trajectory clockwise, which can be changed with the command **CB** (see index).

The following commands let an inhomogeneous combined function bending magnet and a combined function electrostatic deflector act on the map:

MS <radius> <angle> <aperture> $< n_1 >$ $< n_2 >$ $< n_3 >$ $< n_4 >$ $< n_5 >$;

E S <radius> <angle> <aperture> $< n_1 >$ $< n_2 >$ $< n_3 >$ $< n_4 >$ $< n_5 >$;

The radius is measured in meters, the angle in degrees, and the aperture is in meters and corresponds to half of the gap width. The indices n_i describe the midplane radial field dependence which is given by

$$F(x) = F_0 \cdot \left[1 - \sum_{i=1}^{5} n_i \cdot \left(\frac{x}{r_0} \right)^i \right],$$

where r_0 is the bending radius. Note that an electric cylindrical condenser has $n_1 = 1$, $n_2 = -1$, $n_3 = 1$, $n_4 = -1$, $n_5 = 1$, etc, and an electric spherical condenser has $n_1 = 2$, $n_2 = -3$, $n_3 = 4$, $n_4 = -5$, $n_5 = 6$, etc. Homogeneous dipole magnets have $n_i = 0$.

There are various specialized electrostatic deflectors [75]. The element

EC <radius> <angle> <aperture> <N> $< n >$;

invokes an electrostatic combined function deflector without the limitation to 5th order. Here N is an array containing the above indices n_i up to the n-th. The element

ECL <radius> <angle> <aperture> ;

invokes an electrostatic cylindrical deflector, and the element

ESP <radius> <angle> <aperture> ;

invokes an electrostatic spherical deflector.

Since there is frequently various confusion about electric elements and their properties, we describe a few elementary consistency tests and observations that are useful for checking purposes. First, an electric cylindrical condenser is invariant under translation along the y axis, and the y motion behaves like a drift. Furthermore, although the x motion depends on y and b, an offset in the y direction does not alter the x motion. Thus, a map produced by **ECL**, $\mathcal{M}_{ECL}$, and a y offset map $\mathcal{M}_{\triangle y}$, need to commute, i.e. we must have $\mathcal{M}_{ECL} \circ \mathcal{M}_{\triangle y} = \mathcal{M}_{\triangle y} \circ \mathcal{M}_{ECL}$. A particularly powerful way to check this within the transfer map concept of COSY is to make the offset Δy a DA parameter, and correspondingly use the command "**SA** 0 DA(5) ;".

A similar consistency test can be performed for an electric spherical condenser, for which any transfer map must be invariant under any rotation around an axis through the center. One of the meaningful tests based on this observation for which the geometry is easily worked out is a tilt of the plane of motion through the condenser along a central axis that is parallel to the reference orbit. A corresponding tilting map $\mathcal{M}_{\triangle}$ consists of moving to the center, rotating, and moving back; expressed in COSY's axis manipulating tools, it can be achieved for example by "**SA** -R 0 ; **RA** DA(5) ; **SA** R 0 ;". Now choosing a spherical condenser with a deflection of 180° entails that the reference orbit after the device is parallel with the the one before the device, but points in the opposite direction, so that the necessary back rotation requires the same (and not the opposite) angle. So, letting $\mathcal{M}_{ESP180}$ denote the spherical deflector, we must have that $\mathcal{M}_{\triangle} \circ \mathcal{M}_{ESP180} \circ \mathcal{M}_{\triangle}$ agrees with $\mathcal{M}_{ESP180}$. Yet another test is based on the observation that the motion is that of a Kepler problem, which entails that the motion should return to the exact original state after one full revolution, independent of initial conditions. Thus $\mathcal{M}_{ESP360}$, or a combination of k copies of $\mathcal{M}_{ESP(360/k)}$ for any k must lead to an identity map.

The element

DI <radius> <angle> <aperture> $< \epsilon_1 >$ <h$_1$ > $< \epsilon_2 >$ <h$_2$ > ;

lets a homogeneous dipole with entrance edge angle ϵ_1 and entrance curvature h$_1$ as well as exit edge angle ϵ_2 and exit curvature h$_2$ act on the map. All angles are in degrees, the curvatures in 1/m, the radius is in m, and the aperture is half of the gap width. Positive edge angles correspond to weaker x focusing, and positive curvatures to weaker nonlinear x focusing.

In the sharp cut off approximation, the horizontal motion in the homogeneous dipole is based on geometry. The vertical effects of edge angle and curvatures is approximated by a linear and quadratic kick, which is a common approximation of hard-edge fringe-field effects. As described in Section 3.3.7, it is also possible to treat the influence of extended fringe fields on horizontal and vertical motion in detail and full accuracy.

The element

MSS <radius r_0 > <angle ϕ_0 > <aperture> $< \epsilon_1 >$ <h$_1$ > $< \epsilon_2 >$ <h$_2$ > $< w >$;

allows the user to specify the two dimensional structure of the main field in polar coordinates, which is described by a two dimensional array $w_{(i,j)}$. The following factor is imposed to the main field specified by the first seven arguments, namely <radius> through <h$_2$ >, with the same meaning to those of **DI**.

$$\begin{aligned} F(r,\phi) &= \sum_{i=1}^{4}\sum_{j=1}^{4} w_{(i,j)} \cdot (r-r_0)^{i-1} \left(\phi - \frac{\phi_0}{2}\right)^{j-1} \\ &= w_{(1,1)} + w_{(2,1)} \cdot (r-r_0) + w_{(1,2)} \cdot \left(\phi - \frac{\phi_0}{2}\right) + \ldots + w_{(4,4)} \cdot (r-r_0)^3 \left(\phi - \frac{\phi_0}{2}\right)^3 . \end{aligned}$$

A special case of the homogeneous dipole described above is the magnetic rectangle or parallel-faced dipole, in which both edge angles equal one half of the deflection angle and the curvatures are zero. For convenience, there is a dedicated routine that lets a parallel faced magnet act on the map:

DP <radius> <angle> <aperture> ;

Finally, there is a very general combined function bending magnet with shaped entrance and exit edges

MC <radius> <angle> <aperture> <N> <S1> <S2> $< n >$;

Here N is an array containing the above n_i, and S1 and S2 are arrays containing the n coefficients s_1, ... s_n of two n-th order polynomials describing the shape of the entrance and exit edges as

$$S(x) = s_1 \cdot x + ... + s_n \cdot x^n.$$

Again positive zeroth order terms entail weaker x focusing. In the sharp cut off approximation, the edge effects of the combined function magnet are treated as follows. All horizontal edge effects of order up to two are treated geometrically like in the case of the dipole. The vertical motion as well as the contribution to the horizontal motion due to the non-circular edges are treated by kicks. The treatment of the element in the presence of extended fringe fields is described in Section 3.3.7.

Note that when comparing COSY bending elements without extended fringe fields to those of other codes, it is important to realize that some codes actually lump some fringe-field effects into the terms of the main fields. For example, the code TRANSPORT gives nonzero values for the element (x, yy), which is produced by a fringe-field effect, even if all TRANSPORT fringe-field options are turned off.

3.3.3 Wien Filters

Besides the purely magnetic and electric bending elements, there are routines for superimposed electric and magnetic deflectors, so-called Wien Filters or E cross B devices. The simplest Wien Filter consists of homogeneous electric and magnetic fields which are superimposed such that the reference trajectory is straight. This element is called by

WF <radius$_E$> <radius$_M$> <length> <aperture> ;

The radii describe the bending power of the electric and magnetic fields, respectively. The strengths are chosen such that each one of them alone would deflect the beam with the specified radius. For positive radii, the electric field bends in the direction of positive x, and the magnetic field bends in the direction of negative x. For equal radii, there is no net deflection. There is also a combined function Wien Filter:

WC <radius$_E$ > <radius$_M$ > <length> <aperture> <NE> <NM> $< n >$;

Here NE and NM describe the inhomogeneity of the electric and magnetic fields, respectively via

$$F(x) = F_0 \cdot \left[1 + \sum_{i=1}^{n} N_{(i)} \cdot x^i \right]$$

3.3.4 Wigglers and Undulators

COSY INFINITY allows the computation of the maps of wigglers. For the midplane field inside the wiggler, we use the following model:

$$B_m(x, z) = B_0 \cos \left(\frac{2\pi}{\lambda} z + k \cdot z^2 \right)$$

At the entrance and exit, the main field is tapered by an Enge function

$$B(x, z) = \frac{B_m(x, z)}{1 + \exp\left(a_1 + a_2 \cdot z/d + ... + a_{10} \cdot (z/d)^9\right)}.$$

The wiggler is represented by the following routine:

WI $< B_0 >$ $< \lambda >$ <L> $< d >$ <k> <I> <A> ;

where L is the length and d is the half gap. If I=0, the fringe field is modeled with some default values of the coefficients a_i. If I=1, the user is required to supply the values of a_1 to a_{10} for the entrance fringe field in the array A. The exit fringe field is assumed to have the same shape as the entrance fringe field.

3.3.5 Cavities

There is a model for a simple cavity in COSY INFINITY. It provides an energy gain that is position dependent but occurs over an infinitely thin region. The voltage of the cavity as a function of position and time is described by

$$V = P(x, y) \cdot \sin \left(2\pi \left(\nu \cdot t + \frac{\phi}{360} \right) \right),$$

so that ν is the frequency in Hertz, ϕ is the phase in degrees at which the reference particle enters the cavity. The peak voltage P is given in kV.

The cavity is represented by the following routine:

RF <V> $< n > < \nu > < \phi > < d >$;

where V is a two dimensional array containing the coefficients of a polynomial of order n describing the influence of the position as

$$P(x, y) = \sum_{i,j=0}^{n} V_{(i+1,j+1)} \cdot x^i y^j = V_{(1,1)} + V_{(2,1)} \cdot x + V_{(1,2)} \cdot y + \dots$$

and d is the aperture.

3.3.6 Cylindrical Electromagnetic Lenses

COSY INFINITY also allows the use of a variety of cylindrical lenses, in which focusing effects occur only due to fringe-field effects.

The simplest such element consists of only one ring of radius d that carries a current I. The on-axis field of such a ring is given by

$$B(s) = \frac{\mu_0 I}{2d} \cdot \frac{1}{\left(1 + (s/d)^2\right)^{3/2}} \tag{2}$$

which follows readily from the Biot-Savart law. This current ring is represented by the procedure

CMR $< I > < d >$;

A magnetic field of more practical significance is that of the so-called Glaser lens, which represents a good approximation of the fields generated by strong magnetic lenses with short magnetic pole pieces [49]. The lens is characterized by the on-axis field

$$B(s) = \frac{B_0}{1 + (s/d)^2}$$

where B_0 is the maximum field in Tesla and d is the half-width of the field. The Glaser lens is invoked by calling the procedure

CML $< B_0 > < d >$;

There are several magnetic solenoid elements available in COSY INFINITY.

CMSI $< I > < n > < d > < l >$;

invokes a thin solenoid with the theoretical on-axis field distribution

$$B(s) = \frac{\mu_0 I n}{2} \left(\frac{s}{\sqrt{s^2 + d^2}} - \frac{s - l}{\sqrt{(s - l)^2 + d^2}} \right),$$

obtained from (2), where I is the current, n the number of turns per meter, d the radius and l the length of the solenoid.

CMS $< B_0 > < d > < l >$;

invokes the solenoid with the following tanh-based on-axis field model:

$$B(s) = \frac{B_0}{2\tanh(l/2d)} \left(\tanh\left(\frac{s}{d}\right) - \tanh\left(\frac{s-l}{d}\right) \right),$$

where B_0 is the field strength at the center of the solenoid, d is its aperture and l its length. The field fall-off of this element is much more rapid than with **CMSI**. The element

CMST $< In > < R_1 > < R_2 > < l >$;

invokes a thick solenoid with the theoretical on-axis field distribution

$$B(s) = \frac{\mu_0 In}{2(R_2 - R_1)} \left[s \log\left(\frac{R_2 + \sqrt{R_2^2 + s^2}}{R_1 + \sqrt{R_1^2 + s^2}} \right) - (s-l) \log\left(\frac{R_2 + \sqrt{R_2^2 + (s-l)^2}}{R_1 + \sqrt{R_1^2 + (s-l)^2}} \right) \right],$$

where In is the product of the current I and the number of current turns per meter n, R_1 is the inner radius (aperture), R_2 is the outer radius, and l is the length.

The details on the features of these cylindrical magnetic elements can be found in [72].

There is a magnetic round lens with a Gaussian potential

$$V(s) = V_0 \cdot \exp\left(-\left(\frac{s}{d}\right)^2 \right),$$

which is invoked with the procedure

CMG $< V_0 > < d >$;

Besides the magnetic round lenses, there are various electrostatic round lenses. The element

CEL $< V_0 > < d > < L > < c >$;

lets an electrostatic lens consisting of three tubes act on the map. This lens is often called three-cube einzel lens. Figure 1 shows the geometry of the lens which consists of three coaxial tubes with identical radii d, of which the outer ones are on ground potential and the inner one is at potential V_0 in kV. The length of the middle tube is L, and the distance between the central tube and each of the outside tubes is c. Such an arrangement of three tubes can be approximated to produce an axis potential of the form

$$V(s) = -\frac{V_0}{2\omega c/d} \left[\ln\left(\frac{\cosh(\omega(s+L/2)/d)}{\cosh(\omega(s+L/2+c)/d)} \right) + \ln\left(\frac{\cosh(\omega(s-L/2)/d)}{\cosh(\omega(s-L/2-c)/d)} \right) \right],$$

where the value of the constant ω is 1.315. For details, refer to [60].

There is another electrostatic lens,

CEA $< V_0 > < d > < L > < c >$;

which lets a so-called three-aperture einzel lens act on the map. The geometry of the lens is shown in Figure 1. The outer apertures are on ground potential and the inner one is at potential V_0. The axis potential of the system can be approximated to be

$$V(s) = \frac{V_0}{\pi c} \cdot \left[(s+L/2+c)\tan^{-1}\left(\frac{s+L/2+c}{d}\right) + (s-L/2-c)\tan^{-1}\left(\frac{s-L/2-c}{d}\right) \right.$$
$$\left. -(s+L/2)\tan^{-1}\left(\frac{s+L/2}{d}\right) - (s-L/2)\tan^{-1}\left(\frac{s-L/2}{d}\right) \right].$$

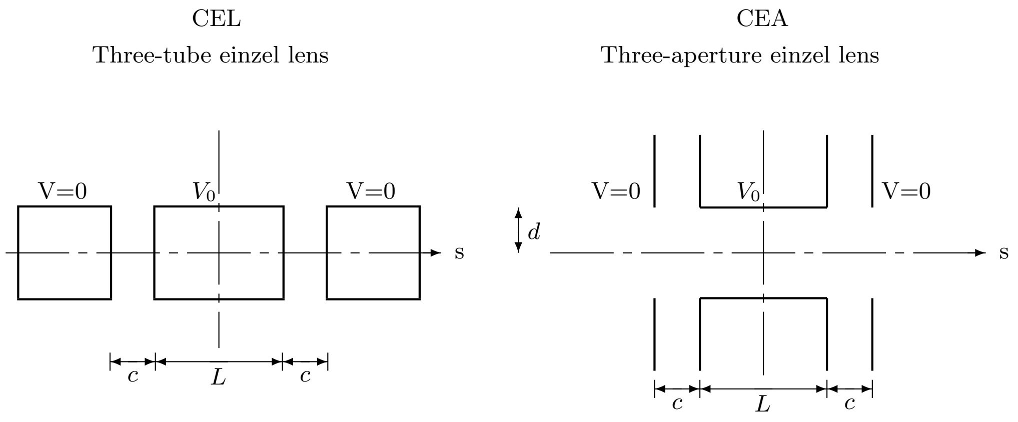


Figure 1: The constitution of electrostatic lenses of the procedures CEL and CEA

An often used approximation for electrostatic lenses is described by a potential distribution of the following form

$$V(s) = V_0 \cdot \exp\left(-\left(\frac{s}{d}\right)^2\right).$$

A lens with this field can be invoked by calling the routine

CEG $< V_0 > < d >$;

All round lenses are computed using COSY INFINITY's 8th order Runge Kutta DA integrator. The computational accuracy can be changed from its default of 10^{-10} using the procedure **ESET** (see index).

