EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/LEP-TH/89-72

Recent Developments in MAD Version 8

H. Grote, F. C. Iselin, CERN, CH-1211 Geneva 23*

Geneva, Switzerland

October 30, 1989

Abstract

In view of experience with the language decoder and memory manager large parts of MAD have been rewritten recently. The dynamic memory manager is replaced by the CERN-written ZEBRA package which has been used successfully in more than 100 other programs for HEP data analysis. A new dynamic table handler keeps large tables in memory as long as they fit; it dumps them on disk or to the Cray SSD (Solid-State Storage Device) if necessary. Tables are accessed by simple subroutine calls in a manner transparent to the user. The input language has been extended to use a more object-oriented approach. Accelerator elements are described in terms of classes of objects, providing easy selection of single elements in a large machine. A powerful mechanism has been implemented for writing tables of selected optical functions and/or element parameters in selected positions of the machine. These tables are generated and written under control of the table handler, and can be read directly by the LEP control system. MAD contains a plot module with various options to plot data from internal tables. The tables can also be fed into a stand-alone plot program. At present the following program modules are complete: Command decoder; Optical functions calculation; Closed orbit correction; Harmon (chromaticity calculation); Plotting. It is foreseen that by the end of the year the following other modules will be ready: Matching including some new features; Lie Algebraic analysis; Tracking by TRANSPORT and Lie algebraic methods; Electron Beam Parameters; Polarization.

*Presented at the International Conference on Accelerator and Large Experimental Physics Control Systems, October 30 - November 3, 1989, Vancouver, Canada.

1 Introduction

The MAD (Methodical Accelerator Design) program [1] has been a useful and versatile tool for the development and understanding of accelerators over the last years, and has been used extensively in the LEP design and commissioning. It is able to perform a great variety of calculations for circular proton (antiproton) and electron (positron) accelerators. Its main features are: hierarchical definition of beam lines, calculation of the linear lattice functions including chromaticity, closed orbit correction, matching of beamlines with given constraints, tracking of particles. All results such as magnet excitation currents, Twiss parameters etc. can be printed or written into special coded files for treatment by further independent programs. All relevant parameters, as well as particle trajectories, can be plotted through a GKS interface (see Figure 1).

2 A new version

2.1 Why a new version at all ?

The new version 8 of the MAD program presented here has been started this year. The program is de facto completely rewritten, whereas the user interface (free-format input, Twiss files etc.) remains practically unchanged, and a good deal of the layout is preserved. The reason for such a major undertaking (MAD 7 has about 80,000 lines of Fortran code) were the following:

1. The table manager of the previous version which had been developed specifically for MAD to manage memory and its automatic backup on disk storage, was difficult to use and therefore error-prone, and still not bug-free in spite of several years of usage. It was felt strongly that a more generally used, well tested, and more powerful memory manager would make the addition of further modules easier. Such a package was found in ZEBRA [2], a CERN developed dynamic memory management system used by over 100 high energy physics experimental groups, including those of LEP experiments.
2. New concepts such as the element class, and a new type of standard table output, were necessary for the usage of MAD in the framework of LEP commissioning, both for the off-line preparation of magnet excitations and other parameters at different ramping energies, and for the on-line usage in the LEP control system, where MAD has to run on different types of Apollo computers for a variety of usages, including plotting.
3. Existing stand-alone program such as SMILE [3] and BEAMPARAM [4] were to be integrated with MAD. This would be simplified considerably by the introduction of ZEBRA.

2.2 Transportability

As was the case with previous versions, MAD 8 was foreseen to work on different types of computers, and was implemented and tested right from the start on IBM/VM, Cray X-MP, Vax/VMS, Apollo 3000 and 10,000 series. Except for the Cray, all computers mentioned require MAD to run in double precision.

This transportability is achieved by using ANSI Fortran 77 only, with necessary exceptions in well specified machine-dependent areas of the input/output modules, since the ANSI standard there is not precise enough and has led to different interpretations by the different manufacturers. The underlying ZEBRA package is implemented on all the systems mentioned above and many more, and is therefore fully transportable as far as MAD is concerned.

2.3 Modularity

The concept of modularity existed already in previous versions of MAD, but has been pushed even further here. MAD 8 is modular in the sense that, first of all, it is entirely data driven: there is, so to speak, no history of execution kept inside the program. Each module can be called again at any time, or not called at all. Of course, modules expect certain data to be present before they can start, and these data are very often produced by other modules. The important point is, however, that they do not know of each other, there is no "cross talk": modules use existing data and produce new data. This does not mean that all calculations are repeated when a module is called again. Much information such as beam line expansions, or element maps are kept and are reused if they are still valid, as can be seen from the timing of the Twiss calculations further down. The user, however, need not care for this, since it happens in a fully automatic fashion; if he changes an element belonging to an expanded beam line that element is replaced in the beam line without his doing when the next TWISS command is executed.

