Particle Accelerators, 1996, Vol. 55, pp. [257–262] / 11–16
Reprints available directly from the publisher
Photocopying permitted by license only

MATHEMATICA APPLICATIONS TO NONLINEAR DYNAMICS ANALYSIS

YURI ALEXAHIN

Joint Institute for Nuclear Research, Dubna, 141980 Russia

(Received 30 January 1996; in final form 30 January 1996)

Various possibilities offered by the **Mathematica** system for single particle dynamics analysis are discussed: symbolic computation in Hamiltonian perturbation theory, running tracking programs for calculating the dynamic aperture, post-processing of tracking results.

Keywords: Nonlinear dynamics; Mathematica system.

1 INTRODUCTION

Mathematica[1] is a powerful software system which can be used for

- numerical and symbolic calculations;
- data analysis and representation;
- control of external programs and processes.

It has a programming language which permits to implement practically any kind of algorithm.

There are a number of instances of **Mathematica** application to problems of particle dynamics in accelerators: generating databases of **MAD** results[2]; running **MAD** for LEP dynamic aperture calculation[3]; symbolic computations in beam optics[4]; derivation of analytical formulae in Hamiltonian perturbation theory.[5]

The objective of the present report is to give a brief description of these applications and to show on simple examples some of the **Mathematica** features which seem to be particularly useful.

2 PROCESSING OF A STANDARD PROGRAM OUTPUT

It is quite often that analysis of a large number of runs of a standard program under varying conditions (e.g. different distributions of errors) is needed. Since the output from the standard program is not always conveniently organized for the specific purpose, extracting and processing the relevant information may pose a formidable problem.

In this case **Mathematica** proves to be an indispensable tool enabling one to:

- read output of standard programs;
- extract necessary data from the output;
- accumulate the data in database;
- analyze and represent the results.

An important feature of **Mathematica** with this respect is an easy conversion between *character strings* and *expressions*. Let for example `track.n , n=1,nr,` be `nr` output files from some tracking program. To read and analyze them in a cycle over `n` **Mathematica** permits to construct the filenames in the following manner:

```
fname=StringJoin [ "track.", ToString[n]].
```

Then at a particular iteration, say `n=13`, the string `fname` value will be `"track.13"`.

As a rule the standard program output consists of intermixed text and numerical data. The simplest way to deal with such a file is to consider it as *a record*. The following command reads in a text file and converts it in *a list* of character strings, one per a line, named `mess`:

```
mess=ReadList[ fname, Record];.
```

Now each string of `mess` (i.e. each line of the output) can be tested on occurrence of a substring, converted into expressions, etc. Consider for example the **MAD** tracking module output which may contain a number of fragments of the type:

```
Particle(s) lost during turn 4, superperiod 1, s = 157.182100 by overflow in LS [3]
Number    x              px             y              py             t              pt
7  1.01550589E+00 1.16274136E-01 4.22375450E-01 -6.02355138E-02
-1.77061405E-01 6.53721162E-06
```

With the help of the auxiliary function

```
inQ[x_String]:=StringMatchQ[ x, "*Particle(s) lost*"];
```

one can determine the numbers of lines containing the message on loss and put them in the list `lines`:

```
lines=Flatten[ Position[ Map[ inQ, mess], True]];
```

Now information on the lost particles can be extracted from the subsequent lines and stored; e.g. particle number following the `i`-th occurrence of the message can be found as

```
npart=ToExpression[ First[ ReadList[
StringToStream[ mess[[ lines[[i]]+2 ]] ], Word]];
```

This chain of commands analyzes the relevant character string (the whole line) into a stream of words and then converts the first one (also a string) into a number.

It is far beyond the scope of this report to cover all the possibilities of **Mathematica** in data handling. A more extensive example of employing **Mathematica** for this purpose can be found in Ref. 2.

3 RUNNING EXTERNAL PROGRAMS

The analytical capabilities of **Mathematica** permit to use it as a supervisor for guiding external programs. If the code insides are inaccessible for **MathLink**[5] then **Mathematica** can support communication with the external program by exchanging ordinary text files (the so called *unstructured communication*). An example of operation in this mode was given in Ref. 3 where **Mathematica** was running **MAD** for dynamic aperture calculation.

The basic steps in the unstructered communication are as follows:

- preparation of a task for external program;
- launching execution of the external program;
- reading and analysis of the external program output;
- formulation of a new task.

In the particular instance **Mathematica** analyzed the **MAD** tracking module output as described in the previous section, found the maximal stable initial amplitude and gave **MAD** a new set of particle start positions for

tracking. **Mathematica** prepared input for **MAD** from prototype files by replacing dummy parameters with actual parameters using *transformation rules*. To demonstrate this mechanism let `proto` be such a file which is stored as a list and contains among others the following two lines

```
...
start, fx=&
fx0
...
```

Each line is represented by one element (character string) of the list. Now let **Mathematica** decide to start tracking with some numerical value `nv` of the initial amplitude which it assigns to variable `ax`. Then substitution

```
task=proto /. "fx0"->ax
```

will produce a file with the numerical value `nv` instead of `fx0`. Not only numerical but also logical and symbolic values may be substituted in a similar way.

