BNL-104841-2014-TECH

AGS/AD/Tech Note No. 425;BNL-104841-2014-IR

BUMPS MODELS PRIMER

J. Niederer

February 1996

Collider Accelerator Department

Brookhaven National Laboratory

U.S. Department of Energy

USDOE Office of Science (SC)

DISCLAIMER

For Internal Distribution Only

Accelerator Division
Alternating Gradient Synchrotron Department
BROOKHAVEN NATIONAL LABORATORY
Upton, New York 11973

Accelerator Division
Technical Note

AGS/AD/Tech. Note No. 425

BUMPS MODELS PRIMER

J. Niederer

February 6, 1996

Bumps Models Primer

J. Niederer

AGS Department
Brookhaven National Laboratory

1. Overview

Among the quaint tribal customs at the AGS is the application of the MAD field error mechanism to model orbit bump situations. As this mechanism was intended for something quite different, there have been occassional complications. This note reviews the program's treatment of field errors, and gives practical examples of submitting bump data, along with some folklore, and limitations. A somewhat simpler and more useful way to explore bump behavior is given, along with graphics tools for observing the effects of various parameter changes.

2. Field Errors in MAD

The MAD field error facility was designed to provide a multipole expansion treatment of field errors based upon a base field parameter, such as the bend angle of a dipole magnet. In the input language, the use of the relative field error attributes leads to the multipole treatment. Also sets of random field error values can be applied to tracking studies that deal with dynamic apertures, and magnetic field multipole error tolerances. For the latter, the radius of application for the multipoles is needed. The main data entry attributes are in terms of products of multipole strength (K) times length (L) of field. K and L are separate attributes for the magnets themselves, and people have been known to confuse among these inputs. This variety of features can complicate a simple application of producing orbit bumps from selected bend magnet strength changes (for example, from back leg windings), particularly when the bend angle, and quadrupole and sextupole strengths are expected to change together.

2.1. The Field Error Command: Efield

The program receives field error data via attribute phrases on the **Efield** command. Each issue of such a command generates an internal field error table, with entries linked to particular magnets through pointers in a map table which contains the ordered list of lattice elements. The lattice map is produced by a **USE** command, which must precede the use of the error entries. Field errors which are described in terms of parameter expressions are evaluated only at the time that the **Efield** command is processed. Thereafter any subsequent changes in the parameter expression chain do not affect the values of the field errors employed. This means that error attributes dependent on parameter expression changes are not updated for use in DO - ... - FTwiss - Set ... - Enddo iteration loops or by Twiss or Neural Driver operations. This limitation can be bypassed somewhat by grouping the field error commands as a subroutine and calling the subroutine whenever the needed parameters are changed. However, such gimmicks may soon exhaust data base pool memory as the internal tables proliferate. Therefore a better way to describe and vary field error strengths for iterative calculations has been provided in BNL MAD. There is also a more complicated way to effect such changes by addressing the internal error table. Both ways will be discussed briefly later.

2.2. Bumps Using Absolute Values of Field Errors

For bends, bump values for the dipole part of the field are expressed with the *DBL* attribute: product of the bump strength (bend angle in radians) * magnet length (in meters). The higher order parts are expressed with the *DKL(i)* attribute(s): the product of multipole i bump strength * magnet length for each desired multipole i, up to i = 10. These input data items are accepted as given; the Error program does not further process these entries. Example 1 obtains a 1% bump contribution for a bend magnet "BBBB", using approximate AGS values and data entry styles. Parameter expressions give the base field (AngA), quadrupole component (K1A), and sextupole component(K2A) in terms of polynomial functions of momentum p. Bump fields are expressed as a fraction of these base fields by multiplying each by a factor Bump = .01.

