IEEE Transactions on Nuclear Science, Vol. NS-30, No. 4, August 1983

MARYLIE: THE MARYLAND LIE ALGEBRAIC TRANSPORT AND TRACKING CODE*

David R. Douglas† and Alex J. Dragt
University of Maryland
College Park, Maryland 20742

Introduction

MARYLIE is a Fortran-language beam transport and tracking code developed at the University of Maryland. It employs algorithms based on a Lie algebraic formalism for charged particle trajectory calculations[1], and is designed to compute transfer maps for and trace rays through single or multiple beam-line elements. This is done without the use of numerical integration or traditional matrix methods; all nonlinearities (including chromatic effects) through third (octupole) order are included. Thus, MARYLIE includes effects one order higher than those usually handled by existing matrix-based programs.

Presently, the following beam-line elements are described by MARYLIE:

- drifts
- normal-entry and parallel-faced dipole bends
- fringe fields for dipoles
- hard-edged magnetic quadrupoles
- fringe fields for quadrupoles
- hard-edged magnetic sextupoles
- hard-edged magnetic octupoles
- hard-edged electrostatic octupoles
- axial rotations
- radio frequency bunchers
- user specified transfer map (through non-linear terms of degree 3)

Particle transport calculations (ray traces) are carried out at speeds comparable to those of current matrix-based codes.

Requirements on Transport Codes

A beam transport or tracking code must compute and manipulate some representation of the transfer map describing particle trajectories through beam lines. Matrix codes, for example, do this by computing coefficients in the Taylor series expansion of the transfer function for a beam line. MARYLIE deals directly with the nonlinear transfer map itself; it computes the following approximate representation for the transfer map,

$$\mathcal{M} = \exp(:f_2:)\,\exp(:f_3:)\,\exp(:f_4:)\ . \quad (1)$$

More specifically, a beam transport code must meet three requirements. First, it must be able to compute transfer maps for individual beam-line elements. Secondly, it must be capable of combining maps for a collection of elements to yield a single map for an entire beam line. Finally, a code must be able to compute the effect of such maps on points in the phase space describing the beam (i.e., it must "trace rays"). These requirements specify, in a natural fashion, the structure employed by MARYLIE.

Structure of MARYLIE

MARYLIE employs a modular structure. Each "module" (generally, a Fortran subroutine) is designed to meet (or assist in meeting) one or more of the above requirements. Overall control within the program is governed by a "main" code which calls subroutines to perform various operations.

The lowest order part of any transfer map is described by the factor $\exp(:f_2:)$. In MARYLIE (as in matrix codes) this portion is represented by a real 6x6 matrix. The nonlinear behavior of a transfer map is specified (through third order) by the polynomials f_3 and f_4. The coefficients of the various monomials occurring in a polynomial are stored at addresses within a linear array using an algorithm given by Giorgilli[2]. Because a transfer map is completely specified by its generators f_n, this method of representation provides a unique description of any transfer map.

When parameters for desired individual elements are input into MARYLIE, a set of "library" subroutines is called. Expressions for the coefficients of the f_n's [and for the matrix representation of $\exp(:f_2:)$] are programmed into a subroutine for each type of element. Arrays containing these coefficients for individual elements are computed and stored, thereby providing the required representation for the transfer map of each element in a beam line.

A set of "array manipulation" subroutines is used to combine maps for a collection of beam-line elements in order to produce a single net map for a complete beam line. Necessary manipulations include taking matrix products of the lowest order portions of the maps, as well as computing various Poisson brackets of the polynomials generating the nonlinear portions. These subroutines employ algorithms which take advantage of the Lie group structure of the set of transfer maps; they simply evaluate the "group product" for pairs of transfer maps.

Finally, another set of subroutines performs the manipulations necessary to compute the effect on a particle of transit through a beam line. The behavior of a particle is completely specified by the six canonical variables employed in Hamilton's equations for trajectories within the beam line. An array (x,p_x,y,p_y,T,P_T) containing numerical values

*Supported in part by D.O.E. Contract Number AS05-80ER10666.
†Now at Lawrence Berkeley Laboratory.

for each of these variables (two transverse coordinate and momentum components, an arrival time deviation, and a deviation from design energy) is read in, and transformed (by the transfer map) into an image array which is read out. The output display separately tabulates the contribution to the image from the lowest order terms and from each order of nonlinearity. If desired, the coefficients of the polynomials f_3 and f_4, and the second-order and third-order transfer matrices may also be listed.

Computer Requirements and Performance

MARYLIE has been installed on the University of Maryland UNIVAC 1100/82 and on a CDC-7600 at the Los Alamos National Laboratory. We find that, for initial testing purposes, third-order calculations often demand the availability of 10 to 15 significant digits for each dynamical variable. Consequently, double precision has been employed on the UNIVAC machine (which uses a 36-bit word). This provides 16 digit precision. The 60-bit word used by the CDC machine provides adequate accuracy (14 digits) when operated in the single precision mode.

Time requirements differ for each machine (with the CDC being the faster). The UNIVAC requires approximately 125 msec/element when computing and combining transfer maps for multiple-element beam lines. Ray traces are performed at a rate of approximately 15 msec/trace. CDC times are substantially shorter. In either case, the operation times involved are much less than those required for numerical integrators, and are indeed comparable to those required for less precise (second-order) matrix calculations.