3.3.7 Fringe Fields

A detailed analysis of particle optical systems usually requires the consideration of the effects of the fringe fields of the elements [91] [93] [28] [15] [16] [40] [57] [92] [14]. While COSY INFINITY does not take fringe fields into account in its default configuration, there are commands that allow the computation of their effects with varying degrees of accuracy and computational expense.

There are two main ways of computing fringe fields of particle optical elements, namely utilizing one of the built-in modes provided by the various modes of the command **FR**, or one of the general element procedures described in detail in Section 3.3.8.

FR <mode> ;

provides various modes for fringe field map computations, which differ at the level of accuracy employed for computations. In the following, the modes will be described in *decreasing* order of accuracy.

In all cases, the mode set with **FR** stays effective until the next call to **FR**, and it is possible to change the computation mode within the computation. Whenever a new fringe field mode is desired, **FR** has to be called again with the new mode (which then remains in effect until the mode is changed with another call to **FR**). The default fringe field mode of COSY INFINITY is **FR 0**.

FR 3 & FR 2.5

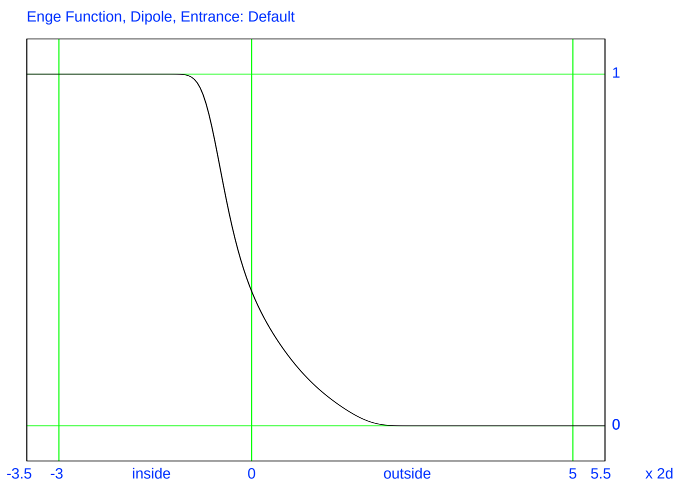


Figure 2: Enge function falloff of COSY's default dipole, drawn by the command FP.

This mode is the most accurate fringe-field mode. The fringe field falloff is based on the standard description of the s-dependence of multipole strengths by a six parameter Enge function. The Enge function is of the form

$$F(z) = \frac{1}{1 + \exp(a_1 + a_2 \cdot (z/D) + ... + a_6 \cdot (z/D)^5)},$$

where z is the distance perpendicular to the effective field boundary. In the case of multipoles, the distance coincides with the arc length along the reference trajectory. D is the full aperture (i.e., in case of multipoles $D = 2 \cdot d$) of the particle optical element, and a_1 through a_6 are the Enge coefficients. Using COSY INFINITY's DA based numerical integrator [68], if a supported element is called, the resulting map including fringe-field effects is computed using the full accuracy of the integrator and a default set of Enge coefficients. The values of the default set represent measurements of a family of unclamped multipoles used for PEP [34], and are listed in Table 1.

However, while in many cases the bulk of the effects can be described well with the default values of the coefficients, they depend on the details of the geometry of the element including shimming and saturation effects in magnetic elements. The coefficients should be adjustable such that the Enge function fits the specific measured or computed data. Fitting programs for this purpose have been written in COSYScript, or can be obtained as a companion of RAYTRACE [64]. Note that in the optimization process it is important that the Enge coefficients are chosen such that the effective field boundary coincides with the origin. It is also important that the fringe field coefficients lead to an Enge function which represents the fringe field well over an interval ranging from at least $3 \cdot D$ inside the element to at least $5 \cdot D$ outside the element, where $D = 2 \cdot d$ is as above.

Once an appropriate set of Enge coefficients has been determined, it is possible to use them by this mode. This is achieved with the command

FC <IMP> <IEE> <IEM> $< a_1 > < a_2 > < a_3 > < a_4 > < a_5 > < a_6 >$;

which sets the Enge coefficients a_1 through a_6 to the specified values. IMP is the multipole order (1 for dipoles, 2 for quadrupoles, etc). IEE identifies the data belonging to entrance (1) and exit (2) fringe fields. IEM denotes magnetic (1) or electric (2) elements. Using **FC** repeatedly, it is possible to set coefficients for the description of all occurring elements.

After setting the Enge coefficients with **FC**, the behavior of the resulting Enge function is diagnosed;

if the resulting fringe fields are inappropriate (e.g., if the fields do not not drop monotonically from 1 inside to 0 outside), an error message is issued. There is a convenient tool to draw Enge functions and the derivatives. The command

FP <IMP> <IEE> <IEM> <string> <order> <IU> ;

draws a picture of the Enge function (order = 0) or the derivative (order = the desired order of the derivative) to the graphics output unit IU (see the Programmer's Manual [Graphics]). A title can be added to the picture by using the string parameter. **FP** uses the Enge coefficients that are loaded ahead of time. Figure 2 is an example of such a picture, and is produced by the following commands:

```
FD ;                       {load default fringe field coefficients}
FP 1 1 1 'Default' 0 -12 ; {output to PDF file pic001.pdf}
```

To illustrate the concept of Enge coefficients, Tables 1 through 5 list some sets of Enge coefficients taken from various magnets.

COSY INFINITY uses a set of Enge coefficients for typical magnets based on measured data from PEP [34] by default, listed in Table 1. Unless specified explicitly using **FC**, regardless magnetic or electric, and entrance or exit, the following coefficients are used as mentioned on page 27 in the description of the procedure **FD**. The user does not have to do anything except for specifying the fringe field computation mode by the command **FR**, because these coefficients are loaded via **FD** as soon as the command **OV** is called.

	a_1	a_2	a_3	a_4	a_5	a_6
Dipole	0.478959	1.911289	-1.185953	1.630554	-1.082657	0.318111
Quadrupole	0.296471	4.533219	-2.270982	1.068627	-0.036391	0.022261
Sextupole and higher	0.176659	7.153079	-3.113116	3.444311	-1.976740	0.540068

Table 1: COSY INFINITY Enge coefficients by default. They are based on measured data from PEP at SLAC [34].

A benign Enge function can be achieved by utilizing only 2 coefficients, instead of 6. Furthermore, one may want the same effective field boundary in both cases. An example of the resulting Enge coefficients is given in Table 2.

	a_1	a_2	a_3	a_4	a_5	a_6
Dipole	-0.003183	1.911302	0.00	0.00	0.00	0.00
Quadrupole	0.00004	4.518219	0.00	0.00	0.00	0.00
Sextupole	-0.000117	7.135786	0.00	0.00	0.00	0.00

Table 2: Enge coefficients for a simple model.

The Large Hadron Collider's High Gradient Quadrupoles of the interaction regions have been designed by G. Sabbi. Based on the magnet end design described in [83], the Enge coefficients given in Table 3 have been obtained.

	a_1	a_2	a_3	a_4	a_5	a_6
Lead end	-0.939436	3.824163	3.882214	1.776737	0.296383	0.013670
Return end	-0.77462	3.75081	2.80154	0.833833	0.131406	0.0362236

Table 3: Enge coefficients of an LHC HGQ [83].

Table 4 lists the Enge coefficients modeling the NSCL's S800 spectrograph magnets which were obtained by fitting measured field data by D. Bazin [35].

		a_1	a_2	a_3	a_4	a_5	a_6
Quad. I	Entr.	0.150894	7.26981	-2.73798	2.0669	-0.256704	0.00
	Exit	0.15839	7.22058	-2.93658	2.62889	-0.333535	0.00
Quad. II	Entr.	0.0965371	6.63297	-2.718	10.9447	1.64033	0.00
	Exit	0.235452	6.60424	-3.42864	4.38392	-0.573524	0.00
Dipole I	Entr.	0.31809	2.11852	-1.0255	0.797148	0.00	0.00
	Exit	0.38027	2.01144	-0.900505	0.773862	0.00	0.00
Dipole II	Entr.	0.395308	2.03151	-0.910001	0.784602	0.00	0.00
	Exit	0.326167	2.08628	-1.01685	0.803716	0.00	0.00

Table 4: Enge coefficients of the S800 spectrograph at NSCL [35].

Finally, Table 5 lists a set of Enge coefficients obtained by F. Méot from a warm large aperture (diameter $\sim 30\ cm$) quadrupole that is part of a QD kaon spectrometer in operation at GSI.

a_1	a_2	a_3	a_4	a_5	a_6
0.1122	6.2671	-1.4982	3.5882	-2.1209	1.723

Table 5: Enge coefficients of a room temperature quadrupole at GSI.

Any set of fringe field parameters (electric/magnetic, entrance/exit) not explicitly set remains in its default configuration, and each **FC** command stays in effect until **FC** (or **FR**; see below) is called again. Therefore, if all dipoles, all quadrupoles, etc. in the system have the same fringe field falloff, it is sufficient to call **FC** only once for each type. In case there are different types, **FC** has to be called each time before the specific element. Sometimes it has proven helpful to lump several calls of **FC** into a procedure. One such procedure that is already part of COSY INFINITY is

FD ;

which sets all values to the default; this procedure is automatically called when COSY INFINITY's DA system is initialized, and it must be called again to reset the Enge coefficients to their default value, in case they have been changed. The accuracy of this computation mode is limited only by the accuracy of the numerical integrator. The computation is performed by COSY INFINITY's 8th order Runge Kutta DA integrator, and the computational accuracy can be set with the procedure

ESET $< \epsilon >$;

where ϵ is the maximum error in the weighted phase space norm discussed in connection with the procedure **WSET** (see index). The default for ϵ is 10^{-10} and can be adjusted downwards if needed.

COSY INFINITY's 8th order Runge Kutta DA integrator outputs a log file called RKLOG.DAT. In a rare occasion, a strange system error might happen. A Windows PC user contacted us in 2017 to report that the system issued an error "severe: write to READONLY file, unit 77, ... RKLOG.DAT", even though RKLOG.DAT is not a read-only file in COSY INFINITY. It turns out that the error was caused by an antivirus program locking the file as a read-only file while scanning. It can be avoided by forcing the antivirus program to exclude the user's work directory from scanning.

Note for the expert user: The trajectories of the particles are apparently affected by the fringe fields even before entering and after exiting the main part of the element. In case of bending elements, this means that the reference particle's trajectory can be different from what one expects from the geometric estimate using the concept of hard edge elements. The fringe-field mode **FR 3** forces the reference trajectory to travel perpendicularly through the transversal/radial axis in the middle of the element. Hence if the bending element is mirror symmetric along this middle line, the map of the element will reflect this symmetry.

If it is desired that this symmetry adjustment is not made, it is possible to utilize the fringe-field mode **FR 2.5**. In this case, the map computation starts well outside the entrance fringe field, travels through the element including the entrance and the exit fringe fields, and ends well outside the exit fringe field. This includes the necessary negative length adjustments using drifts for the outsides of both the entrance and the exit effective field boundaries. Normally, in this method the symmetry discussed above is broken, though for typical fringe fields, the offset is quite small. The method for the mode **FR 3**, described here, has been implemented in early 2013, and the earlier map computation under the mode **FR 3** corresponds to the mode that is now referred to as **FR 2.5**.

Since very detailed fringe field calculations are often computationally expensive, COSY INFINITY allows to compute their effects with lower degrees of accuracy.

<u>**FR 2 & FR 1.9**</u>

The fringe-field mode **FR 2** produces less accurate fringe fields than mode **FR 3** and **FR 2.5**, at a gain of computation time of typically more than one order of magnitude. Mode **FR 2** uses parameter dependent symplectic map representations of fringe field maps stored in files to approximate the fringe field via symplectic scaling [56] [55]. The default reference maps are stored in the file SYSCA.DAT. If needed, other reference files that give better representations of the user data can be created and stored in files by **WSM** (see index). How this is done can be seen in the procedure

CRSYSCA ;

This procedure produces the file SYSCA.DAT, which is shipped with the code. Such maps can be declared to be the new standard with the command

FC2 <IMP> <IEE> <file> ;

which declares file to be the actual reference file for the fringe field described by IMP and IEE; the meaning of IMP and IEE is discussed above for the command **FC**. The original default files can be reactivated by

FD2 ;

The fringe-field mode **FR 2** is especially helpful in the final design stages of a realistic system after approximate parameters of the elements have been obtained by neglecting fringe fields or with fringe-field mode **FR 1** (see below). The last step of the optimization can then be made using the default scaled fringe field maps. A high degree of accuracy almost equal to that of the fringe-field mode **FR 3** discussed above can be obtained by computing new fringe field reference maps with the command **WSM** based on the approximate values obtained by the previous fits.

Note for the expert user: it is possible to set the fringe-field mode to **FR 1.9**, which differs from mode **FR 2** only by the fact that each fringe field map is composed with the inverse of its linear part. This approach leaves the linear part of the system's map unaltered when turning fringe fields on and off, rendering the refitting of imaging properties or tunes unnecessary, but allows for an approximate but quite accurate study of only the nonlinear effects introduced by the fringe fields.

<u>**FR 1**</u>

This mode entails approximate fringe fields with an accuracy comparable to the fringe field integral method [92]. In fact, internally mode **FR 1** is exactly the same as mode **FR 3**, but it forces the numerical integration algorithm to go through the fringe field region in only two steps.

FR 0

All fringe fields are disregarded in this default mode. In this mode, a sharp cutoff approximation is used for all elements.

Stand Alone Fringe Fields

It is also possible to calculate stand alone fringe field maps. If the mode is set to **FR -1**, only the entrance fringe field maps of all listed elements are computed; if the fringe-field mode is set to **FR -2**, only exit fringe field maps are computed. In both cases, the computational accuracy is equivalent to that of mode **FR 3**.

Fringe fields produced with modes **FR -1** or **FR -2** can be thought of as *fringe field elements* with zero length. However, the apertures, strengths, etc. of the magnets have an influence on the results. (These are not thin lens models; the finite length fringe field maps are composed with negative drifts to give in the end a total length of zero.) To clarify this, notice that the following two code fragments are equivalent:

```
FR 3 ;
MQ L Q D ;
FR 0 ;
```

and

```
FR -1 ;
MQ L Q D ;
FR 0 ;
MQ L Q D ;
FR -2 ;
MQ L Q D ;
FR 0 ;
```

The fringe field maps computed using the modes **FR -1** or **FR -2** can be used in two ways: if the fringe fields do not change anymore, the data can be stored and re-used with the commands **SM** and **AM,** or **PM** and **RM** (see Section 3.2.3). In the case the maps of entrance or exit fringe fields are re-used in this way, it is important to turn all fringe fields off with the command **FR 0**, because otherwise the fringe fields would be taken into account twice. It is also important that in the case of bending elements with non-perpendicular entrance or exit (see Section 3.3.2), the fringe field maps computed using **FR -1** and **FR -2** do not contain the effects of any curved entrance and exit plane. Thus, in the case fringe field maps are re-used later with turned off fringe fields, it is important to leave all edge effects in the body of the element. Using modes **FR -1** and **FR -2**, it is also possible to determine new fringe field reference maps that can be used with symplectic scaling using the commands **WSM** and **RSM**.