2.4 ZEBRA

The ZEBRA dynamic memory management system [2] allows to structure data residing in one or several memory "pools". The basic unit of the structure is the "bank". Banks are linear arrays the arbitrary length of which can be changed dynamically during execution. In addition, banks may be dropped (releasing the space they occupy), and new banks may be created at any time. Banks can be arranged in tree-like (top-down) and chain-like (horizontal) structures. The pointers to all banks are updated automatically during dynamic changes of the pool. Further user-defined pointers into the structure are treated in the same way: thus the same banks may actually be structured in several different ways, in order to fit different usages inside the program best, and practically without limitation.

The user of the MAD program, however, need not know ZEBRA at all: if he does not already find the information he wants in one of the different standard or user-composed output streams, he can use dedicated routines to extract information from most banks in the form of simple Fortran arrays.

3 New features

3.1 The element class

Large accelerators have a high number of individual elements, e.g. LEP has close to 9000 elements in the fully expanded beamline inside MAD. This leads to the natural introduction of an "element class". Elements of the same element class have certain parameters in common, e.g. the magnet type and the length, and differ in others, e.g. in the strength. This concept of element classes was introduced in MAD version 8 when LEP entered its final construction phase, in order to provide an easy mechanism that allows to specify common features of elements in one place. It does this by defining a sort of "template" which is then used to create individual elements. Example:

```
MQ      :  QUADRUPO, L = 1.6    !  Standard quadrupoles
MQA     :  QUADRUPO, L = 2.0    !  Straight-section quadrupoles
MQC     :  QUADRUPO, L = 2.0    !  Super-conducting quadrupoles
!----- Cell Quadrupoles
      QD      :  MQ,      K1 = KQD
      QF      :  MQ,      K1 = KQF
```

MQ defines a particular engineering type of quadrupole in MAD jargon, previously implemented by the TYPE attribute (See page 3.1 of the MAD User's Reference Manual [5]). MQ has a length L, but the default strength K1=0. The quadrupoles QF and QD are instances of the class MQ. They inherit the length L of their class, but have their strengths K1 set to non-zero values. In the jargon of MAD, the element MQ has become a keyword.

3.2 TFS table output and the OPTICS command

The computers of the LEP control system exchange information via so-called TFS tables [6]. They exist normally as coded Ascii files, but may be converted to binary where speed is a critical issue. These tables consist of a descriptive part for the contents, the date and time of creation, and the data format, followed by the data themselves in free format, arranged in columns of equal length. Each data item consists therefore of a vector of numbers, text, or logical variables. Such tables can therefore be tailored to the specific needs of a program: it is unnecessary slow to scan a file with, let us say, 28 position and optics parameters for all LEP elements (this is a standard output file) if one only needs the positions of all beam monitors.

In order to allow this the OPTICS command was introduced. It allows to select an arbitrary number of parameters per element to be written into the output file in TFS format, and furthermore a selection of elements, individually, or via classes.

Example:

```
USE, CELL
SELECT, OPTICS, SBEND, QUAD, SEXT
OPTICS, FILE = "cell.optics.f", CENTRE, COLUMN = NAME, S, BETX, BETY
```

Table 1: An OPTICS output table example

```

@ GAMTR          %f      64.3336
@ ALFA           %f      0.241615E-03
@ XIY            %f      -.455678
@ XIX            %f      2.05279
@ QY             %f      0.250049
@ QX             %f      0.249961
@ CIRCUM          %f      79.0000
@ DELTA          %f      0.000000E+00
@ COMMENT        %20s "DATA FOR TEST CELL"
@ ORIGIN         %24s "MAD 8.01      IBM - VM/CMS"
@ DATE           %08s "19/06/89"
@ TIME           %08s "09.47.40"
* NAME           S              BETX              BETY
$ %16s           %f              %f              %f
  B1              36.6600          24.8427          126.380
  SF1             37.6200          23.8830          130.925
  QF1             39.5000          23.6209          132.268
  B2              75.8000          124.709          25.2153
  SD1             77.1200          130.933          23.8718
  QD1             79.0000          132.277          23.6098

```

The resulting TFS table is listed in Table 1.

In this way, to give just one real example, tables with magnet strengths are written by MAD and fed into the control system. Likewise, MAD can read beam monitor position tables, and can create tables with corrector settings.

The TFS tables can be created on any computer where MAD 8 runs, including, of course, the Apollos of the LEP control system, and can then be shipped via networks to their destination. This is clearly useful when the calculations would take many minutes of CPU on a relatively slow Apollo 3000, and only seconds on a Cray X-MP, or IBM 3090.