Example 1. Absolute Bump Values

```
p           Param = 29.
AngA        Param =  .028 + F0( p )
LenA        Param = 95. * .0254
K1A         Param =  .037 + F1( p )
K2A         Param = - .03 + F2( p )

BBBB   RBend     Angle  = AngA, L = LenA,  K1 = K1A, K2 = K2A
.....

Bump   Param = .01
       Efield,        BBBB,                                        &
                      dBL       = Bump * AngA * LenA,              &
                      dBKL(1)   = Bump * K1A * LenA,               &
                      dBKL(2)   = Bump * K2A * LenA
 Or Alternatively,
       Efield,        BBBB,                                  &
                      dBL       = Bump * BBBB[Angle] * BBBB[L], &
                      dBKL(1)   = Bump * BBBB[K1]    * BBBB[L], &
                      dBKL(2)   = Bump * BBBB[K2]    * BBBB[L]
```

2.3. Bumps Using Relative Values of Field Errors

For bends, relative bump values are expressed with the *dBLR* attribute: the product of relative dipole bump strength * magnet length of the given magnet(s), and the *dKLR(i)* attribute(s): the product of relative bump multipole strength * magnet length for each desired multipole i, up to 10. An effective multipole *Radius* attribute is also expected. In this case, the multipole strengths are computed as:

dKL(i) = dKLR(i) * Angle * i ! / Radius ^^n

When applied, these error terms are further divided by the magnet length L. To submit data in the relative format, so that the result is to be relative to an already submitted value, rather than to the internal multipole result, these program supplied factors in the multipole terms have to be removed in some manner. Example 2 shows a way to balance these factors in the bump application, although there is no obvious reason to use this format over the absolute one of Example 1. The ratio of L / Angle is effectively the bend radius of the machine if the magnets are uniformly powered, so the ratio is typically called RHO. Any multipole radial dependence is ignored by setting the *Radius* attribute to 1.

Example 2. Relative Bump Values

```
RHO         Param     = LenA / AngA

        Efield,       BBBB,    Radius     = 1.,                    &
                      dBL      = Bump,                             &
                      dBKL(1)  = Bump * BBBB[K1]  * RHO,           &
                      dBKL(2)  = Bump * BBBB[K2]  * RHO / 2.
```

3. Newer Trim Field Attributes on Magnet Definitions

To simplify the description of trim field values in iterations such as bump setup explorations and orbit feedback models, trim field attributes have been added to the basic magnet element statements. Mostly this is for convenience, as the trim fields could be included in the already complicated expressions for the various main magnet fields. All trim values are expressed as signed fractions of the corresponding field component that appears on the magnet definition statement. When given, these trim values are properly interpreted and their effects updated throughout the lattice before each cycle of a lattice tracking calculation.

For Bends:

dH0 Fractional change to current bend *Angle* attribute.

dK1 Fractional change to current bend *K1* (quadrupole strength) attribute.

dK2 Fractional change to current bend *K2* (sextupole strength) attribute.

These are multiplicative factors.

Example 3. Use of Trim Field Data Items

```
BBBB  RBend   Angle= AngA,        L    = LenA,          &
              K1   = K1A,         K2   = K2A,      &
              dH0  = Bump,        dK1  = Bump,     &
              dK2  = Bump
```

For Quadrupoles:

dK1 Fractional change to current quadrupole *K1* (quadrupole strength) attribute.

For Sextupoles:

dK2 Fractional change to current sextupole *K2* (sextupole strength) attribute.

4. Reaching into the Error Tables

There are applications in which it is very useful to be able to iterate quantities buried in the derived internal error tables. One of these cases involves varying quad displacements to try to match a measured orbit to possible sources of error, such as the centering of one or more quads. The names of the alignment data group (object) variables are found in the **Aligndat** structure in the *FMDict.callx* supplemental dictionary file. The label "Aligndat" is in quotes, to be case sensitive, helping avoid accidental hits to the name. The more useful error items are *Adx, Ady,* and *Ads* for the position displacements, and *Adphi, Adtheta,* and *Adpsi* for angle errors. In Example 4 an FVary command with range and step attributes is used in iterating a horizontal displacement error (adx) in the error table for magnet Qhf2:

Example 4. Displacement Error in Table

```
V.hf2  FVary  Qhf2(1:-4)[adx]  Step = .0005,  Lower = -.0050,  Upper = .0050
```

The parentheses phrase after the Qhf2 lattice element name tells the program to look into particular internal tables for the attribute called *adx.* The items in parentheses are the so called occurrence (= 1) and source (= - 4) indices, (ioccur : isource). A source of -4 denotes using the current Matching lattice map to

obtain pointers to data groups such as the error tables in searching for the parameter name. During matching a temporary copy of this map is pulled and modified somewhat for the matching process. The occurrence count may be needed if the given element appears more than once in the expanded lattice map.