General Fringe Field Maps

Besides the computation of fringe-field effects in the formalism of Enge type multipole functions, fringe-field effects can also be computed by any of the general particle optical elements **GE**, **MGE**, or **MF** discussed in Section 3.3.8. This allows the highly accurate treatment of strongly overlapping fringe fields or fringe fields that cannot be represented well by Enge functions.

We end this section on fringe-field effects with a few general comments. In the case of straight multipole elements, the total fringe field in the midplane is the sum of the individual multipole components which fall off with their respective Enge functions. The nonlinearities of the off-plane fields are computed in COSY INFINITY from this information in agreement with Maxwell's equations [4]. In the case of the dipole element **DI**, the Enge function modulates the falloff of the midplane dipole field perpendicular to the edge of the magnet. As long as the edges are long enough, this allows a very accurate description both for straight and circular edges, where circular edges may require Enge coefficients that differ slightly from those of straight edges with the same aperture. Again, the off-midplane fields are computed in agreement with Maxwell's equations.

In the case of all other bending elements, certain models have to be used to describe the details of the fringe field falloff in the Enge model. In the case of the inhomogeneous magnet **MS**, the inhomogeneity of the field which is determined by the distance to the center of deflection is modulated with an Enge falloff. In the case of the combined function magnet **MC**, the inhomogeneity of the field is modulated by a falloff function following as in the case of the dipole whose edge angles and curvatures are chosen to match the linear and quadratic parts of the curves described by S1 and S2. The remaining higher order edge effects are superimposed by nonlinear kicks before and after the element.

For general purpose bending magnets, it is rather difficult to formulate field models that describe all details to a high accuracy, and hence the accuracy of the computation of aberrations is limited by these unavoidable deficiencies. In case field measurements are available, the general element approach described above allows a detailed analysis of such measured data.

3.3.8 General Particle Optical Elements

In this section, we present procedures that allow the computation of an arbitrary order map for a completely general optical element whose fields are described by measurements.

One way to compute a map of a general optical element is to use the procedure **GE**, which uses measurements along the independent variable s. Its use ranges from special measured fringe fields over dedicated electrostatic lenses to the computation of maps for cyclotron orbits. It can also be used to custom build new elements that are frequently used (see Section 5.7 on page 54).

GE <n> <m> <S> <H> <V> <W> ;

lets an arbitrary particle optical element act on the map. The element is characterized by arrays specifying the values of multipole strengths at the n positions along the independent variable contained in the array S. The array H contains the corresponding curvatures at the positions in S. V and W contain the electric and magnetic scalar potentials in S.

The elements in V and W have to be DA variables containing the momentary derivatives in the x direction (variable 1) and s direction (variable 2), and m is the order of the s-derivatives. One way to compute these DA variables is to write two COSY functions that compute V and W as a function of x and s. Suppose these functions are called VFUN(X,S) and WFUN(X,S), then the requested DA variable can be stored in V and W with the commands

```
V(I) := VFUN(0+DA(1),S(I)+DA(2)) ;
W(I) := WFUN(0+DA(1),S(I)+DA(2)) ;
```

Another way to compute a map of a general optical element is to use the procedures **MGE** and **MF**. While **MGE** uses measured data of the field along the independent variable s, **MF** uses measured data of the field on the midplane in Cartesian coordinates [67] [66]. For deflecting elements, **MF** is more direct for the user. The command

MF <s> <BY> <N$_x$ > <N$_z$ > < $\triangle x$ > < $\triangle z$ > <S> < d > <S$_x$ > <S$_z$ > <S$_\phi$ > ;

lets an arbitrary particle optical element act on the map. The element is characterized by a two dimensional array BY$_{(i_x,i_z)}$ specifying the values of the field strength in the y direction B_y in the midplane along an equidistant grid. Figure 3 shows how the data grid is specified and the Cartesian coordinates corresponding

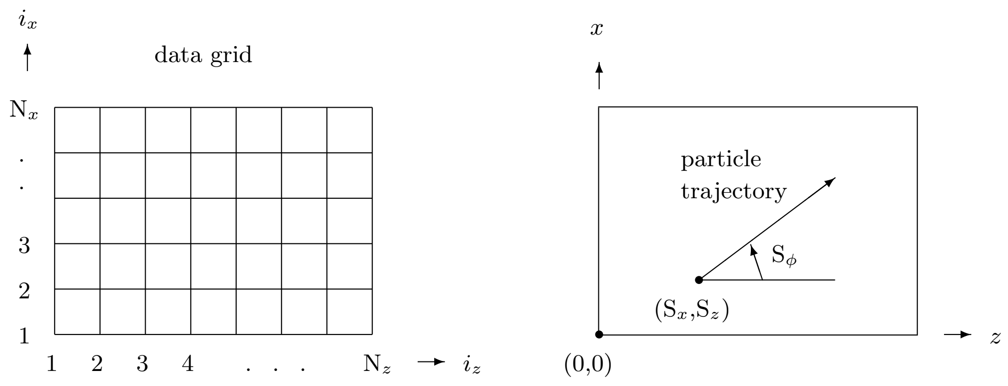


Figure 3: The specification of measured field data of the procedure MF

to the data grid. N$_x$ and N$_z$ are the numbers of measured data grid points in the x and z direction. $\triangle x$ and $\triangle z$ are the lengths of each grid in the x and z direction. As shown in Figure 3, S$_x$ and S$_z$ are the values of (x, z) coordinates of the starting point of the reference particle in the element, and S$_\phi$ is the angle (degree) at the starting point of the reference particle. s is the arclength along the reference particle, and d is the aperture.

The interpolation to evaluate the values of the field strength in the element is done by the method of Gaussian interpolation. S describes the width of the Gaussian curves. The value of the field strength B_y at the coordinates point (x, z) is interpolated by the following equation.

$$B_y(x,z) = \sum_{i_x}\sum_{i_z} \mathrm{BY}_{(i_x,i_z)} \cdot \frac{1}{\pi \mathrm{S}^2} \exp\left(-\left(\frac{x - x_{(i_x)}}{\triangle x \cdot \mathrm{S}}\right)^2 - \left(\frac{z - z_{(i_z)}}{\triangle z \cdot \mathrm{S}}\right)^2\right),$$

where $x_{(i_x)}$ and $z_{(i_z)}$ are the coordinates of the (i_x, i_z)-th grid point. A note has to be made to choose the suitable S. If S is too small, the mountains structure of Gaussians is observed. On the other hand, if S is too large, the original value supplied by the measured data is washed out. The suitable value of S depends on the original function shape of the measured data. For constant fields, the suitable S may be about 1.8. For quickly varying fields, it may be about 1.0. And larger values of S provide more accurate evaluation of the derivatives. In general, suitable values of S may be around $1.2 < \mathrm{S} < 1.6$.

Another note about the Gaussian interpolation is, since a Gaussian function falls down quickly, the time consuming summation over all the Gaussians is not necessary. The summation is well approximated by the 8S neighboring Gaussians of each side. For the value outside the area, the edge value is used. When such a situation happens, the total number of such points is reported as follows:

```
*** WARNING IN MF, OUT OF RANGE OF DATA AT       123 POINTS
```

In the case of quickly varying fields, a larger area of data has to be prepared.

Since the procedure **MF** consumes the memory size in the program, a small size is prepared for the download version of cosy.fox. If the measured data is bigger than 700×30 gridpoints, change the size for the array in cosy.fox in the following line.

```
VARIABLE MFD 6 700 30 ;
```

If this modification requires increasing the size of COSY INFINITY's internal memory, it is important to replace all occurrences of the parameter in question in all Fortran files. For example, if the error message demands the PARAMETER LVAR to be increased, change the value of LVAR in the PARAMETER statements in foxy.f, foxgraf.f, and dafox.f.

MGE is similar to **MF** except that data for multipole terms are specified. It can be used for multipoles whose field distribution cannot be described analytically by Enge functions etc. The command

MGE <NP> <A> <N$_s$ > $< \triangle s >$ <S> $< d >$;

lets a superimposed magnetic multipole based on measured data act on the map. N$_s$ is the number of measured data grid points along s, where each point is spaced equidistantly by $\triangle s$. S describes the width of the Gaussian as **MF**, and d is the aperture. NP is the maximum number of multipole components. The measured data is passed by a two dimensional array $\mathrm{A}_{(i_p,i_s)}$, where i_p denotes the multipole component as 1 for quadrupole, 2 for sextupole and so on, and $i_s = 1, ..., N_s$ denotes the i_s-th data point. A should be prepared to represent the field strength of the i_p-th component at the pole tip at the i_s-th position.

The same interpolation method is used as **MF**, so do the same cautions apply including the one on the memory size. The value of field strength B of the i_p-th component at the coordinates point s is interpolated as

$$B(i_p, s) = \sum_{i_s} \mathrm{A}_{(i_p,i_s)} \cdot \frac{1}{\sqrt{\pi}\mathrm{S}} \exp\left(-\left(\frac{s - s_{(i_s)}}{\triangle s \cdot \mathrm{S}}\right)^2\right),$$

where $s_{(i_s)} = \triangle s \cdot (i_s - 1)$ is the coordinate of the i_s-th grid point. Note that the total length of the element is $\triangle s \cdot (\mathrm{N}_s - 1)$.

The map of the general element is computed using COSY INFINITY's 8th order Runge Kutta DA integrator. The computational accuracy can be changed from its default of 10^{-10} using the procedure **ESET** (see index).

3.3.9 Absorber Wedges

COSY INFINITY allows the computation of the maps of absorber wedges. To activate the wedge computation mode, "**WAS 1 ;**" has to be placed before the **OV** command, under which wedge computations are to be executed. The command

WAS <mode> ;

controls the wedge computation mode. If mode is 0, wedge computations cannot be executed. Mode 1 activates the wedge computations.

The element

WA $< S_1 > < S_2 > < n >$ <length> <aperture> ;

lets a wedge absorber with shaped entrance and exit edges act on the map. The physical properties of the absorbing material have to be specified by calling **BBC** prior to **WA**, which sets the parameters for the Bethe-Bloch formula as described below. The entrance and exit edge surfaces are specified by S_1, S_2 and n, where S_1 and S_2 are two dimensional arrays containing the coefficients of polynomials of order n

describing the shape of the edge surfaces as

$$g_1(x,y) = \sum_{i,j=0}^{n} S_{1(i+1,j+1)} \cdot x^i y^j = S_{1(1,1)} + S_{1(2,1)} \cdot x + S_{1(1,2)} \cdot y + \ldots$$

$$g_2(x,y) = \sum_{i,j=0}^{n} S_{2(i+1,j+1)} \cdot x^i y^j = S_{2(1,1)} + S_{2(2,1)} \cdot x + S_{2(1,2)} \cdot y + \ldots$$

As described in Figure 4, the positive value of the polynomials g_1 and g_2 corresponds to the inward direction in the wedge. Note that the polynomials g_1 and g_2 must not have nonzero constant parts, namely it must be $S_{1(1,1)} = S_{2(1,1)} = 0$. For mirror symmetric edges, $S_{1(i,j)} = S_{2(i,j)}$ for all i and j, $1 \le i,j \le n+1$. The length is that of the reference orbit in the wedge absorber.

The model for the energy loss particles experience depending on the distance s traveled within the absorber is given by the Bethe-Bloch formula

$$\frac{dE}{ds} = -K\rho \frac{Z}{A} \frac{z^2}{\beta^2} \left(\log \left(\frac{2m_e c^2 \beta^2 \gamma^2 T_{\max}}{I^2} \right) - 2\beta^2 - \delta - 2\frac{C}{Z} \right)$$

with the parameters

- $K = 15.35375 \left(MeV \cdot cm^3 \right) (m \cdot g)^{-1}$.
- Z is the atomic number of the absorber material.
- A is the atomic mass of the absorber material.
- ρ is the density of the absorber material in $g \cdot cm^{-3}$.
- I is the ionization potential in MeV.
- δ is the density correction parameter.
- C is the shell correction parameter.

and the maximal kinetic energy transferred to a single electron in the absorber in a collision

$$T_{\max} = \frac{2m_e c^2 \beta^2 \gamma^2}{1 + 2\gamma m_e/m_0 + (m_e/m_0)^2}.$$

Note that in this case E is the total energy of the particle and that the unit of the dE/ds is $MeV \cdot m^{-1}$. The right hand side of the Bethe-Bloch formula depending on energy has been implemented as the function **BETHEBLOCH(E)**, where E is the total energy of the particle.

Prior to calling the function **BETHEBLOCH** one needs to call the procedure

BBC $< Z > < A > < \rho > < I > < \delta > < C >$;

to set the proper parameters for the Bethe-Bloch formula.

If the user wishes to use a different energy loss function rather than the Bethe-Bloch formula, the user can replace the function **BETHEBLOCH** with an appropriate function by coding it in cosy.fox.

There is a procedure

EL $< E_i > < ct_i > < l > < E_f > < ct_f >$;

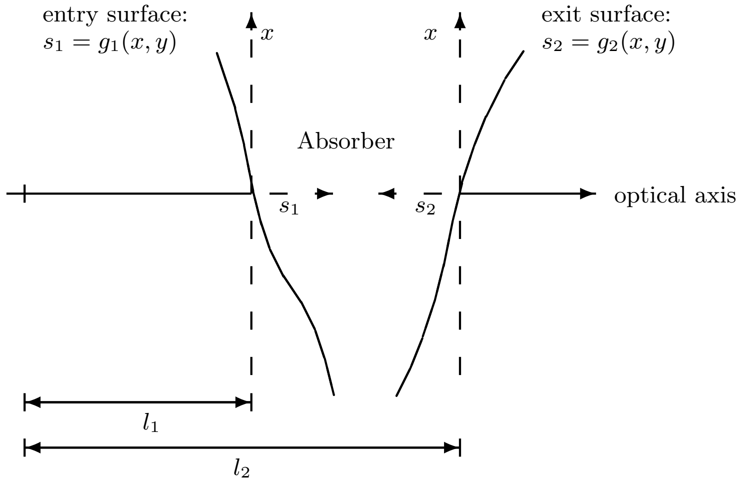


Figure 4: The geometric setup for the procedures WA and WL.

which calculates the kinetic energy and time of flight (times the speed of light c) as an expansion in the arclength s for the currently set beam after it travels through an absorber of thickness l. The input parameters are the initial energy and time of flight (times c) E_i and ct_i and the thickness l, and the output variables are E_f and ct_f for the final energy and time of flight (times c) after the wedge respectively. Since internally the function **BETHEBLOCH** is invoked, the user has to call **BBC** to set the absorber characteristics prior to calling **EL**.

The procedure

WL $< S_1 > < S_2 > < n > < l_1 > < l_2 > < l_f >$;

allows to compute the total distance that a particle travels inside the wedge depending on its initial conditions. As **WA**, S_1 and S_2 are two dimensional arrays containing the coefficients of polynomials of order n, g_1 and g_2, describing the shape of the entrance and exit edge surfaces. l_1 and l_2 are the positions of entrance and exit surface respectively, and l_1 is typically zero. Then, l_f is an output for the total length traveled inside the wedge for the particle depending on the initial conditions as a DA-vector taking account of the shape of edges.

3.3.10 Glass Lenses and Mirrors

COSY INFINITY also allows the computation of higher order effects of general glass optical systems. At the present time, it contains elements for spherical lenses and mirrors, parabolic lenses and mirrors, and general surface lenses and mirrors, where the surface is described by a polynomial. There is also a prism. All these elements can be combined to systems like particle optical elements, including misalignments. The dispersion of the glass can be treated very elegantly by making the index of refraction a parameter using the function **PARA**.