The prepared input file can be fed into **MAD** by command `task>>"!mad"`

4 SYMBOLIC COMPUTATIONS IN PERTURBATION THEORY

Symbolic computations have already been extensively used in accelerator physics, e.g. for generating Lie algebraic maps (see Ref. 7). **Mathematica** had introduced some new concepts (pure functions, upvalues etc.) which proved to be useful in both linear optics[4] and nonlinear dynamics analysis.[5]

In Ref. 5 Deprit's algorithm for normalization of 2DoF nonlinear Hamiltonian was implemented as **Mathematica** notebook. With the present version of **Mathematica** for HP (which has an intrinsic limitation on memory in use) it was possible to obtain in symbolic form normalized Hamiltonian terms of fourth order in sextupole strength.

The basic idea of Deprit's algorithm is to consider transition from the unperturbed (linear) dynamics to the nonlinear one as a Lie transformation with perturbation parameter as the independent variable. This leads to a well-structured set of differential equations (Deprit's equations) which links expansion coefficients of the original Hamiltonian, the normalized one and the Lie transformation generator. This set can be easily integrated in action-angle variables.

An important concept in **Mathematica** employed at this step is the notion of *upvalues* which permits to define operations involving a function without

explicitly introducing it. In the particular case it helps to obtain the starting Hamiltonian expansion in terms of basis functions of action-angle variables

```
J[i], phi[i], i=1,2
u[e_List, m_List]:=Product[ J[i]^(e[[i]]/2), {i,2}]*

Exp[ I Sum[ phi[i]*m[[i]], {i,2}]];
```

without invocation of this explicit definition. It is sufficient to express coordinates through basis functions and then introduce the rule of multiplication as the upvalue for `u`

```
u[e1_, m1_]*u[e2_, m2_] ^:= u[e1+e2, m1+m2];
```

This allows to avoid explicit introduction of action-angle variables and makes unnecessary a large amount of mathematical operations.

Generally the solution of Deprit's equations is found in terms of integral operator which is inverse to directional (Lie) derivative. In the thin lens approximation it is reduced to a sum over nonlinear elements. The following feature of **Mathematica** facilitates to implement this reduction. Let `f[a, b, ..., z]` be any *listable* mathematical operation defined on symbolic variables `a, b, ..., z`. Substitution of the lists

```
a={a1, a2,..., aN}; b={b1, b2,..., bN}; ... z={z1, z2,..., zN};
```

into `f` will produce a list

```
{f[a1, b1, ..., z1], f[a2, b2, ..., z2], ..., f[aN, bN, ..., zN]}.
```

In this way integration of `f` is essentially reduced to summation of the list elements.

5 FUTURE DEVELOPMENTS

There are interesting possibilities of **Mathematica** application to some of the problems discussed at this Workshop, e.g. compatibility of lattice description in different standard programs.[8] **Mathematica** can easily translate from one input language to another performing simple calculations in the process (e.g. finding bending angle from field strength and energy) and even consult external programs in more complicated cases.

Another example is the Numerical Accelerator Project.[9] **Mathematica** can run in parallel with **Sixtrack** (*"peer-to-peer"* mode[6]) performing express analysis and deciding whether to launch another accompanying particle, abort tracking, etc.

References

[1] S. Wolfram: "Mathematica: A System for Doing Mathematics by Computer", 2nd ed. (Addison-Wesley Publishing Co., 1993).
[2] J.M. Jowett: "Generating Mathematica databases of MAD results, imperfect LEP2 as an example", in CERN PS/AR Note 94–17, Geneva, 1994.
[3] Y. Alexahin: "A Study of the Low Emittance Lattice for LEP2", CERN SL/94–46, Geneva, 1994.
[4] B. Autin, E. Wildner: "Beam Optics, a Program for Symbolic Beam Optics", CERN PS/DL/Note 95–18, Geneva, 1995.
[5] Y. Alexahin: "Improving the Dynamic Aperture of LEP2", CERN SL-95–110, Geneva, 1995.
[6] MathLink Reference Guide. Version 2.2. Wolfram Research, Inc. Technical Report, 1993.
[7] A. Dragt: "Lectures on Nonlinear Orbit Dynamics", 1981 Summer School on High Energy Particle Accelerators, FNAL, AIP 1982.
[8] R. Talman: "A Proposed Flat Accelerator Lattice Description", this Workshop.
[9] E. McIntosh, T. Pettersson, F. Schmidt: "Proposal for a Numerical Accelerator Pilot Project", this Workshop.