The use of the Lie algebraic formalism also allows the construction of a code with minimal memory requirements. To represent a single transfer map requires only the storage of one 6x6 matrix and one 182 element linear array (containing the 182 coefficients of monomials in f_3 and f_4). This is to be contrasted to the memory that would be required to store the several hundred matrix elements necessary for each beam-line element were one to try to implement a third-order matrix-based code.

Tests of MARYLIE

We have employed three types of tests to verify the accuracy of MARYLIE. First, we have compared the analytical expressions on which it is based to those yielded by the matrix formalism on which TRANSPORT is based.[3] These expressions agree, indicating that MARYLIE is based on correct algorithms (at least through second order).

Secondly, we have performed a variety of checks on the code to ensure that its "library" and "array manipulation" subroutines are self-consistent. Specifically, observe that the generators f_n for a transfer map of a beam-line element are in general highly nonlinear functions of the element's length. A sensitive test of both the library and array manipulation routines is thus provided if we combine the transfer map for two identical elements of a fixed length, and compare the results to the map for an element of the same type, with doubled length. The results should agree; we find that they do, to within the round off error of the computer.

A variation on this test is made by computing the transfer map for a beam line, and combining it with the map for the "mirror image" beam line with negative lengths. The result should be the identity map. We find this is the case, to within computer round off error.

The final test we have employed is to compare the output of MARYLIE with that from numerical integration programs. We observe that MARYLIE accurately reproduces the results of numerical integration; all differences are found to be of fourth or higher order in the initial conditions. These results lead us to conclude that MARYLIE is performing properly as a third-order ray-trace program.

Tracking with MARYLIE

As described so far, MARYLIE may be used to relate incoming and outgoing values of the quantities (x,p_x,y,p_y,T,P_T) for a general beam line. In the case that the beam line closes on itself, i.e. the case of a "circular" machine, MARYLIE can, in principle, be used to compute chromaticities and nonlinear corrections to the usual lattice functions. All this information is contained in the polynomials f_3 and f_4.

Because of its high speed, it is also attractive to consider using MARYLIE to compute the effect of a large number of turns in a circular machine. As it stands, the Lie algebraic representation (1) gives a transfer map which is exactly symplectic. For computational simplicity, this symplectic feature is not completely utilized at present. Currently, the action of $\mathcal{M}$ on the general initial condition (x,p_x,y,p_y,T,P_T) is expanded in a power series and all terms beyond degree 4 are discarded. The expansion is correct through terms of degree 3 since MARYLIE is a third-order code, and some terms of degree 4 are retained in order to satisfy the symplectic condition through degree 4.

However, with only a slight increase in computation time, it now appears to be possible to evaluate the effect of $\mathcal{M}$ on a general initial condition in such a way that the result is correct through degree 3, as before, but symplectic to all degrees. Consequently, one can use all that is known about the transfer map for a circular machine, and, at the same time compute the effect of a large number of turns while maintaining the symplectic condition exactly. This is an improvement over current tracking methods which, although completely symplectic, simply approximate the transfer map by impulsive kicks.

It is also worth noting that, rather than iterating the transfer map by repeated single passes through a lattice (as is done with current tracking codes), MARYLIE may be used directly to square the transfer map repeatedly to produce very high powers. That is, the sequence

$$\mathcal{M}, \mathcal{M}^2, \mathcal{M}^4, \mathcal{M}^8, \mathcal{M}^{16}, \ldots$$

can be produced. Thus, for example, the transfer map for 1024 turns can be produced in $\log_2(1024) = 10$ operations (using about 125 msec of UNIVAC time per operation), rather than in the $\sim 10^3$ iterations required by usual methods. Evidently, the execution time to compute the map for N turns increases only as $\log_2(N)$. It is believed that this procedure may be useful in cases where the transfer map is only slightly nonlinear.

The method for evaluating the transfer map while maintaining the symplectic condition to all orders, and the utility of computing M^N by successive squaring, are still under study and will be reported upon elsewhere.

Conclusion

Because of their high speed and low storage requirements, Lie algebraic methods are an efficient means of computing charged particle beam transport. The polynomials employed to generate the transfer map give a complete and succinct description of the nonlinear properties of a beam line. In the case of a circular machine, one can extract from these polynomials all desired information about nonlinear orbit properties including chromaticities and corrections to lattice functions.

At the same time, it is possible to compute high powers of a map with relatively little effort, and it appears to be possible, with only a slight increase in computer time, to maintain the symplectic condition exactly. Thus, Lie algebraic methods also appear to be well suited to particle tracking for a large number of turns.

References

1. A.J. Dragt and J.M. Finn, J. Math. Phys. 17, 2215 (1976); A.J. Dragt, IEEE Trans. Nuc. Sci. NS-26, 3601 (1979); D.R. Douglas and A.J. Dragt, IEEE Trans. Nuc. Sci. NS-28, 2522 (1981); A.J. Dragt, "Lectures on nonlinear orbit dynamics," AIP Conference Proceedings 87, R.A. Carrigan et al., editors (1982); D.R. Douglas, "Lie algebraic methods for particle accelerator theory," University of Maryland Ph.D. thesis (1982); A.J. Dragt and D.R. Douglas, Brookhaven National Laboratory Report BNL-31761, 346 (1982); A.J. Dragt and E. Forest, "Computation of nonlinear behavior using Lie algebraic methods, or how to snatch commutators out of the shambles," University of Maryland Physics Publication #83-123 (1983).

2. A. Giorgilli, Comp. Phys. Comm. 16, 331 (1979).

3. K.L. Brown, SLAC-75, revision 3 (1975).