The command

GLS <R1> <R2> <N> <L> $< d >$;

lets a spherical glass lens act on the map. R1 and R2 are the radii of the spheres; positive radii correspond to the center of the sphere to be to the right. N is the index of refraction, L is the thickness, and d the aperture radius. The command

GL <P1> $< n_1 >$ <P2> $< n_2 >$ <N> <L> $< d >$;

lets a glass lens whose surface is specified by two polynomials of orders n_1 and n_2 act on the map. P1 and P2 are two dimensional arrays containing the coefficients of the polynomials in x and y that describe the s position of the entrance and exit surface as a function of x and y in the following way:

$$P(x,y) = \sum_{i,j=0}^{n} P_{(i+1,j+1)} \cdot x^i y^j = P_{(1,1)} + P_{(2,1)} \cdot x + P_{(1,2)} \cdot y + \dots$$

N is the index of refraction, L the thickness of the lens and d its aperture. The command

GP <PHI1> <PHI2> <N> <L> $< d >$;

lets a glass prism act on the map. PHI1 and PHI2 are the entrance and exit angles measured with respect to the momentary reference trajectory, N is the index of refraction, L the thickness along the reference trajectory, and d is the aperture radius.

Besides the refractive glass optical elements, there are mirrors. In the following mirror elements, d is the aperture radius. The command

GMS <R> $< d >$;

lets a spherical mirror with radius R act on the map. The command

GMP <R> $< d >$;

lets a parabolic mirror with central radius of curvature R act on the map. The command

GMF <PHI> $< d >$;

lets a flat mirror with the tilt angle PHI act on the map. The command

GM <P> $< n >$ $< d >$;

lets a general glass mirror act on the map. P is a two dimensional array containing the coefficients of the polynomial in x and y that describes the surface in the same way as with **GL**, and is the dimension.

3.3.11 Misalignments

The differential algebraic concept allows a particularly simple and systematic treatment of misalignment errors in optical systems. Such an error is represented by a coordinate change similar to the one discussed in Section 4.1. COSY INFINITY offers three different misalignment commands. The first command

SA <DX> <DY> ;

offsets the optic axis by DX in x direction and DY in y direction. DX and DY are counted positive if the optic axis is shifted in direction of positive x and y, respectively. The command

TA <AX> <AY> ;

represents a tilt of the optic axis by an angle in degrees of AX in x direction and AY in y direction. AX and AY are counted positive if the direction of tilt is in the direction of positive x and y, respectively. The command

RA <ANGLE > ;

represents a rotation of the optic axis around ANGLE measured in degrees. ANGLE is counted positive if the rotation is counterclockwise if viewed in the direction of the beam. The routine **RA** can be used to rotate a given particle optical element by placing it between counteracting rotations. This can for example be used for the study of skew multipoles. However, note that it is not possible to rotate different multipole components by different angles. This can be achieved with the routines **MMS** and **EMS** discussed in Section 3.3.1.

In order to simulate a single particle optical element that is offset in positive x direction, it is necessary to have the element preceded by an axis shift with negative value and followed by an axis shift with positive value. Similarly simple geometric considerations tell how to treat single tilted and rotated elements.

The misalignment routines can also be used to study beams that are injected off the optical axis of the system. In this case, just one of each misalignment commands is necessary at the beginning of the system.

We note that the misalignment routines, like most other COSY routines, can be called both with real number and differential algebraic arguments, in particular using the **PARA** argument (see Section 5.2). The first case allows the simulation of a fixed given misalignment, whereas the second case allows to compute the map depending on the misalignment.

In the first case, the values of the computed transfer map are only approximate if **SA** and **TA** are used. The accuracy increases with decreasing misalignments and increasing calculation orders. For the study of misalignments of elements, the actual accuracy is usually rather high since the values of the misalignments are usually very small. In the case of a deliberate offset of the beam, for example for the study of injection and extraction processes, it may be necessary to increase the computation order to obtain accurate results. In the second case, the results are always accurate. The command **RA** always produces accurate results in both cases.

3.4 Lattice Converters

There are tools to convert existing lattices described in the other formats into COSYScript language. The following subsections explain the currently available lattice converters. The converters are web-based, and the links to the web pages of the converters can be found at https://cosyinfinity.org .

We appreciate receiving other converters to COSYScript written by users to be available to the other users.

3.4.1 MAD Input

Many existing accelerator lattices are described in the MAD standard [58] [59]. To allow the use of such MAD lattices in COSY INFINITY, there is a conversion utility that transforms MAD lattices to the COSY INFINITY lattices. This utility was originally written by Roger Servranckx using the original MAD compiler source code which was written by Christopher Iselin. The current program has been adjusted to MAD version 8.22 by Weishi Wan and Kyoko Makino. The MAD to COSY converter is provided on the web at https://cosyinfinity.org .

The converter is based on MAD version 5, but will also accept most higher level input. The important beamline elements are translated into the respective ones in COSY INFINITY; these include drifts, multipoles, superimposed multipoles, and bends. Some elements supported by MAD are translated to drifts and may have to be adjusted manually.

To generate a COSY deck from a MAD deck, the end of the MAD deck should have the form

```
USE, <name of beamline>
     COSY
     STOP
```

where according to the MAD syntax, the USE command specifies the beamline to be translated, and the command COSY actually generates the COSYScript source.

3.4.2 SXF Input

The SXF format (Standard eXchange Format) is meant to be a general lattice description language and is intended to facilitate the cooperation between different groups and the comparison of results obtained with different codes. The language specifications that are in most parts very similar to MAD were developed by H. Grote, J. Holt, N. Malitsky, F. Pilat, R. Talman, G. Tahern and W. Wan.

3.4.3 GICOSY Input

GICOSY used at GSI and University of Gießen is based on COSY 5.0, with additions done later at Gießen in the years 1986 - 1998. The code was written by Martin Berz, Bernd Hartmann, Klaus Lindemann, Achim Magel, Helmut Weick, and in former times was also referred to as simply GICO. (From the GICOSY web site https://web-docs.gsi.de/~weick/gicosy/.)

To allow the use of GICOSY lattices in COSY INFINITY, there is a conversion utility that transforms GICOSY lattices to the COSY lattices. This utility was written by Shashikant Manikonda.

The GICOSY to COSY converter is provided on the web at https://cosyinfinity.org.

3.4.4 OptiM Input

OptiM is a code for linear and nonlinear optics calculations written by Valeri Lebedev at Fermilab. (From the OptiM web site https://pbar.fnal.gov/organizationalchart/lebedev/OptiM/optim.htm.)

To allow the use of OptiM lattices in COSY INFINITY, there is a conversion utility that transforms OptiM lattices to the COSY lattices. This utility was written by Pavel Snopok [87].

4 Analyzing Systems with COSY INFINITY

4.1 Image Aberrations

Very often not the matrix elements of the transfer map are of primary significance, but rather the maximum size of the resulting aberration for the phase space defined with **SB** and the parameters defined with **SP**. COSY INFINITY provides two tools to obtain the aberrations directly. The command

PA <unit> ;

prints all aberrations to the output unit in a similar way as **PM**. If not all aberrations are of interest, the COSY function

MA(<phase space variable>,<element identifier>)

returns the momentary value of the aberration. The phase space variable is a number from 1 to 6 corresponding to x, a, y, b, t, d, and the element identifier is an integer whose digits denote the above variables. For example, **MA**(1,122) returns the momentary value of the aberration due to the element (x, xaa).

For comparison and other reasons, it is often helpful to express the map in other coordinates than those used by COSY INFINITY (see Section 3.2.1, for example the ones used in TRANSPORT [33] and GIOS. The routine

PT <unit> ;

prints the map in Transport and GIOS coordinates to the output unit.

We want to point out that in the differential algebraic concept, it is particularly simple to perform such nonlinear coordinate changes to arbitrary orders. In order to print maps in yet different coordinates, the user can make a procedure that begins with a unity map, applies the transformation to COSY coordinates, applies the COSY map, and then applies the transformation back to the original coordinates.

4.2 Analysis of Spectrographs

To first order, the resolution $\Delta\delta$ of an imaging spectrograph is given by the following simple formula:

$$\Delta\delta = \frac{(x,x)\cdot 2X_0}{(x,d)}$$

where X_0 is the half width of the slit or aperture at the entrance of the device. Here δ can be any one of the quantities δ_k, δ_m and δ_z, and it is assumed that to first order, the final position does not depend on the other quantities, or all particles have the same initial values for the other quantities.

In all but the simplest spectrographs, however, it is important to consider higher order effects as well as the finite resolution of the detectors. Usually these effects decrease the resolution, more so for larger initial phase spaces and low detector resolutions. The resolution of the spectrograph under these limitations can be computed with the following command

AR <MAP> <X> <A> <Y> <B> <D> <PR> <N> <R> ;

where MAP is the map of the spectrograph to be studied, X, A, Y, B and D are the half widths of the beam at the entrance of the spectrograph, PR is the resolution of the detector, and R is the resulting resolution of the spectrograph. To compute the resolution, a total of N particles are distributed randomly and uniformly within a square initial phase space and then sent through the map. Then the measurement error is introduced by adding a uniformly distributed random number between -PR and PR to the x coordinate. The width of the resulting blob of measurements is computed, where it is assumed that the blob is again filled uniformly.

In many cases the resolution of spectrographs can be increased substantially with the technique of trajectory reconstruction [19]. For this purpose, positions of each particle are actually measured in two detector planes, which is equivalent to knowing the particle's positions and directions.

Assuming that the particle went through the origin, the energy of the particle is uniquely determined by some complicated nonlinear implicit equations. Using DA methods, it is possible to solve these equations analytically and relate the energy of the particle to the four measured quantities. Besides the energy, it is also possible to compute the initial angle in the dispersive plane, the initial position in the non-dispersive plane, and the angle in the non-dispersive plane. The accuracy of these equations is limited only by the measurement accuracy and by the entering spot size in the dispersive plane. This is performed by the command

RR <MAP> <X> <A> <Y> <B> <D> <PR> <AR> <N> <O> <MR> <R> ;

where the parameters are as before, except that AR is the resolution in the measurement of the angle, and O is the order to which the trajectory reconstruction is to be performed. On return, MR is the nonlinear four by four map relating initial a, y, b and d to the measured final x, a, y, b. Using these relationships as well as the measurement errors and the finite dispersive spot size, the resolution array R containing the resolutions of the initial a, y, b and d is computed by testing N randomly selected rays and subjecting them to statistical measurement errors similarly as with the computation of the uncorrected resolution.

4.3 Analysis of Rings

Instead of by their transfer matrices, the linear motion in particle optical systems is often described by the tune and twiss parameters. These quantities being particularly important for repetitive systems, they allow a direct answer to questions of linear stability, beam envelopes, etc. In many practical problems, their dependence on parameters is very important [11] [14] [17]. For example, the dependence of the tune on energy, the chromaticity, is a very crucial quantity for the design of systems. Using the maps with knobs, they can be computed totally automatically without any extra effort. The command

TP <MU> ;

computes the tunes which are stored in the one dimensional array with three entries MU which is defined by the user. In most cases, an allocation length of 100 should be sufficient, and so the declaration of MU could read

```
VARIABLE MU 100 3 ;
```

If the system is run with parameters, MU will contain DA vectors describing how the respective tunes depend on the parameters. Note that COSY INFINITY can also compute amplitude dependent tune shifts in the framework of normal form theory. This is described in detail in this section.

For the computation of amplitude tune shifts and other characteristics of the repetitive motion, COSY INFINITY contains an implementation of the DA normal form algorithm described in [10] [7] [14] [24]. This replaces the COSY implementation of the less efficient and less general mixed DA-Lie normal form [47]. Normal Form algorithms provide nonlinear transformations to new coordinates in which the motion is simpler. They allow the determination of pseudo invariants of the system [14] [13] [20] [30] [66] [57], and they are the only tool so far to compute amplitude tune shifts. As pointed out in [11], chromaticities and parameter dependent tune shifts alone can be computed more directly using the command **TP** described above. The command

NF <EPS> <MA> ;

computes the normal form transformation map MA of the momentary transfer map. This variable has to be allocated by the user, and in most cases

```
VARIABLE MA 1000 8 ;
```

should be sufficient. Since the normal form algorithm sometimes has problems with the possible occurrence of small denominators, it is not always possible to perform a transformation to coordinates in which the motion is given by circles. The variable EPS sets the minimum size of a resonance denominator that is not removed. The command

TS <MU> ;

employs the normal form algorithm to compute all the tune shifts of the system, both the ones depending on amplitude and the ones depending on parameters like chromaticities, which alone can be computed more efficiently as shown above. MU is a one dimensional array with three entries which is defined by the user in a similar way to TP. On return, MU will contain the tune shifts with amplitudes and parameters as DA vectors. If the system is run with parameters, MU will contain DA vectors describing how the respective tunes depend on the amplitudes (first, third and possibly fifth exponents for x, y and t) and parameters (beginning in columns five or 7).

Note that in some cases when the system is on or very near a resonance or is even unstable, the normal form algorithm may fail due to occurrence of a small denominator. In this case, the respective tunes will be returned as zero. This also happens sometimes if the map is supposed to be symplectic yet is slightly off because of computational inaccuracies. In this case, the use of the procedure **SY** (see index) is recommended.

The command

TSC $< w_x > < w_y > < N_x > < N_y > < \delta >$ <file> ;

scans the tunes in the rectangular area $-w_x \le x \le w_x$, $-w_y \le y \le w_y$ at $N_x \times N_y$ equi-distant sampling points and the resulting tune footprint data is output to a file with the specified file name. The first and the second columns are the x, y coordinate values of each sampling point, and the third and the fourth columns are the x, y tune values. The energy deviation can be specified via δ. The command

RFILT $< R_{\max} >$ <file> <fileR_{limit} > ;

filters the resulting tune footprint data stored in the file as a result of **TSC** such that only those sampling points within the radius $R_{\max}$, i.e. $r = \sqrt{x^2 + y^2} < R_{\max}$, are kept, and outputs to a different file with the specified name fileR_{limit}. The command **RFILT** can be used for any other data file as long as the first and the second columns are the x, y coordinate values.

The normal form method can also be used to compute resonance strengths, which tell how sensitive a system is to certain resonances. Often the behavior of repetitive systems can be substantially improved by reducing the resonance strengths. These are computed with the procedure

RS <RES> ;

where upon return RES is a complex DA vector that contains the resonance strengths; for details, refer to [14]. The $2 \cdot N$ exponents n_i^+, n_i^- in each component describe the resonance of the tunes ν as

$$(\vec{n}_i^+ - \vec{n}_i^-) \cdot \vec{\nu}.$$

The linear and nonlinear momentum compaction $(dl/dp) \cdot p/l$ can be computed with the routine

MCM <M> <L> <C> ;

Alternatively, it also possible to compute the Energy compaction (dr_5/dr_6) with the routine

ECM <M> <L> <C> ;

Finally it is also possible to analyze the spin motion with normal form methods. The command

TSP <MU> $< \bar{n} >$ <KEY> ;

computes the parameter dependence of spin tune and the invariant spin axis $\bar{n}$. The command

TSS <MU> < $\bar{n}$ > <KEY> ;

computes the parameter and amplitude dependence of spin tune as well as the invariant spin axis $\bar{n}$. The spin tunes are stored in the one dimensional array with three entries MU which is defined by the user, in a similar way as the array used by TP and TS. If KEY is 0, the original orbital variables are used. If KEY is not 0, the orbital variables are transformed to the parameter dependent fixed point.

4.4 Repetitive Tracking

COSY INFINITY allows efficient repetitive tracking of particles through maps. The command

TR <N> <NP> <ID1> <ID2> <D1> <D2> <TY> <NF> <IU>;

tracks the momentary particles selected with **SR** or **ER** through the momentary map for the required number of iterations N. After each |NP| iterations the position of the phase space projection ID1-ID2 is drawn to the graphics output unit IU. The phase space numbers 1 through 6 correspond to x, a, y, b, d, t, and the numbers -1, -2, -3 correspond to the x, y and z components of the spin. If any of these components get larger than D1, D2, they will not be drawn.