The names of the field error data group (object) variables are found in the **Fielder** structure also in the *FMDict.callx* dictionary file. The label "Fielder" is in quotes, to be case sensitive. The useful content is *Erfld(2 * 4)*. The first index keys to the X component (1) and the Y component(2). The Y is zero unless the magnet is tilted. The second index keys to the order of the multipole, starting with the lowest nominal order; for the quads, the basic quadrupole field error is stored according to index = 1. In example 5, the quad field error is reached by the FVary command.

Example 5. Field Error in Table

```
V.hf2  FVary  Qhf2(1:-4)[erfld(1,1)] Step = .001, Lower = -.02,  Upper = .021
```

The services that support these features are quite general, making it easy to add more variables simply by adding their names to one of the structure definitions. In these examples, the map is searched for the first occurrence of the lattice element called Qhf2. This map entry will have pointers to the original statement module (data object), to an internally built matrix table, and to any error tables for the element. Each of these kinds of data table is linked to a description of its contents which the program can use to locate a particular variable by name.

Another source that can be searched is the set of tables (structures) linked through the current lattice map (isource = -2). The default source, for which an index is usually not supplied, is isource = -1 for the ordinary pool objects that result from simple element definitions. Some commands, such as the Ftwiss class, may have summary tables appended, whose members can also be reached without further identification. For FTwiss, the names of variables in these summary tables may similarly be found in their structure definitions **Optics0, Optics1** in the *FMDict.callx* dictionary file.

5. Iterative Tracking

The **Runtwiss** command is a quite powerful tool for studying orbit behavior under the influence of one or more variables, accompanied by graphical displays of the resulting orbits. An example display is given on the attached figure, in which the resulting horizontal orbit is drawn for several increments in a common bump trim strength. The variables involved, with steps and limits, are given on ordinary FVary commands. The names of these FVary commands are written on a LIST or similar clone statement, the name (Vvv_bump here) of which is in turn noted on a Rundata command. Rundata commands are used to link various service commands, such as drawing, varys, Snoopy data logging, sliders, tracking and others to major iterations managed by Runtwiss and the neural and feedback simulation. A Runtwiss command uses this information, along with menu drivers and a called plot data file, to compute Twiss runs and draw the results, using screen menus to select among the variables submitted to the program.

Example 6. Partial Twissdriver Command Group

```
VZf7    Fvary,    ZF7bmp,      Step = .0020,  ... etc
VZj10   Fvary,    ZJ10bmp,     Step = .0020,  ... etc
VZh20   Fvary,    ZH20bmp,     Step = .0020,  ... etc
VZxqh   Fvary,    Xqh,         Step = 100.,   ... etc
Vvvpd   FVary,    Pdelta,      Step = .00125, ... etc

Vvv_bump  Blist = Vvvpd,  Vzxqh,  VZf7,  VZh20,  VZj10

Rdtw    Rundata   Model   = MAD,       Machine = AGS,    &
                  Vlist   = Vvv_bump,                    &
                  Draws   = DrawC5,    T.schema,         &
                .... etc
Rtww    Runtwiss  Menu,  Plotdef = "Pdef.ax",  Tempv = C5_vals
```

