XMAP: A DIFFERENTIAL ALGEBRA TOOL GENERATING ACCELERATOR MAPS

J. Irwin and S. Peggs

SSC Central Design Group*
c/o Lawrence Berkeley Laboratory
One Cyclotron Road
Berkeley, California

Introduction

Differential algebras are an offspring of the field of Nonstandard Analysis[1,2]. The importance of their application in the field of nonlinear dynamics was recognized by M. Berz[3-5], who created a FORTRAN subroutine library with the name Differential Algebra (DA) Package. This package allows for the efficient manipulation of multivariable power series, in which each variable is truncated at a specified order. XMAP is a FORTRAN program which produces the map across an accelerator beam line, in the form of a truncated power series expansion, by calling the DA package algorithms. This beam line, for example, could be one complete turn of the SSC. The expansion variables are a free mixture of any or all of the six phase space coordinates, and optical parameters, like the length or strength of a set of quadrupoles.

The scope of applications for differential algebra tools like XMAP is quite broad. For example, applications which have been considered for the SSC include:

i) lattice design,

ii) dependence of resonance widths on optical parameters,

iii) parametric dependence of smear and tune shifts, and

iv) parametric maps for operational simulations.

It should be reemphasized that in principle any parameter can be a variable. For example: magnet lengths, magnet strengths, drift lengths, rms multipole error strengths, and rms magnet displacements.

A simple example - expansion of a FODO cell

The performance of XMAP is best illustrated by means of a concrete example. Figures 1, 2, and 3 demonstrate an actual XMAP run, using a very simple example lattice representing a single FODO cell. Motion across the cell is considered only in one dimension, the horizontal, but is expanded as a polynomial map not only in X and PX, but also in KF, the strength of the focussing quadrupole in the middle of the lattice. The normalized momentum, PX, is essentially the same as the horizontal angle, X' . Despite the simplicity of this expansion, which is necessary for the sake of brevity and clarity, this example includes many of the realistic features which are present in a more typical run, with a full size SSC lattice.

Figure 1 is the file XMAP.CMD, which is immediately read on execution of XMAP. Its structure is short and simple, and includes comments by line number, after the key word END signifying the end of the active part of the file. A few more comments are in order here. Setting the relativistic factor GAMMA to infinity is a reasonable approximation for the SSC, which will have a storage energy of 20 TeV. It is necessary to specify the MAPTYPe, in order that the polynomial map will correspond exactly to the model of motion used, for example, in the tracking program TEAPOT[6]. While KF is expanded to third order, and X and PX to first order, the maximum combined order for the expansion is set at the truncated value of 3, less than 3 + 1 + 1 = 5 .

The negative sign before the expansion order −3 for KF ensures that the expansion is made about the nominal value 7071.067 which gives a 90 degree phase advance per cell. In contrast, the nominal

```
fodo.flat
fodo.xmap
0.0
1
3
kf          7071.067      -3
x              1.0e-3      1
xp             0.0         1
END
```

Comments, by line number

1 Input lattice file name (flat format)
2 Output map file name
3 GAMMA, the relativistic factor. Zero => infinite energy
4 MAPTYPe, = 0 for THINTRAK, 1 for TEAPOT
5 NTORD, the truncation order, is the maximum combined order of any coefficient
6+ List of expansion variables, terminated with 'end' or 'END'
The following are recognised as canonical coordinates:
x, px, y, py, d, and ct (UPPER case is ok).
If the expansion order is
+VE, variable is NOT overwritten with the nominal value
-VE, variable is overwritten, and expanded to ABS(order)

Figure 1. The control file, XMAP.CMD. Some of the line-by-line comments have been omitted.

initial displacement of X = 1.0e-3 meters will not be adopted - the expansion will be about X = 0.0 . So, the net result is that XMAP generates an expansion of the form

$$X_{out} = \sum_{ijk} c_{ijk}\, X_{in}^{\,i}\, PX_{in}^{\,j}\, (KF - 7071.067)^k \qquad [1]$$

where the indices and exponents are in the range

$$0 \le i \le 1, \quad 0 \le j \le 1, \quad 0 \le k \le 3, \quad i+j+k \le 3 \qquad [2]$$

A second expansion is also generated, for the output angle PX_{out} .