When tracking many particles, it is useful not to include the initial particles in the plot. This can be achieved by giving a negative integer to NP.

The particles selected for repetitive tracking by the command **TR** are different from those used under the command **BP**, thus they should be clearly separated using the command **CR** if the user's input program uses both **TR** and **BP**.

TY specifies the mode of symplectification, and NF turns on and off the display mode in normal form variables.

TY=0

Symplectic tracking using the EXPO generating function is performed [41] [42] [40] [84].

$1 \leq |\mathbf{TY}| \leq 4$

Symplectic tracking using the generating function of type |TY| (see [8] [14]) is performed. For TY> 0, a fixed-point iteration is used to determine the symplectified map. For TY< 0, a Newton iteration is used to determine the symplectification. While the Newton method is more robust, the fixed point iteration tends to be faster if it works.

TY=-12 or **TY=-13**

Symplectic tracking is performed by symplectifying the linear map $\mathcal{M}_L$ and representing the map $\mathcal{N} = \mathcal{M} \circ \mathcal{M}_L^{-1}$ by the generating function of type |TY+10|. Because the linear part of $\mathcal{N}$ is the unity map, only the generating functions of type 2 (for TY= -12) and 3 (for TY=-13) can be used for that purpose.

TY=-21

The tracking is performed without symplectification.

NF=0

The points will be displayed in conventional variables.

NF=1

The points will be displayed in normal form variables.

The algorithm used for tracking is highly optimized for speed. Using the vector data type for particle coordinates, it works most efficiently if many particles are tracked simultaneously. On scalar machines, optimum efficiency is obtained when more than about 20 particles are tracked simultaneously. On true high-performance vector machines, the algorithm fully auto-vectorizes, and for best efficiency, the number of particles should be a multiple of the length of the hardware vector. On most modern CPUs, also limited amounts of vectorization are performed automatically, and the algorithms used for the vector data type are optimally mapped in these environments as well. In these cases, logistics overhead necessary for the bookkeeping is almost completely negligible, and the computation time is almost entirely spent on arithmetic. It is also worth mentioning that using an optimal tree transversal algorithm, zero terms occurring in a map do not contribute to computation time.

The command

TRT <string>;

prints the title supplied by the string in a tracking picture produced by **TR**. The command **TRT** should be called just before a **TR** call, and the title is valid only for that **TR** call. If **TRT** is not called just before **TR**, no title is printed.

Sometimes it is desirable to resume tracking particles. The next command

TRR <mode>;

sets the resuming mode on (1), and off (0). By default, the resuming mode is off. When the resuming mode is desired, **TRR** has to be called with 1 ahead of calling **TR**. Once the resuming mode is turned on, it remains active until it is turned off.

One can output a plot at every N-th iteration starting from the initial particles as follows. If the command **TRR** is called ahead of time, "`TRR 0 ;`" should be called to turn off the resuming mode in the beginning.

```
TR  0  1 ... ;  {plot of the initial particles}
TRR 1 ;         {turn on the resume tracking mode}
TR  N -N ... ;  {plot at the   N-th iteration}
TR  N -N ... ;  {plot at the 2*N-th iteration}
```

There are two ways to apply aperture cuts through repetitive tracking using **TR**. The command

TRAP $< x >$ $< y >$ <C> ;

cuts the rays in every iteration by the sizes x and y in the transverse plane. The shape of the aperture (collimator) is elliptic with |C|=1, and rectangular with |C|=2. C=0 turns this aperture cut off. By default, this aperture cut is turned off. If a negative number is given to C, no aperture cut is applied to the initial particles. To apply this aperture cut, the command **TRAP** has to be called ahead of calling **TR**, and the feature is on until it is turned off.

To insert a cutting aperture (collimator) at a specific location in the system, call the command

AP $< x >$ $< y >$ <C> ;

at the location while defining the system to compute a transfer map to be used for **TR**. **AP** can be placed at multiple locations through the system. The command cuts the rays at the specified location by the sizes x and y in the transverse plane. The shape of the aperture is elliptic with C=1, and rectangular with C=2. If any other value is given to C, this aperture is ignored. A **UM** call resets the **AP** apertures, meaning that all the necessary **AP** apertures have to be called after **UM** and before **TR**.

If **TRAP** and **AP** (with C=1, 2) co-exist, **TRAP** is ignored. Using a pre-coded system setup that includes **AP**s, sometimes it may be necessary to turn on and off the **AP** aperture cutting. Turning off **AP**s can be handled in the following two ways, and both ways activate the co-existed **TRAP**. One way is to give C by a variable, and switch the value of C from on (C=1, 2) to off (C=0) before calling **AP**s. Another way is to reset the internal counter of **AP**s before calling **TR** by writing "`NCT := 0 ;`", however this method has to be conducted carefully as it directly alters the value of an internal global variable.

The command

TRAPN <N> ;

reports the number of rays survived through the aperture cuts of **TR** via the variable in the command argument. The length 1 is enough for the command argument variable.

VARIABLE N 1 ;

The history of the number of survived particles can be reported by the command

TRAPHIST <file> <L> ;

to a file with the specified file name. Typically, use an internal global variable LTRI for <L>, i.e.

```
TRAPHIST traphist.txt LTRI ;
```

When tracking many particles with cutting apertures, the number of survived particles may be much less than the number of initial particles. The command

TRAYPAC ;

packs rays to keep only the survived particles.

Similar to the command **PRAY** (see page 13), some users may want to print coordinates of the rays during tracking using **TR**. The next command

TRPRAY <unit> ;

prints the coordinates of the rays to the specified output unit after each NP iterations as well as the initial values. Each ray is output in one line with the iteration number (the first column) and the ray number (the second column), where the ray number 0 corresponds to the reference ray. The command **TRPRAY** should be called just before a **TR** call, and the ray printing feature is valid only for that **TR** call.

The command with the same functionality for **PSPI** (see page 15) is

TRPSPI <unit> ;

printing the spin vector coordinates corresponding to the rays to the specified output unit after each NP iterations as well as the initial values. Each spin vector is output in one line with the iteration number (the first column) and the ray number (the second column), where the ray number 0 corresponds to the reference ray. The command **TRPSPI** should be called just before a **TR** call, and the spin vector printing feature is valid only for that **TR** call. When **TRPSPI** is called at the same time with **TRPRAY**, different output unit numbers have to be supplied.

4.5 Symplectic Representations

In this section, we will present two different representations for symplectic maps, each one of which has certain advantages. Particle optical systems described by Hamiltonian motion satisfy the symplectic condition (see for example [50] [14])

$$M \cdot J \cdot M^t = J$$

where M is the Jacobian Matrix of partial derivatives of $\mathcal{M}$, and J has the form

$$J = \begin{pmatrix} 0 & 0 & 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 & 0 & +1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{pmatrix}$$

As long as there is no damping, all particle optical systems are Hamiltonian, and so the maps are symplectic up to possibly computation errors if they are generated numerically. There is a COSY function that determines the symplectic error of a map:

SE(<M>)

Here M is an array of DA quantities describing the map. Note that the momentary value of the transfer map is stored in the global COSY variable MAP. The value of the function is the weighted maximum norm of the matrix $(M \cdot J \cdot M^t - J)$. The weighting is done such that the maximum error on a cubic phase space with half edge W is computed. The default value for W is 0.1, which may be too large for many cases. The value of W can be set with the procedure

WSET <W> ;

While the orbital part of maps usually satisfies the symplectic symmetry, the spin matrix must satisfy orthogonality. Similar to the function **SE**,

OE(<SM>)

determines the orthogonality error of the spin matrix SM. The current system spin matrix is stored in the array SPNR.

In some instances, it may be desirable to symplectify maps that are not fully symplectic. While the standard elements of COSY INFINITY are symplectic to close to machine precision, the low accuracy fringe-field modes (see Section 3.3.7) violate symplecticity noticeably. Depending on the coarseness of the measured field data, this may also occur in the general element discussed in Section 3.3.8. To a much lesser extent symplecticity is violated by intrinsic elements requiring numerical integration, like the high-precision fringe fields and the round lenses discussed in Section 3.3.6. The command

SY <M> ;

symplectifies the map M using the generating function (see below) which is most accurate for the given map.

Symplectic maps can be represented by at least one of four generating functions in mixed variables:

$$\begin{aligned}
&F_1(q_i,q_f) \text{ satisfying } (\vec{p}_i,\vec{p}_f) = (\vec{\nabla}_{q_i}F_1, -\vec{\nabla}_{q_f}F_1)\\
&F_2(q_i,p_f) \text{ satisfying } (\vec{p}_i,\vec{q}_f) = (\vec{\nabla}_{q_i}F_2, \vec{\nabla}_{p_f}F_2)\\
&F_3(p_i,q_f) \text{ satisfying } (\vec{q}_i,\vec{p}_f) = (-\vec{\nabla}_{p_i}F_3, -\vec{\nabla}_{q_f}F_3)\\
&F_4(p_i,p_f) \text{ satisfying } (\vec{q}_i,\vec{q}_f) = (-\vec{\nabla}_{p_i}F_4, \vec{\nabla}_{p_f}F_4)
\end{aligned}$$

In the generating function representation there are no interrelationships between the coefficients due to symplecticity like in the transfer map, so the generating function representation is more compact. Furthermore, it is often an important tool for the symplectification of tracking data [8] [14] [40]. The command

MGF <M> <F> <I> <IER> ;

attempts to compute the I th generating function of the specified map M. If IER is equal to zero, this generating function exists and is contained in F. If IER is nonzero, it does not exist. While in principle, any generating function that exists represents the map, especially for high order maps, certain inaccuracies often result for numerical reasons. If I is chosen to be -1, the generating function representing the linear part of the map best is determined. For I $= -2$, the generating function representing the whole map best is computed. The case I $= -2$ is very expensive computationally and should only be used in crucial cases for high orders. In both cases, on return I contains the number of the chosen generating function.

The map which corresponds to a generating function F of type I is obtained by

GFM <M> <F> <I> ;

Other redundancy free representations of symplectic transfer maps are Lie factorizations including the Dragt-Finn factorization [14] [3] [38]. They are based on Lie transformation operators of the form

$$\exp(:f:) = 1 + :f: + \frac{:f:^2}{2} + ...$$

where f is a function of the canonical coordinates q_i and p_i. The colon denotes a Poisson bracket waiting to happen, i.e. $:f:g = \{f,g\}$. When $\vec{x}_f$ describes a final set of canonical coordinates with $\vec{x} = (q_1,p_1,\ldots,q_n,p_n)$ and $\vec{x}_i$ describes an initial set, then $\vec{x}_f = \exp(:f:)\vec{x}_i$ is a symplectic mapping. Those Lie transformation operators have the property

$$e^{:f:}(g(\vec{x})) = g\left(e^{:f:}\vec{x}\right)$$

for any function $g : \mathbf{R}^{2n} \to \mathbf{R}$ with n being the dimension of the required configuration space. Therefore we find

$$e^{:f:}(e^{:g:}\vec{x}) = (e^{:g:}\vec{x}) \circ \left(e^{:f:}\vec{x}\right).$$

The circle $\circ$ symbolizes the composition of maps. Two composed symplectic maps are therefore represented by the product of their Lie transformation operators in reversed order. As an example, a symplectic map can be written in the form

$$\tilde{L}e^{:f_>:}\vec{x} + \vec{C}$$

were $\tilde{L}$ is an operator such that $\tilde{L}\vec{x}$ is the linear part of the map, $f_>$ is a polynomial in the x_i containing only orders higher than 2. Finally $\vec{C}$ represents the constant part of the map. As mentioned previously this representation is equal to

$$\left(e^{:f_>:}\vec{x}\right) \circ \left(\tilde{L}\vec{x}\right) + \vec{C}$$

Besides this factorization, there are various others that are similar and have certain advantages [3]. They are shown in the table below. As shown in [3], it is one of the strong points of the map representation and the differential algebraic techniques that the computation of these Dragt-Finn factorization is possible to arbitrary order with a relatively simple algorithm. It is actually much easier to compute them from the map than using Lie algebraic techniques alone. The command

MLF <MA> <C> <M> <F> <I> ;

computes the factorization from the transfer map MA. On return, the vector C contains the constant part, M the linear part and F contains the f_i from the list below. In case of the last four factorization, F has to be an array. I is the identifier of the factorization following the numbering in the list.

1: $\mathcal{M}(\vec{x}) =_n \tilde{L} \exp(:f_> :)\, \vec{x} + \vec{C}$

-1: $\mathcal{M}(\vec{x}) =_n \exp(:f_> :)\, \tilde{L}\vec{x} + \vec{C}$

2: $\mathcal{M}(\vec{x}) =_n \tilde{L} \exp(:f_3:) \exp(:f_4:) \ldots \exp(:f_{n+1}:)\, \vec{x} + \vec{C}$

-2: $\mathcal{M}(\vec{x}) =_n \exp(:f_{n+1}:) \ldots \exp(:f_3:)\, \tilde{L}\vec{x} + \vec{C}$

3: $\mathcal{M}(\vec{x}) =_{2^{n+1}} \tilde{L} \exp(:f_{3,3}:) \exp(:f_{4,5}:) \exp(:f_{6,9}:) \ldots \exp\left(:f_{(2^n+2),(2^{n+1}+1)}:\right) \vec{x} + \vec{C}$

-3: $\mathcal{M}(\vec{x}) =_{2^{n+1}} \exp\left(:f_{2^n+2,2^{n+1}+1}:\right) \ldots \exp(:f_{6,9}:) \exp(:f_{4,5}:) \exp(:f_{3,3}:)\, \tilde{L}\vec{x} + \vec{C}$

Here f_i denotes homogeneous polynomials of exact order i and $f_{i,j}$ polynomials with orders from i to j. Given a factorization, the command

LFM <MA> <C> <M> <F> <I> ;

calculates the according map. The command

LFLF <C> <M> <F> <P> <I> <J> ;

computes the factorization of type J with exponent P from a factorization of type I with exponent F. Without the map representation this would be a very elaborate task, because the Campbell-Baker-Hausdorff formula would be needed to the appropriate order.

5 Examples

This section provides several examples for the use of core features of COSY INFINITY. The code demo.fox which is distributed with COSY INFINITY contains many more programs that can serve as demonstrations. Further ideas how to use COSYScript can also be obtained by studying cosy.fox.

5.1 A Simple Sequence of Elements

After having discussed the particle optical elements and features available in COSY INFINITY in the previous sections, we now discuss the computation of maps of simple systems.