3.3 Execution time

With respect to version 7, a speed-up factor of 2.5 has been obtained in the calculation of Twiss parameters as a result of improved algorithms and code optimization, in particular in one routine performing matrix operations. During the calculation of Twiss parameters the element matrices are stored away. This results in much shorter execution times for subsequent TWISS commands. Table 2 gives CPU times for several computers, both for version 7 and 8, for the first and second TWISS command. The benchmark is one TWISS command for $\delta p = 0$ on the full LEP machine. The headings "TWISS 1" and "TWISS 2" refer to the first and second execution of TWISS with the same δp respectively.

Table 2: Timing Benchmarks for MAD (times in CPU seconds)

Computer System	version 8		version 7	
	TWISS 1	TWISS 2	TWISS 1	TWISS 2
Apollo 3500	1571.	401.	3884.	3884.
Apollo 10000	72.	19.		
Cray XMP	27.9	9.5	37.8	37.8
IBM 3090	23.7	7.2	59.3	59.3

3.4 Background storage

A special module has been developed for MAD 8 that pushes banks out onto background storage and loads them back when they are needed. This only happens when the available memory becomes too small, and may therefore vary from one implementation to another. The package uses normally random access disk storage in a way as to minimize the number of I/O interrupts. On the Cray X-MP at CERN, however, it uses the SSD (Solid-State Storage Device) available there, a kind of slower bulk memory. This leads to impressive real time reductions compared with the random disk access, of the order of a factor of ten.

3.5 Plotting

Although plotting has existed in previous versions, it is probably justified to count it under the new features, since it has now been made modular in the same style as mentioned before. Consequently, there are no longer separate plot routines producing a specialized plot each, but rather a first part arranging the input data, and a general second part producing the plots from these data for all possible applications. This is greatly simplified by the usage of ZEBRA, since it allows a very flexible and complex self-describing data structure to be created for a great variety of plots. New plots can therefore easily be added when new results from new modules become available, without modifying the (unavoidably) intricate plot module code.

The plotting makes use of a high-level interface to GKS that was developed for MAD 7 already. It is in fact stand-alone and provides features such as automatic scaling, drawing and labelling axes, and others. GKS itself is an ISO and ANSI standard and as such readily available, at least in Europe. At CERN it runs on all computers mentioned before which means that the MAD plots on the Apollos in the LEP control system are produced by exactly the same code as the metafiles written on the Cray.

A graphical presentation of data becomes invaluable when large quantities of numbers have to be checked and scrutinized for possible anomalies. This is why a spin-off of the MAD plot module has been introduced as a stand-alone program in the LEP control system, allowing to plot the contents of any TFS table, be it created by MAD or elsewhere.

Figure 1: LEP low beta section no. 2

4 Status and outlook

At present, the following modules inside MAD 8 are complete: command decoder; optical functions calculation; closed orbit correction; calculation of chromaticities; plotting. The next modules to be implemented are: Matching which has already been started; Lie algebraic analysis; tracking by TRANSPORT [7] and Lie algebraic methods.

Furthermore, two existing programs will be integrated with MAD 8 and become new modules in the way: SMILE [3] performing electron spin polarization calculations to (almost) arbitrary high orders, and BEAMPARAM [4], a program providing beam parameters such as beam sizes, luminosities, beam currents, and beam lifetimes, and RF related parameters such as cavity voltage, phase shift, and coupling factors. Both programs will

be able to use much of the information already present in MAD, and will thus become considerably shorter.

Work has been started on a menu-driven interface to MAD inside the LEP control system. Its aim is to allow less experienced users such as operators to use MAD in a safe way without having to remember all the commands, or having to consult a manual. This interface will very likely be specific to the Apollo computers used there.

5 Final remark

The experience gained with the memory manager ZEBRA is conclusive: our expectations have been met fully. Apart from the fact that up to now we have not encountered a single bug in this package, its major advantage lies in the increased power both of the data structure definition, and of the safer (since automatic) pointer control as compared to the previous memory manager. The former has simplified the physics modules notably and has thus further added to the reliability of the code, the latter has improved the modularity of the program. The flexibility of the data structure has been a great help during the initial phase of the development.

References

- [1] F.C. Iselin, E. Keil and J. Niederer, Conference on Computing in High-Energy Physics, Amsterdam (1985) 316.
- [2] R. Brun and J. Zoll, CERN Computer Centre Q100 (1987).
- [3] S.R. Mane, Phys. Rev. A36 (1987) 120.
- [4] P.M. Gygi-Hanney, J.M. Jowett and E. Keil, CERN/LEP-TH/88-2 (1987).
- [5] F.C. Iselin and J. Niederer, CERN/LEP-TH/88-38 (1988).
- [6] P. Defert, P. Hofmann and R. Keyser, CERN-LEP LAW note 9 (1989).
- [7] K.L. Brown, D.C. Carey, F.C. Iselin and F. Rothacker, CERN 80-04 (1980).