Figure 2 is the input file, FODO.FLAT, specified in XMAP.CMD as representing the lattice of interest. It is an ASCII file, in the so called *flat* format which is described at somewhat greater length elsewhere in these proceedings[7]. Briefly, *flat* format is a low level lattice description which is more easily read by computers than by human beings. It loosely corresponds to the *standard* format[8] which allows compatible input of the same ASCII file into many lattice design and tracking programs, for example, MAD, DIMAD, TEAPOT, and TRANSPORT. Because of its greater simplicity, fewer lines of code are necessary to read the *flat* format. This is part of a more general effort towards greater modularity in SSC software - towards more, shorter, compatible codes[9].

FODO.FLAT is broken into three sequential segments, which are shown as three parallel columns in Figure 2 . Different segments are separated by the END keyword. The first segment reads in parameter names, followed by their nominal values. Any one of these parameters is available as an expansion variable. The second segment describes model magnets. For example, a *qd* magnet is of type *quadrupole*, has a length *lquad*, a strength *kd*, a tilt of 0.0 radians (all attributes can be entered numerically), and has no statistical multipole error parameters, like b_n or σ_{bn}, attached. The *dipole* magnet has a special type, *tbend*, corresponding to the TEAPOT model of a sector bend as a drift, followed by a thin multipole, followed by a drift. The third segment describes the sequence of

*Operated by the Universities Research Association, Inc. for the Department of Energy.

CH2669-0/89/0000-1337$01.00©1989 IEEE

```
lbend          halfqf         halfqf
100.0          quadrupole     -1
               lhalfquad
angle          kf             dipole
0.01           0.0            -1
               -1
kf                            qd
7071.067       qd             -1
               quadrupole
kd             lquad          dipole
-7071.067      kd             -1
               0.0
lhalfquad      -1             halfqf
0.000001                      -1
               dipole
lquad          tbend          END
0.000002       lbend
               angle
END            0.0
               -1

               END
```

Figure 2. The input file, FODO.FLAT, representing a FODO cell in *flat* format. To save space, the three file segments as shown as three columns - in practice there is only one "word" per line.

real magnets which comprise the lattice. Real multipole field errors could also be included, if the -1 were replaced by the maximum order of the fields to be described.

When a full size SSC is described, the *flat* lattice file becomes very large. It can be generated from an ideal lattice described in a relational DataBase Management System (rDBMS) database[7], from a *standard* format ASCII file, or from a TEAPOT *machine* file - a binary file describing the complete state of a simulated accelerator. The library of flat format reading subroutines, which are linked to XMAP along with the DA package, will be modified in the near future. This will enable XMAP to accept *flat* lattices in the Standard Data Set (SDS) binary format which is becoming a central feature in SSC software development[9,10]. In this representation, the three ASCII segments become five binary blocks of data, which are very close to FORTRAN common blocks, or C structures.

Figure 3 shows the XMAP output file, with the name FODO.XMAP requested in XMAP.CMD . The actual file was about twice as long as shown here, but the part generated by the flat format input routines has been removed. After echoing the conditions specified by XMAP.CMD, and the lattice as read from FODO.FLAT (not shown here), the two expansions in the form of equation [1] are listed. Note that the nominal value of X has been reset to zero. This ASCII output can be read by other differential algebra programs, such as XORT, for further manipulations.

Performance issues

When a single turn of a more or less typical SSC is expanded to order 5 in all four transverse coordinates, truncated at a maximum combined order of 5, XMAP takes approximately 4700 CPU seconds on a Sun 4/260 workstation to do its job. The CPU time required is significantly less on one of the CRAYs available from LBL over MFENET. The amount of CPU time required to calculate a map approximately doubles every time the order of the map is increased by one. However, the amount of memory required also places a limit on the maximum practical expansion order - the nominal running size of XMAP is about 19 Megabytes. If there are V variables, expanded and truncated at order N, then the amount of memory required scales like