We begin with the computation of the transfer map of a quadrupole doublet to fifth order. Here the COSYScript input resembles the input of many other optics codes [18]. This example program is available as beamdemo_ele.fox at the COSY INFINITY download site.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
  OV 5 2 0 ; {order 5, phase space dim 2, # of parameters 0}
  RP 10 4 2 ; {kinetic energy 10MeV, mass 4 amu, charge 2}
  UM ;    {sets map to unity}
  DL .1 ; {drift of length .1 m}
  MQ .2 .1 .05 ; {focusing quad; length .2 m, field .1 T, aperture .05 m}
  DL .1 ;
  MQ .2 -.1 .05 ; {defocusing}
  DL .1 ;
  PM 6 ; {prints map to display}
  ENDPROCEDURE ;
RUN ; END ;
```

The first few lines of the resulting transfer map look like this:

```
  0.7084973    -0.1798231      0.000000       0.000000       0.000000      100000
  0.6952214     1.234984       0.000000       0.000000       0.000000      010000
   0.000000      0.000000      1.234984    -0.1798231      0.000000      001000
   0.000000      0.000000     0.6952214     0.7084973      0.000000      000100
 -0.7552786E-01-0.5173667E-01  0.000000       0.000000       0.000000      300000
  0.2751173     0.1728297      0.000000       0.000000       0.000000      210000
 -0.4105720    -0.2057599      0.000000       0.000000       0.000000      120000
  0.3541071     0.8137949E-01  0.000000       0.000000       0.000000      030000
   0.000000      0.000000     0.5676314E-01-0.5150461E-01  0.000000      201000
```

The different columns correspond to the final coordinates x, a, y, b and t. The lines contain the various expansion coefficients, which are identified by the exponents of the initial condition. For example, the third column, hence the final coordinate y, of the last line is the number `0.5676314E-01`, where the exponents are noted as `201000`, which means xxy. So, the value of the expansion coefficient (y, xxy) is 0.05676314.

5.2 Maps with Knobs

The DA approach easily allows to compute maps not only depending on phase space variables, but also on system parameters. This can be very helpful for different reasons. For example, it directly tells how sensitive the system is to errors in a particular quantity. In the same way it can be used to find out ideal positions to place correcting elements. Furthermore, it can be very helpful for the optimization of systems, and sometimes very fast convergence can be achieved with it (for details, see the Programmer's Manual [Optimization]).

In the context of COSY INFINITY, the treatment of such system parameters or knobs is particularly elegant.

In the following example, we compute the map of a system depending on the strength of one quadrupole. The COSY function **PARA(I)** is used, which identifies the quantity as parameter number I by turning it into an appropriate DA vector.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
   OV 5 2 1 ;              {order 5, phase space dim 2, parameters 1}
   RP 10 4 2 ;             {sets kinetic energy, mass and charge}
   UM ;
   DL .1 ;
   MQ .2 .1*PARA(1) .05 ;  {quadrupole; now field is a DA quantity}
   DL .1 ;
   MQ .2 -.1 .05 ;
   DL .1 ;
   PM 11 ;                 {prints map depending on quad strength to unit 11}
   ENDPROCEDURE ;
RUN ; END ;
```

Since COSYScript supports freedom of types at compile time, the second argument of the quad can be either real or DA. For details, consult the Programmer's Manual.

The idea of maps with knobs can also be used to compute the dependence on the particle mass and charge as well as on energy in case time of flight terms are not needed. In the following example, the map of the quad doublet is computed including the dependence on energy, mass and charge.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
   OV 5 2 3 ;              {order 5, phase space dim 2, parameters 3}
   RP 10*PARA(1) 4*PARA(2) 2*PARA(3) ;    {sets kinetic energy, mass
                                           and charge as DA quantities}
   UM ;
   DL .1 ;
   MQ .2 .1 .05 ;
   DL .1 ;
   MQ .2 -.1 .05 ;
   DL .1 ;
   PM 11 ;                 {prints map with dependence on energy,
                            mass and charge, to unit 11}
   ENDPROCEDURE ;
RUN ; END ;
```

5.3 Grouping of Elements

Usually it is necessary to group a set of elements together into a cell. For example, since most circular accelerators are built of several at least almost identical cells, it is desirable to refer to the cell as a block. Similar situations often occur for spectrometers or microscopes if similar quad multiplets are used repetitively.

Grouping is easily accomplished in COSY INFINITY by just putting the elements into a procedure. In the following example, the strength of a quadrupole in the cell of an accelerator is adjusted manually such that the motion in both planes is stable. Since the motions are stable if the two traces are less than two in magnitude, the map is printed to the screen which allows a direct check.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ; VARIABLE QS 1 ;     {declare a real variable}
   PROCEDURE CELL Q H1 H2 ;         {defines a cell of a ring}
      DL .3 ; DI 10 20 .1 0 0 0 0 ; DL .1 ; MH .1 H1 .05 ;
      DL .1 ; MQ .1 Q .05 ;     DL .3 ; MH .1 H2 .05 ;
      ENDPROCEDURE ;
   OV 3 2 0  ; RPP 1000 ;           {third order, one GeV protons}
   QS := .1 ;                       {set initial value for quad}
   WHILE QS#0 ; WRITE 6 ' GIVE QS ' ; READ 5 QS ;
      UM ; CELL QS 0 0 ;  PM 6 ; WRITE 6 ME(3,3) ;
      ENDWHILE ;
   ENDPROCEDURE ; RUN ; END ;
```

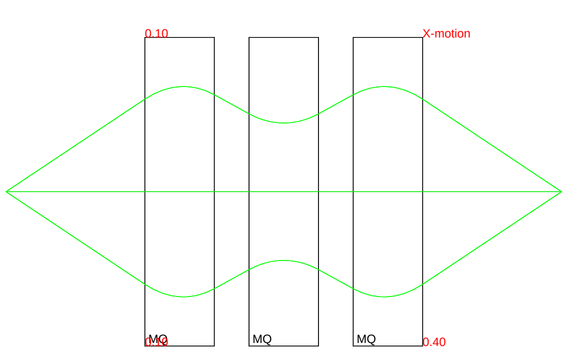
Such groupings can be nested if necessary, and parameters on which the elements in the group depend can be passed freely. Note that calling a group entails that all elements in it are executed; so grouping is not a means to reduce execution time, but a way to organize complicated systems into easily manageable parts. Reduction of execution time can be achieved by saving maps of subsystems that do not change using SM and AM discussed above.

5.4 Optimization

One of the most important tasks in the design of optical systems is the optimization of certain parameters of the system to meet certain specifications. Because of the importance of optimization, there is direct support from COSYScript via the **FIT** and **ENDFIT** commands. COSY INFINITY provides several Fortran based optimizers; a detailed description of the optimizers available in COSY INFINITY can be found in the Programmer's Manual [Optimization].

In the first example we illustrate a simple optimization task: to fit the strengths of the quadrupoles of a symmetric triplet to perform stigmatic point-to-point imaging. To monitor the optimization process, the momentary values of the quad strengths and the objective function are printed to the screen. Furthermore, a graphics display of the system at each step of the optimizer is displayed in two graphics windows, here identified with the graphics output units -1 and -2, one for each phase space projection, creating a movie-like effect. The Programmer's Manual [Supported Graphics Drivers] lists the graphics drivers currently supported in COSY INFINITY. At the end, the final pictures of the x and y projection of the system are printed in PDF format, identified with the graphics output unit -12, for inclusion in this manual. This example program is available as beamdemo_fit.fox at the COSY INFINITY download site.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
```



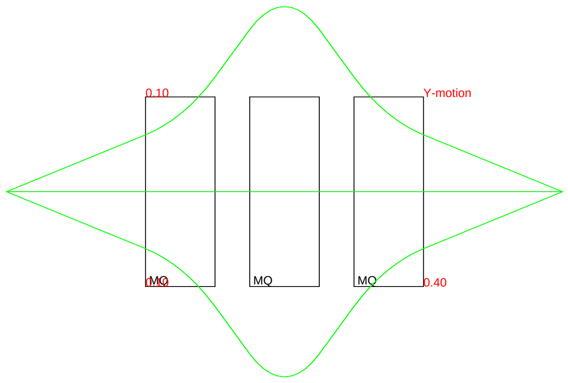


Figure 5: The x, y pictures of ray trajectories through a FIT–ENDFIT optimized quadrupole triplet. The pictures are the resulting PDF files pic001.pdf, pic002.pdf by executing the COSYScript example program beamdemo_fit.fox.

```
  VARIABLE Q1 1 ; VARIABLE Q2 1 ; VARIABLE OBJ 1 ;
  PROCEDURE TRIPLET A B ;
    MQ .1  A .05 ; DL .05 ; MQ .1 -B .05 ; DL .05 ; MQ .1  A .05 ;
    ENDPROCEDURE ;
  OV 1 2 0 ; RP 1 1 1 ;
  SB .15 .15 0 .15 .15 0 0 0 0 0 0 ;
    {sets half widths of beam .15 m in x, y and .15 rad in a, b}
  Q1 := .5 ; Q2 := .5 ; {start values of Q1, Q2}
  FIT Q1 Q2 ;
    UM ; CR ; {clears the rays}
    ER 1 3 1 3 1 1 1 1 ; {ensemble of rays, 3 in a, b}
    BP ; {begins a picture}
    DL .2 ; TRIPLET Q1 Q2 ; DL .2 ;
    EP ; {ends the picture}
    PG -1 -2 ; {outputs the x,y pictures to default windows}
    OBJ := ABS(ME(1,2))+ABS(ME(3,4)) ;
      {defines the objective OBJ.
       ME(1,2): map element (x,a),  ME(3,4): map element (y,b)}
    WRITE 6 'Q1, Q2: ' Q1 Q2  'OBJECTIVE: ' OBJ ;
    ENDFIT 1E-5 1000 1 OBJ ;
      {fits OBJ by Simplex algorithm. This is point-to-point for both x, y}
  PG -12 -12 ;
    {output final pictures to PDF files pic001.pdf and pic002.pdf}
  ENDPROCEDURE ;
RUN ; END ;
```

The final x, y pictures, also output to PDF files, are shown in Figure 5.

Besides providing “canned” optimization strategies, COSYScript allows to follow one’s own path of optimizing a system, which typically consists of several runs with varying parameters and subsequent optimizations.

In the following example, the goal is to vary several parameters of the system manually, fit the quad strengths, and then look at the spherical aberrations. This process is repeated by inputting different values for the parameters until the spherical aberrations have been reduced to a satisfactory level. When this is achieved, the pictures of the system are output directly to PostScript (PS) files, identified with the graphics output unit -10, with the names pic001.ps and pic002.ps.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
   VARIABLE Q1 1 ; VARIABLE Q2 1 ; VARIABLE L1 1 ; VARIABLE L2 1 ;
   VARIABLE OBJ 1 ; VARIABLE ISTOP 1 ;
   PROCEDURE TRIPLET ;
      UM ; CR ; ER 1 4 1 4 1 1 1 1 ; BP ;
      DL L1 ; MQ .1 Q1 .05 ; DL L2 ; MQ .1 -Q2 .05 ;
      DL L2 ; MQ .1 Q1 .05 ; DL L1 ; EP ; PP -1 0 0 ;
      ENDPROCEDURE ;
   OV 3 2 0 ; RP 1 1 1 ; SB .08 .08 0 .08 .08 0 0 0 0 0 0 ; ISTOP := 1 ;
   WHILE ISTOP#0 ;
      WRITE 6 ' GIVE VALUES FOR L1, L2: ' ; READ 5 L1 ; READ 5 L2 ;
      Q1 := .5 ; Q2 := .5 ; CO 1 ;
      FIT Q1 Q2 ; TRIPLET ; OBJ := ABS(ME(1,2))+ABS(ME(3,4)) ;
         ENDFIT 1E-5 1000 1 OBJ ;
         CO 3 ; TRIPLET ;
         WRITE 6 ' SPHERICAL ABERRATION FOR THIS SYSTEM: ' ME(1,222) ;
         WRITE 6 ' CONTINUE SEARCH? (1/0) ' ; READ 5 ISTOP ;
      ENDWHILE ; PP -10 0 0 ; PP -10 0 90 ;
   ENDPROCEDURE ; RUN ; END ;
```

This example shows how it is possible to phrase more complicated interactive optimization tasks in COSYScript. One can even go far beyond the level of sophistication displayed here; by nesting sufficiently many **WHILE**, **IF**, and **LOOP** statements, it is often possible to optimize a whole system in one interactive session without ever leaving COSY INFINITY. For example, the first order design in [5] which is subject to quite a number of constraints and requires a sophisticated combination of trial and optimization was performed in this way.

5.5 Normal Form, Tune Shifts and Twiss Parameters

The following example shows the use of normal form methods and parameter dependent Twiss parameters for the analysis of a repetitive system. For the sake of simplicity, we choose here a simple FODO cell that is described by the procedure CELL. The map of the cell is computed to fifth order, with the energy as a parameter. In the cell itself, the quadrupole strength is another parameter.

As a first step, the parameter dependent tunes are computed and written to the output unit 7, following the algorithm in [11]. Next follow the tunes depending on parameters and amplitude; this is done with DA normal form theory [10] [14]. Finally, several other quantities and their parameter dependence are computed using the procedure **GT**. They include the parameter dependent fixed point, the parameter dependent Twiss parameters, as well as the parameter dependent damping (which here is unity because no radiation effects are taken into account).

```
INCLUDE 'COSY' ;
PROCEDURE RUN  ;
   VARIABLE A 100 2 ; VARIABLE B 100 2 ; VARIABLE G 100 2 ;
   VARIABLE R 100 2 ; VARIABLE MU 100 2 ; VARIABLE F 100 6 ;
   PROCEDURE CELL ;
      DL .1 ; DI 1 45 .1 0 0 0 0 ; DL .1 ; MQ .1 -.1*PARA(2) .1 ; DL .2 ;
      ENDPROCEDURE ;
   OV 5 2 2 ; RP 1*PARA(1) 1 1 ; UM ; CELL ;
   TP MU ; WRITE 7 ' DELTA DEPENDENT TUNES '         MU(1) MU(2) ;
   TS MU ; WRITE 7 ' DELTA AND EPS DEPENDENT TUNES ' MU(1) MU(2) ;
   GT MAP F MU A B G R  ;
   WRITE 7 ' DELTA DEPENDENT FIXED POINT ' F(1) F(2) F(3) F(4) ;
   WRITE 7 ' DELTA DEPENDENT ALPHAS '      A(1) A(2) ;
   WRITE 7 ' DELTA DEPENDENT BETAS '       B(1) B(2) ;
   WRITE 7 ' DELTA DEPENDENT GAMMAS '      G(1) G(2) ;
   WRITE 7 ' DELTA DEPENDENT DAMPINGS '    R(1) R(2) ;
   ENDPROCEDURE ; RUN ; END ;
```