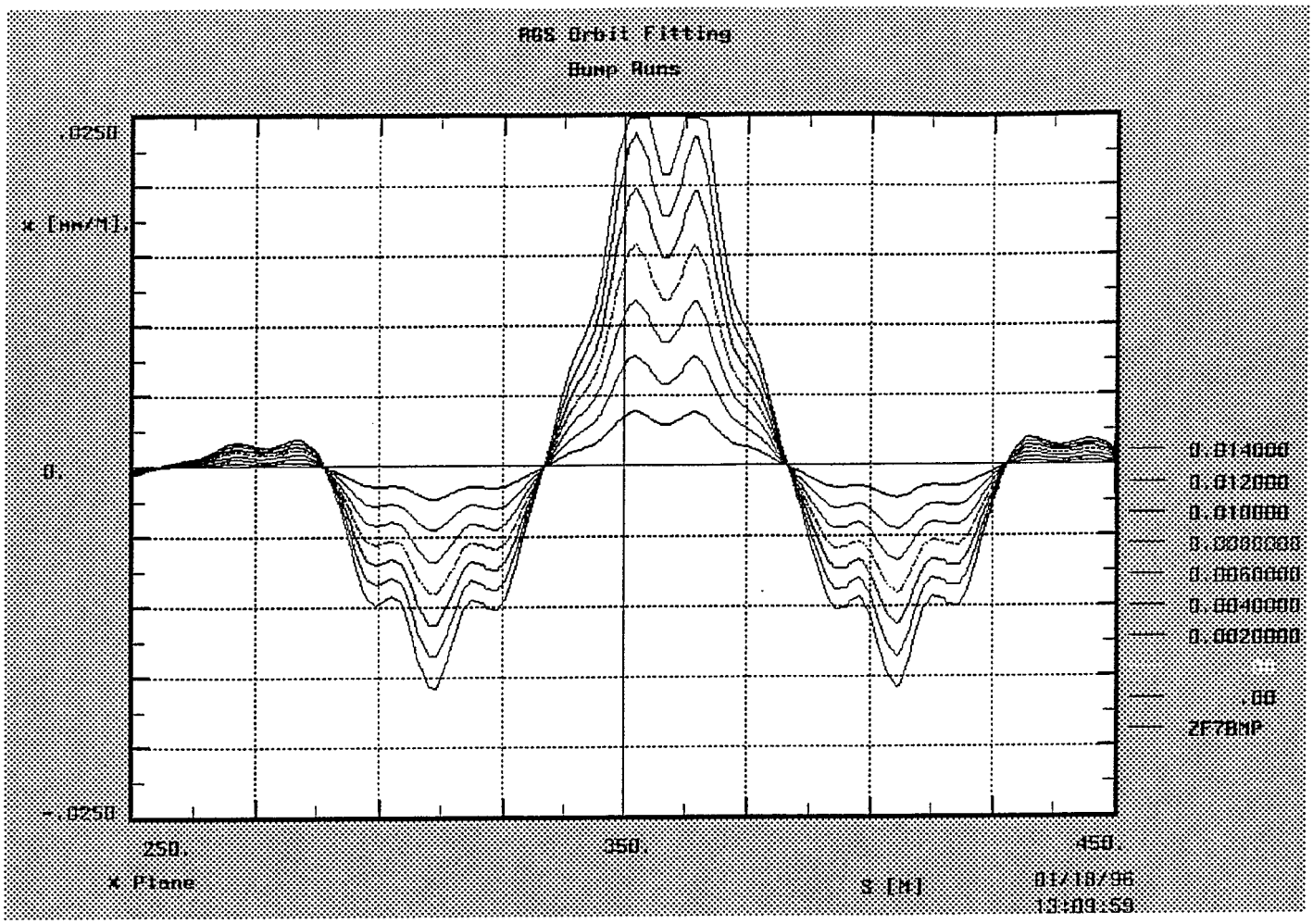
Documents

Unix Typesetter Format - nroff / psroff Files.

Host	**rapt.ags.bnl.gov**
This	/usr/disc2/jn/Docum+/Bumps.man
Runtwiss	/usr/disc2/jn/Docum+/Runtwiss.man
Plot Example	/usr/disc2/jn/Docum+/Tplots.man

To Print from rapt: (To Room 218 Ags 2nd Floor)

```
alias  it. 'cat * | psroff -t -ms > ppp; lp ppp'
    it.  Manual Name
```


ARS Orbit Fitting
Bump Runs
.0250
x [mm/mm]
0.
-.0250
250.
350.
450.
x Plane
S [m]
01/10/96
13:09:59
0.014000
0.012000
0.010000
0.00800000
0.00600000
0.00400000
0.00200000
.00
ZF78MP