$$\text{memory required} \sim \frac{(N+V)!}{N!\,V!} \sim O(N^V) \qquad [3]$$

Fortunately, both speed and memory performance limits are met at about the same time for a lattice of the size of the SSC.

```
The input lattice file name is          fodo.flat
The output map file name is             fodo.xmap
Relativistic factor GAMMA  =            infinite
Maptype:                                teapot
Maximum combined order NTORD =          3

There are  3 expansion variables, as follows
NUMBER  NAME        NOMINAL VALUE     ORDER
  1      x           0.00000D+00        1
  2      xp          0.00000D+00        1
  3      kf          0.70711D+04        3

Expansion of X(out)
========================================
Coordinate  Net order    Coefficient          Exponents
   1           1     -0.20723761850049D+00     1 0 0
   1           1      0.27069675392160D+03     0 1 0
   1           2      0.13104293340445D-01     1 1 0
   1           2     -0.27069675328356D-03     1 0 1
   1           2     -0.27069675318602D-09     0 1 1
   1           3     -0.32674205015127D-05     1 1 1
   1           3      0.18046450182511D-15     1 0 2
   1           3      0.90232245342796D-22     0 1 2

Expansion of PX(out)
========================================
Coordinate  Net order    Coefficient          Exponents
   2           1     -0.14646658971054D-02     1 0 0
   2           1      0.49987497744763D+00     0 1 0
   2           2      0.14018568335795D-03     1 1 0
   2           2     -0.29263735972205D-06     1 0 1
   2           2     -0.27069675328356D-03     0 1 1
   2           3      0.47067127446516D-07     1 1 1
   2           3      0.27069675254798D-09     1 0 2
   2           3      0.18046450182511D-15     0 1 2
```

Figure 3. The map output file, FODO.XMAP, slightly modified and abbreviated for the sake of clarity.

Each power series inside the DA package has a dedicated static storage area, and a unique name. A particular name points to the first and last coefficients of a series, inside one large array. Operations on power series are made by DA subroutine calls, in which the names are passed as parameters. For example, CALL DAMUL(A,B,C) multiplies A by B, and places the result in an area labeled by C. This structural arrangement permits the programming flexibility needed to create XMAP. The fundamental calculational algorithms are programmed as if all variables were power series, since it is difficult to have FORTRAN program branches depending on whether the type of a variable is a power series or a scalar. It is quite easy to declare space allocation when the power series are set up according to input information from the XMAP.CMD control file.

At first sight, this structural approach could significantly slow program execution, by complicating normal multiplication and addition operations. So far this has not been a serious problem, but execution times will soon limit the number of variables and/or the maximum truncation order. It is one of our opinions (J.I.) that an upgraded version of the DA subroutines should become part of a math library optimized to the local hardware. This will result in important execution time savings!

It is the opinion of the other author (S.P.), that the dynamic memory allocation advantages offered by rewriting the DA package in C have already been demonstrated by the conversion of another large code, TEAPOT, from FORTRAN to C [11]. Further, the use of an object oriented language such as C++ in "overloading" operators like * (multiply) would make differential algebra tools much more transparent, in removing the subroutine calls. An object oriented project like this is already in progress[12].

Acknowledgements

Sincere thanks go to Martin Berz, Vern Paxson, Chris Saltmarsh, and Tjet Sun, for their willing contributions to this work.

References

1. Robinson, Proc. Royal Acad. Amsterdam Ser. A, 64, 432-440, (1961).
2. Laugwitz, JBer. Deutsch. Math. Vereinigung 75, 66, (1973).
3. Berz, NIM A258, 431, (1987).
4. Forest, Berz and Irwin, SSC-166, Berkeley, (1988).
5. Berz and Forest, SSC-N-571, Berkeley, (1988).
6. Schachinger and Talman, Particle Accelerators, 22, 35, (1987).
6. Schachinger and Talman, Particle Accelerators, 22, 35, (1987).
7. Barr, Peggs and Saltmarsh, these proceedings.
8. Carey and Iselin, Proc. of the SSC Summer Study, p.389, Snowmass, (1984).
9. Paxson et al, these proceedings.
10. Saltmarsh, SDS usage documentation, SSC report in preparation, Berkeley.
11. Paxson, private communications.
12. Michelotti, these proceedings.