5.6 Repetitive Tracking

In the following example, we want to study the nonlinear behavior of a ring by a qualitative analysis of tracking data using the command **TR**. The ring consists of 18 identical cells. Nine of these cells are packed into a half ring by the procedure HALFRING. At execution, the system asks for the values of the strengths of the two hexapoles which influence its degree of nonlinearity. The tracking data for each setting are displayed in a graphics window, here identified with the graphics output unit -1, and then also output to a PostScript (PS) file with the graphics output unit -10, which lists the setting information via the command **TRT**. One example case is shown in Figure 6.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ; VARIABLE QS 1 ; VARIABLE H1 1 ; VARIABLE H2 1 ; VARIABLE N 1 ;
   PROCEDURE CELL Q H1 H2 ;         {defines a cell of a ring}
      DL .3 ; DI 10 20 .1 0 0 0 0 ; DL .1 ; MH .1 H1 .05 ;
      DL .1 ; MQ .1 Q .05 ;     DL .3 ; MH .1 H2 .05 ;
      ENDPROCEDURE ;
   PROCEDURE HALFRING Q H1 H2 ; VARIABLE I 1 ;
      LOOP I 1 9 ; CELL Q H1 H2 ; ENDLOOP ; ENDPROCEDURE ;
   OV 3 2 0  ; RPP 1000 ;     {third order, one GeV protons}
   QS := -.05 ; H1 := .01 ;
   WHILE H1#0 ; WRITE 6 ' GIVE HEXAPOLE STRENGTHS ' ; READ 5 H1 ; READ 5 H2 ;
      UM ; HALFRING QS H1 H2 ;
      WRITE 6 ' GIVE NUMBER OF TURNS ' ; READ 5 N ;
      CR ;
      SR .005 0 .005 0 0 0 0 0 1 ;
      SR .01  0 .01  0 0 0 0 0 1 ;
      SR .015 0 .015 0 0 0 0 0 1 ;
      SR .02  0 .02  0 0 0 0 0 1 ;
      TR N 1  1 2  .03 .002   0 0 -1 ;
      TRT SI(N)&' turns, H1='&SF(H1,'(F6.3)')&', H2='&SF(H2,'(F6.3)') ;
      TR N 1  1 2  .03 .002   0 0 -10 ; {output to a PS file}
      ENDWHILE ;
   ENDPROCEDURE ; RUN ; END ;
```

Instead of PostScript (PS), the user may want to output directly to a PDF file (the graphics unit number -12). Refer to the Programmer's Manual [Supported Graphics Drivers] for the graphics drivers

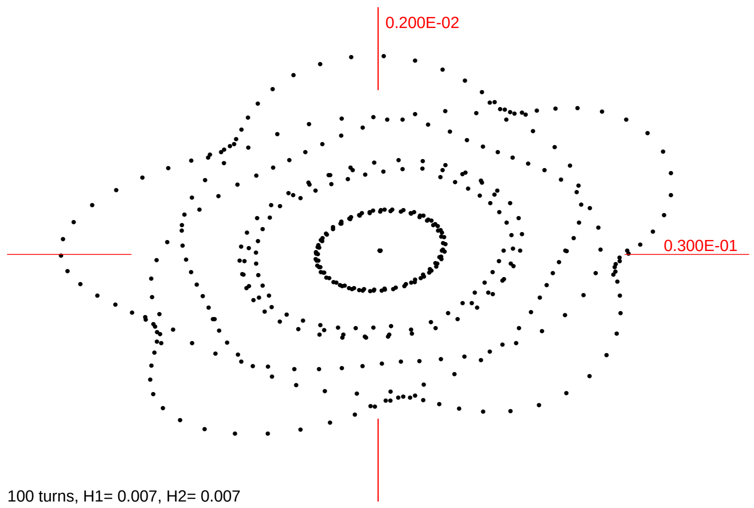


Figure 6: An example tracking picture, produced by the command TR using an example COSYScript program listed in Section 5.6. The resulting PostScript (PS) file is hand-edited to increase the size of the dots for the inclusion on this page. Follow the instructions in Section 5.6 for hand-editing.

currently supported in COSY INFINITY.

For the purpose of efficiency, the command **TR** outputs dots to the graphics output via the COSY intrinsic procedure GRDOT (see the Programmer's Manual). The font size of the dots is adjusted to appear comfortable in US letter size or A4 size, even for large scale production runs using TR. However, when a TR graphics output picture file is to be included with much smaller size in a document, like this manual or scientific papers, the dots may become too small to be displayed meaningfully. In such a situation, the font size of the dots can be increased by hand-editing some of the TR produced graphics output picture files. Below, the instructions are given how to hand-edit PostScript (PS) and PDF graphics output picture files produced by COSY INFINITY via the graphics output unit numbers -10 and -12, both of which are ASCII text files, thus the user can hand-edit using an ASCII text editor. The hand-editing instructions are current as of August 2013.

How to increase the size of dots – PostScript (PS) (the graphics output unit number -10)

1. Using an ASCII text editor, open the TR produced PostScript (PS) file. The file must be the one directly obtained via the graphics output unit number -10 by executing COSY INFINITY.
2. Search lines containing the word "`scalefont`".
3. Identify the "`scalefont`" command line that determines the font size of dots. This should be the 22nd line (or thereabouts) of the PS file, and it is the **second** "`scalefont`" command line. Note that it is not the first "`scalefont`" command line, which determines the default font size of the PS file. It also can be located as the line just before the first dot outputting line, where a dot is represented by "`<B7>`" in the PS file.
4. Comment out the "`scalefont`" command line identified by the previous step. For this, place a "`%`" mark in the beginning of the line, and the line will look like:
   `%/Helvetica findfont 0.005200000 scalefont setfont`
5. Save and close the PS file.

How to increase the size of dots – PDF (the graphics output unit number -12)

1. Using an ASCII text editor, open the PDF file produced by TR. The file must be the one directly obtained via the graphics output unit number -12 by executing COSY INFINITY. Different from most other pdf producing tools, COSY purposely outputs its pdf graphics as ASCII to allow for later (expert) modification as necessary. When the user is not used to the process here, we recommend to make a backup copy of the PDF file before proceeding further.

2. There are several topics requiring extra care when handling PDF files.

   (a) A PDF file contains the byte sizes of its contents as crucial (and sensitive!!) information. When the byte size is altered, the file may get corrupted. Thus, hand-editing has to be performed very carefully so that the total character count is not changed. If the file is damaged, start over utilizing the backup copy of the original PDF file.

   (b) PDF files are sensitive to the action of saving/writing. While one software program such as Acrobat is opening the PDF file, another software program such as an ASCII text editor cannot save/write to the same PDF file.

   (c) Most software programs that can handle/open PDF files such as Acrobat use binary form when saving/writing. This is the case even if the original PDF file is an ASCII text file like the one produced by TR discussed here. Thus any saving/writing by any other software program other than an ASCII text editor must be avoided during the process discussed here.

3. Search lines containing the word "`Tf`".

4. Identify the "`Tf`" command line that determines the font size of dots. This should be the 8th line (or thereabouts) of the PDF file, and it is the **second** "`Tf`" command line. Note that it is not the first "Tf" command line that determines the default font size of the PDF file. It also can be located as the line just before the first dot outputting line, where a dot is represented by "`<B7>`" in the PDF file.

5. Comment out the "`Tf`" command line identified by the previous step. For this, replace the first character of the line by a "`%`" mark. Originally the identified "`Tf`" command line looks like this:
   `/F1 0.008000000 Tf`
   After the proper change, the line will look like:
   `%F1 0.008000000 Tf`
   As cautioned in the step 2, this change has to be performed carefully not to alter the number of characters in this line.

6. Save and close the PDF file.

5.7 Introducing New Elements

When looking into the physics part of COSY INFINITY, it becomes apparent that all particle optical elements described above are nothing but procedures written in COSYScript. Due to the openness of the approach, users can construct their own particle optical elements.

Here we want to show how a user can define his own particle optical element and work with it. As a first example, we begin with a skew quadrupole that is rotated against the regular orientation by the angle ϕ. The action of such a quad can be obtained by first rotating the map by $-\phi$, then let the quad act, and finally rotate back. All these steps are performed on the DA variable containing the momentary value of the transfer map, which is the global COSY array MAP. For the conversion of degrees to radians, the global COSY variable DEGRAD is used. Note that many important global variables of COSY are described in Section 5.8.

```
INCLUDE 'COSY' ;
PROCEDURE RUN ;
   PROCEDURE SQ PHI L B D ;    {computes the action of a skew quad}
      PROCEDURE ROTATE PHI ;         {local procedure for rotation}
         VARIABLE M 1000 4 ; VARIABLE I 1 ;
         M(1) :=  COS(PHI*DEGRAD)*MAP(1) + SIN(PHI*DEGRAD)*MAP(3) ;
         M(3) := -SIN(PHI*DEGRAD)*MAP(1) + COS(PHI*DEGRAD)*MAP(3) ;
         M(2) :=  COS(PHI*DEGRAD)*MAP(2) + SIN(PHI*DEGRAD)*MAP(4) ;
         M(4) := -SIN(PHI*DEGRAD)*MAP(2) + COS(PHI*DEGRAD)*MAP(4) ;
         LOOP I 1 4 ; MAP(I) := M(I) ; ENDLOOP ; ENDPROCEDURE ;
      ROTATE -PHI ; MQ L B D ; ROTATE PHI ; ENDPROCEDURE ;
   OV 5 2 0 ; RP 1 1 1 ;
   UM ; DL .1 ; SQ -30 .2 .1 .1 ; DL .1 ; SQ 30 .2 .1 .1 ; PM 6 ;
   ENDPROCEDURE ; RUN ; END ;
```

It is clear that a similar technique can be used to study misaligned elements. In a similar way, it is easily possible to generate a “kick-environment” in COSY INFINITY, where every particle optical element is just represented by a kick in its center.

This technique is also useful in many other ways. For example, if a certain element is rather time consuming to compute, which can be the case with cylindrical lenses to high orders, one can write a procedure that computes the map of the element, including the dependence on some of its parameters, and saves the map somewhere. When called again with different values, the procedure decides if the values are close enough to the old ones to just utilize the previously computed map with the parameters plugged in, or if it is necessary to compute the element again. In case the parameters are varied only slightly, a very significant speed up can be achieved in this way, yet for the user the procedure looks like any other element.

5.8 Introducing New Features

The whole concept of COSY INFINITY is very open in that it easily allows extensions for specific tasks. The user is free to provide his own procedures for particle optical elements or for many other purposes. To interface with COSY INFINITY most efficiently, it is important to know the names of certain key global variables, functions and procedures. Furthermore it is important to know that all quantities in COSY INFINITY are in SI units, with the exception of voltages, which are in kV.

For some applications, it is helpful to access some of COSY INFINITY’s global variables. Since the physics of the code is written in its own language, all these variables are directly visible to the user. The first set of relevant global variables are the natural constants describing the physics. These variables are set after the routine **OV** or **DEF** is called and can be utilized for calculations by the user. The data are taken from [45] (**CAUTION:** The data was updated in September 2001 in cosy.fox). In order to match other codes, the variables can be changed by the user in cosy.fox if necessary.

AMU	Atomic Mass Unit	$1.66053873 \cdot 10^{-27}$ kg
AMUMEV	Atomic Mass Unit in MeV	computed as AMU$\cdot c^2/e \cong$ 931.4940136 MeV
EZERO	The charge unit e	$1.602176462 \cdot 10^{-19}$ C
CLIGHT	The speed of light c	$2.99792458 \cdot 10^{8}$ m/s
PI	the value of π	3.14159265358979323846264338327950288 42

The second set of variables describes the reference particle. These variables are updated every time the procedure **RP** is called.

E0	Energy in MeV
M0	Mass in AMU
Z0	Charge in units
V0	Velocity in m/s
P0	Momentum p_0c in MeV
CHIM	Magnetic Rigidity
CHIE	Electric Rigidity
ETA	Kinetic Energy over mc^2

Finally, there are the variables that are updated by particle optical elements:

MAP	Array of 8 DA vectors containing Map
RAY	Array of 8 VE vectors containing Coordinates
SPOS	Momentary value of the independent variable

COSY INFINITY contains several procedures that are not used explicitly by the user but are used internally for certain operations. Firstly, there are the three DA functions

DER(<n>,<a>)

INTEG(<n>,<a>)

PB(<a>,<b>)

which compute the DA derivation with respect to variable n, the integral with respect to variable n, and the Poisson bracket between a and b. Another helpful function is

NMON(<NO>,<NV>)

which returns the maximum number of coefficients in a DA vector in NV variables to order NO. An important procedure is

POLVAL <L> <P> <NP> <A> <NA> <R> <NR> ;

where <P>, <A>, and <R> are arrays, and **POLVAL** lets the polynomial described by the NP DA vectors or Taylor models stored in the array P act on the NA arguments A, and the result is stored in the NR Vectors R.

In the normal situation, L should be set 1. After **POLVAL** has already been called with L= 1, and if it is called with the same polynomial array P again, a certain part of internal analysis of P can be avoided by calling **POLVAL** with L= -1 or L= 0. (There are other advanced settings for L, but their use is discouraged for normal users because they may interfere with the internal use of **POLVAL** of various COSY tools.)

The type of A is free, but all the array elements of A have to be the same type; it can be either DA or CD, in which case the procedure acts as a concatenator, it can be real, complex or intervals, in which case it acts like a polynomial evaluator, or it can be of vector type VE, in which case it acts as a very efficient vectorizing map evaluator and is used for repetitive tracking. If necessary, adding `0*A(1)` to subsequent array elements `A(I)` can make the type of the argument array element agree to that type of `A(1)`.

Further details on using the COSY INFINITY environment for active programming tasks can be found in the Programmer's Manual.

6 Acknowledgements

For very valuable help with an increasing number of parts of the program, we would like to thank Meng Zhao, Weishi Wan, Georg Hoffstätter, Ralf Degenhardt, Nina Golubeva, Vladimir Balandin, Jens Hoefkens, Alexander Ovsyannikov, Béla Erdélyi, Laura Chapin, Shashikant Manikonda, Pierluigi Di Lizia, Roberto Armellin, Youn-Kyung Kim, Pavel Snopok, Alexey Poklonskiy, Johannes Grote, Alexander Wittig, He Zhang, Ravi Jagasia, Eremey Valetov, David Tarazona, and Adrian Weisskopf, who all at various times were at Michigan State University. We would also like to thank numerous COSY users for providing valuable feedback, many good suggestions, and streamlining the implementation on various machines, and our special thanks go to Carol Johnstone, Markus Neher, George Corliss and Nathalie Revol. We would like to thank Jorge More for providing the public domain optimizer LMDIF.

COSY INFINITY makes use of the following programs and libraries on various platforms:
GRWin by Tsuguhiro Tamaribuchi for plotting on Windows, AquaTerm for plotting on Mac OS X, PGPLOT for plotting on Linux, WeFunction icon set from www.wefunction.com for GUI icons in the COSY GUI Java program.

Financial support was appreciated from the U.S. Department of Energy, the U.S. National Science Foundation, the Deutsche Forschungsgemeinschaft, Michigan State University, the National Superconducting Cyclotron Laboratory, University of Gießen, the SSC Central Design Group, Lawrence Berkeley National Laboratory, Los Alamos National Laboratory, Fermilab, Argonne National Laboratory, the Alfred P. Sloan Foundation, and the Studienstiftung des Deutschen Volkes.

7 COSYScript

The COSYScript language is based on a **minimal and compact syntax**. Experience shows that the COSY Syntax Table combined with some examples usually allows users to work with COSYScript within minutes. Refer to the Programmer's Manual for further details.

COSYScript is **object oriented** with **parametric polymorphism** (dynamical type assignment). The language is compiled and linked to a meta-format on the fly and immediately executed. Combined with the ability to include pre-compiled code, this leads to a **very rapid turnaround** from input completion to execution. Combined with built-in tools for **optimization**, this makes the tool particularly suitable for **simulation**, as a control language, and for **fast prototyping**.

Great emphasis is put on **performance**, evidenced by negligible overhead to the cost of the operations on the types. COSYScript usually outperforms code based on the C++ and F90 interfaces discussed in the Programmer's Manual.

7.1 COSYScript Syntax Table

BEGIN ;		**END** ;
VARIABLE <name> <length> ;		
PROCEDURE <arguments> ;		**ENDPROCEDURE** ;
FUNCTION <arguments> ;		**ENDFUNCTION** ;
<name>	**:=**	<expression> ; (Assignment)
IF <expression> ;	**ELSEIF** <expression> ;	**ENDIF** ;
WHILE <expression> ;		**ENDWHILE** ;
LOOP <name> <beg> <end> ;		**ENDLOOP** ;
PLOOP <name> <beg> <end> ;		**ENDPLOOP** <comm. rules> ;
FIT <variables> ;		**ENDFIT** <parameters, objectives> ;
WRITE <unit> <expressions> ;		**READ** <unit> <names> ;
SAVE <filename> ;		**INCLUDE** <filename> ;

References

[1] V. Balandin, M. Berz, and N. Golubeva. Computation and analysis of spin dynamics. *AIP CP*, 391:276, 1996.

[2] M. Berz. Differential algebraic description of beam dynamics to very high orders. *Particle Accelerators*, 24:109, 1989.

[3] M. Berz. Arbitrary order description of arbitrary particle optical systems. *Nuclear Instruments and Methods*, A298:426, 1990.

[4] M. Berz. Computational aspects of optics design and simulation: COSY INFINITY. *Nuclear Instruments and Methods*, A298:473, 1990.

[5] M. Berz. Isochronous beamlines for free electron lasers. *Nuclear Instruments and Methods*, A298:106–112, 1990.

[6] M. Berz. Forward algorithms for high orders and many variables. In *Automatic Differentiation of Algorithms: Theory, Implementation and Application*, pages 147–156. SIAM, Philadelphia, 1991.

[7] M. Berz. *High-Order Computation and Normal Form Analysis of Repetitive Systems, in: M. Month (Ed), Physics of Particle Accelerators*, volume 249, page 456. American Institute of Physics, New York, 1991.

[8] M. Berz. Symplectic tracking in circular accelerators with high order maps. In *Nonlinear Problems in Future Particle Accelerators*, page 288. World Scientific, 1991.

[9] M. Berz. Automatic differentiation as non-Archimedean analysis. In *Computer Arithmetic and Enclosure Methods*, page 439, Amsterdam, 1992. Elsevier Science Publishers.

[10] M. Berz. Differential algebraic formulation of normal form theory. In *M. Berz, S. Martin and K. Ziegler (Eds.), Proc. Nonlinear Effects in Accelerators*, page 77, London, 1992. IOP Publishing.

[11] M. Berz. Direct computation and correction of chromaticities and parameter tune shifts in circular accelerators. In *Proceedings XIII International Particle Accelerator Conference, JINR D9-92-455*, pages 34–47(Vol.2), Dubna, 1992.

[12] M. Berz. Differential algebraic description and analysis of spin dynamics. *AIP CP*, 343, 1995.

[13] M. Berz. Differential algebras with remainder and rigorous proofs of long-term stability. *AIP CP*, 391:221, 1996.

[14] M. Berz. *Modern Map Methods in Particle Beam Physics.* Academic Press, San Diego, 1999. Also available at https://www.bmtdynamics.org/pub.

[15] M. Berz, B. Erdélyi, and K. Makino. Fringe field effects in small rings of large acceptance. *Physical Review ST-AB*, 3:124001, 2000.

[16] M. Berz, B. Erdélyi, and K. Makino. Towards accurate simulation of fringe field effects. *Nuclear Instruments and Methods A*, 472,3:533–540, 2001.

[17] M. Berz, B. Erdélyi, W. Wan, and K. Ng. Differential algebraic determination of high-order off-energy closed orbits, chromaticities, and momentum compactions. *Nuclear Instruments and Methods*, A427:310–314, 1999.

[18] M. Berz, H. C. Hofmann, and H. Wollnik. COSY 5.0, the fifth order code for corpuscular optical systems. *Nuclear Instruments and Methods*, A258:402–406, 1987.

[19] M. Berz, K. Joh, J. A. Nolen, B. M. Sherrill, and A. F. Zeller. Reconstructive correction of aberrations in nuclear particle spectrographs. *Physical Review C*, 47,2:537, 1993.

[20] M. Berz and K. Makino. Arbitrary order maps, remainder terms, and long term stability in particle accelerators. In *Charged Particle Optics III*, volume 3155, pages 186–192. SPIE, 1997.

[21] M. Berz and K. Makino. Normal form methods and optimization for nonlinear properties of cooling channels - part I. In *Proc. of the APS/DPF/DPB Summer Study on the Future of Particle Physics (Snowmass 2001)*, number T504, 2002. http://www.slac.stanford.edu/econf/C010630/papers/T504.PDF.

[22] M. Berz and K. Makino. Normal form methods and optimization for nonlinear properties of cooling channels - part II. In *Proc. of the APS/DPF/DPB Summer Study on the Future of Particle Physics (Snowmass 2001)*, number T701, 2002. http://www.slac.stanford.edu/econf/C010630/papers/T701.PDF.

[23] M. Berz and K. Makino. Recent advances in differential algebraic methods. In *Proc. of the APS/DPF/DPB Summer Study on the Future of Particle Physics (Snowmass 2001)*, number T509, 2002. http://www.slac.stanford.edu/econf/C010630/papers/T509.PDF.

[24] M. Berz and K. Makino. Constructive generation and verification of Lyapunov functions around fixed points of nonlinear dynamical systems. *International Journal of Computer Research*, 12,2:235–244, 2003.

[25] M. Berz and K. Makino. New approaches for the validation of transfer maps using remainder-enhanced differential algebra. *Nuclear Instruments and Methods A*, 519:53–62, 2004.

[26] M. Berz and K. Makino. Advanced computational methods for nonlinear spin dynamics. *IOP Journal of Physics*, 295:012143, 2011. doi:10.1088/1742-6596/295/1/012143.

[27] M. Berz and K. Makino. COSY INFINITY Version 10.2 programmer's manual. Technical Report MSUHEP20221203, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, 2023. See also https://cosyinfinity.org.

[28] M. Berz, K. Makino, and B. Erdélyi. Fringe field effects in muon rings. *AIP CP*, 530:38–47, 2000.

[29] M. Berz, K. Makino, and C. J. Johnstone. Propagation of a large-emittance muon beam through a straight, quadrupole-based precooling channel. In *Neutrino Factories and Superbeams*, volume 721, page 413. AIP Conference Proceedings, 2004.

[30] M. Berz, K. Makino, and Y.-K. Kim. Long-term stability of the Tevatron by validated global optimization. *Nuclear Instruments and Methods*, 558:1–10, 2006.

[31] M. Berz, K. Makino, and W. Wan. *An Introduction to Beam Physics.* CRC Press, Taylor & Francis Group, London, Boca Raton, 2014.

[32] 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*, A258:364–373, 1987.

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

[34] K. L. Brown and J. E. Spencer. Non-linear optics for the final focus of the single-pass-collider. *IEEE Transactions on Nuclear Science*, NS-28,3:2568, 1981.

[35] J. A. Caggiano, D. Bazin, B. S. Davids, R. Foutus, D. Karnes, P. Johnson, B. Sherrill, and A. Zeller. S800 spectrograph dipole mapping. Annual Report of the Michigan State University National Superconducting Cyclotron Laboratory, 1996.

[36] D. C. Carey. *The Optics of Charged Particle Beams.* Harwood Academic, New York, 1987, 1992.

[37] L. M. Chapin, J. Hoefkens, and M. Berz. The COSY language independent architecture: Porting COSY source files. *IOP CP*, 175:37–45, 2002.

[38] A. J. Dragt and J. M. Finn. Lie series and invariant functions for analytic symplectic maps. *Journal of Mathematical Physics*, 17:2215, 1976.

[39] 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.

[40] B. Erdélyi. *Symplectic Approximation of Hamiltonian Flows and Accurate Simulation of Fringe Field Effects.* PhD thesis, Michigan State University, East Lansing, Michigan, USA, 2001.

[41] B. Erdélyi and M. Berz. Optimal symplectic approximation of Hamiltonian flows. *Physical Review Letters*, 87,11:114302, 2001.

[42] B. Erdélyi and M. Berz. Local theory and applications of extended generating functions. *International Journal of Pure and Applied Mathematics*, 11,3:241–282, 2004.

[43] D. Errede, K. Makino, M. Berz, C. J. Johnstone, and A. van Ginneken. Stochastic processes in muon ionization cooling. *Nuclear Instruments and Methods A*, 519:466–471, 2004.

[44] M. M. Alsharo'a et al. Recent progress in neutrino factory and muon collider research within the muon collaboration. *Physical Review ST-AB*, 6:081001, 2003.

[45] Yao et al. Review of particle physics. *Journal of Physics G*, 33:1, 2006.

[46] S. Ozaki et al. for the Muon Collaboration. Feasibility study-II of a muon-based neutrino source. Technical Report 52623, Muon Collider Collaboration, BNL, 2001.

[47] E. Forest, M. Berz, and J. Irwin. Normal form methods for complicated periodic systems: A complete solution using Differential algebra and Lie operators. *Particle Accelerators*, 24:91, 1989.

[48] A. Geraci, T. Barlow, M. Portillo, J. Nolen, K. Shepard, M. Berz, and K. Makino. High-order maps with acceleration for optimization of electrostatic and radio-frequency ion-optical elements. *Review of Scientific Instruments*, 73,9:3174–3180, 2002.

[49] W. Glaser. *Grundlagen der Elektronenoptik.* Springer, Wien, 1952.

[50] H. Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, MA, second edition, 1980.

[51] J. Grote, M. Berz, and K. Makino. High-order representation of Poincare maps. *Lecture Notes on Computational Science and Engineering*, 50:59–66, 2005.

[52] J. Grote, M. Berz, and K. Makino. High-order DA methods for the determination of Poincare sections. *Nuclear Instruments and Methods*, 558,1:106–111, 2006.

[53] J. Grote, K. Makino, and M. Berz. Verified computation of high-order Poincaré maps. *Transactions on Systems*, 4,11:1986–1992, 2005.

[54] P. W. Hawkes and E. Kasper. *Principles of Electron Optics*, volume 1-3. Academic Press, London, 1996.

[55] G. Hoffstätter and M. Berz. Efficient computation of fringe fields using symplectic scaling. *AIP CP*, 297:467, 1993.

[56] G. Hoffstätter and M. Berz. Symplectic scaling of transfer maps including fringe fields. *Physical Review E*, 54(5):5664–5672, 1996.

[57] G. H. Hoffstätter. *Rigorous Bounds on Survival Times in Circular Accelerators and Efficient Computation of Fringe–Field Transfer Maps.* PhD thesis, Michigan State University, East Lansing, Michigan, USA, 1994. also DESY 94-242.

[58] C. Iselin. MAD - a reference manual. Technical Report LEP-TH/85-15, CERN, 1985.

[59] C. Iselin and J. Niederer. The MAD program, version 7.2, user's reference manual. Technical Report CERN/LEP-TH/88-38, CERN, 1988.

[60] X. Jiye. *Aberration Theory in Electron and Ion Optics.* Advances in Electronics and Electron Physics, Supplement 17. Academic Press, Orlando, Florida, 1986.

[61] C. J. Johnstone, M. Berz, D. Errede, and K. Makino. Optimization and beam control in large-emittance accelerators: Neutrino factories. In *2003 International Conference Physics and Control*, pages 964–973. IEEE, 2003.

[62] C. J. Johnstone, M. Berz, D. Errede, and K. Makino. Muon beam ionization cooling in a linear quadrupole channel. *Nuclear Instruments and Methods A*, 519:472–482, 2004.

[63] C. J. Johnstone, M. Berz, and K. Makino. Staging acceleration and cooling in a neutrino factory. *Nuclear Instruments and Methods*, 558,1:282–291, 2006.

[64] S. Kowalski and H. Enge. RAYTRACE. Technical report, MIT, Cambridge, Massachussetts, 1985.

[65] C. O. Maidana, M. Berz, and K. Makino. Muon beam ring cooler simulations using COSY INFINITY. *IOP CP*, 175:211–218, 2002.

[66] K. Makino. *Rigorous Analysis of Nonlinear Motion in Particle Accelerators.* PhD thesis, Michigan State University, East Lansing, Michigan, USA, 1998. Also MSUCL-1093.

[67] K. Makino and M. Berz. Arbitrary order aberrations for elements characterized by measured fields. In *Charged Particle Optics III*, volume 3155, pages 221–227. SPIE, 1997.

[68] K. Makino and M. Berz. COSY INFINITY version 8. *Nuclear Instruments and Methods*, A427:338–343, 1999.

[69] K. Makino and M. Berz. Effects of kinematic correction on the dynamics in muon rings. *AIP CP*, 530:217–227, 2000.

[70] K. Makino and M. Berz. Perturbative equations of motion and differential operators in nonplanar curvilinear coordinates. *International Journal of Applied Mathematics*, 3,4:421–440, 2000.

[71] K. Makino and M. Berz. Recent applications of COSY to nonlinear beam dynamics problems. In *Proc. of the APS/DPF/DPB Summer Study on the Future of Particle Physics (Snowmass 2001)*, number T510, 2002. http://www.slac.stanford.edu/econf/C010630/papers/T510.PDF.

[72] K. Makino and M. Berz. Solenoid elements in COSY INFINITY. *IOP CP*, 175:219–228, 2002.

[73] K. Makino and M. Berz. Tetra cooler ring simulation in COSY INFINITY. In *Neutrino Factories and Superbeams*, volume 721, page 418. AIP Conference Proceedings, 2004.

[74] K. Makino and M. Berz. COSY INFINITY version 9. *Nuclear Instruments and Methods*, 558:346–350, 2006.

[75] K. Makino and M. Berz. Dynamics in electrostatic rings via high-order transfer maps. *Microscopy and Microanalysis*, 21 Suppl. 4:36, 2015.

[76] K. Makino, M. Berz, D. Errede, and C. J. Johnstone. High order map treatment of superimposed cavities, absorbers, and magnetic multipole and solenoid fields. *Nuclear Instruments and Methods A*, 519:162–174, 2004.

[77] K. Makino, D. Errede, and M. Berz. Cooling channel simulations based on map methods. In *Proc. of the APS/DPF/DPB Summer Study on the Future of Particle Physics (Snowmass 2001)*, number T702, 2002. http://www.slac.stanford.edu/econf/C010630/papers/T702.PDF.

[78] S. L. Manikonda and M. Berz. An accurate high-order method to solve the Helmholtz boundary value problem for the 3D Laplace equation. *International Journal of Pure and Applied Mathematics*, 23,3:365–378, 2005.

[79] S. L. Manikonda, M. Berz, and K. Makino. High-order verified solution of the 3D Laplace equation. *Transactions on Computers*, 4-11:1604–1610, 2005.

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

[81] K. Y. Ng. Reliability of α_1 and α_2 from lattice codes. In *Proc. of Snowmass 96: New Directions for High Energy Physics*. Snowmass, Colorado, June 25-July 12, 1996.

[82] A. A. Poklonskiy, D. Neuffer, C. J. Johnstone, M. Berz, K. Makino, D. A. Ovsyannikov, and A. D. Ovsyannikov. Optimizing adiabatic bunchers and phase rotators. *Nuclear Instruments and Methods*, 558:135–141, 2006.

[83] G. Sabbi. Magnetic field analysis of HGQ coil ends. Technical Report TD-97-040, Fermilab, 1997.

[84] M. L. Shashikant, M. Berz, and B. Erdélyi. COSY INFINITY's EXPO symplectic tracking for LHC. *IOP CP*, 175:299–305, 2002.

[85] P. Snopok, M. Berz, K. Makino, and C. Johnstone. Simulation and optimization of the Tevatron accelerator. *Lecture Notes on Computational Science and Engineering*, 50:199–209, 2005.

[86] P. V. Snopok, C. J. Johnstone, M. Berz, D. A. Ovsyannikov, and A. D. Ovsyannikov. Study and optimal correction of a systematic skew quadrupole field in the Tevatron. *Nuclear Instruments and Methods*, 558:142–146, 2006.

[87] Pavel Snopok. A converter program for Tevatron lattices from OptiM to COSY INFINITY. Technical Report MSUHEP-40909, Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, 2004.

[88] K. G. Steffen. *High Energy Beam Optics*. Wiley-Interscience, New York, 1965.

[89] W. Wan. *Theory and Applications of Arbitrary-Order Achromats*. PhD thesis, Michigan State University, East Lansing, Michigan, USA, 1995. also MSUCL-976.

[90] W. Wan and M. Berz. Analytical theory of arbitrary-order achromats. *Physical Review E*, 54(3):2870–2883, 1996.

[91] W. Wan, C. Johnstone, J. Holt, M. Berz, K. Makino, M. Lindemann, and B. Erdélyi. The influence of fringe fields on particle dynamics in the large hadron collider. *Nuclear Instruments and Methods*, A427:74–78, 1999.

[92] H. Wollnik. *Optics of Charged Particles*. Academic Press, Orlando, Florida, 1987.

[93] F. Zimmermann, C. Johnstone, M. Berz, B. Erdélyi, K. Makino, and W. Wan. Fringe fields and dynamic aperture in muon storage rings. Neutrino Factory/Muon Collider Notes MUC-NOTE-NEUTRINO-SRC-0095, Fermi National Accelerator Laboratory, 2000. see http://www-mucool.fnal.gov/notes/notes.html , also CERN SL 2000-011 A-P, and CERN NuFact Note 21, CERN.

Index