GMAD Accelerator Description Language

I. Agapov[*]

March 10, 2006

Abstract

In this memo the GMAD Accelerator description language is introduced.

[*]RHUL,Egham, UK

1 Introduction

The beamline description format relevant for collimation and background studies presents a problem. Beam dynamics codes normally need only optical properties of the magnets to be specified. On the other hand, describing the geometry of a beamline in too detailed a way (say, like in [3]) is also not desired. The solution that we adopt in BDSIM [2] is to extend the MAD [1] language to take simple geometry into account. It is also allowed to specify geometries in any external formats provided the drivers to these formats are available to the code. So, a lattice description of arbitrary complexity can be achieved, from a MAD-like optics deck to a comprehensive CAD-level description.
The GMAD parser has been implemented and is shipped with the BDSIM [2] distribution. This memo shortly introduces the GMAD specification. The latest specification is available with the BDSIM distribution.

2 Lattice description

The beamline, beam properties and physics processes are specified in the input file written in the GMAD language which is a variation of MAD language extended to handle sophisticated geometry and parameters relevant to radiation transport. GMAD is described in this section. Examples of input files can be found in the BDSIM distribution in the *examples* directory.

2.1 Program structure

A GMAD program consists of a sequence of element definitions and control commands. For example, tracking a 1 GeV electron beam through a FODO cell will require a file like this:

```
qf: quadrupole, l=0.5*m, k1=0.1;
qd: quadrupole, l=0.5*m, k1=-0.1;
d: drift, l=0.5*m;
fodo : line=(qf,d,qd,d);
use,period=fodo;
beam, particle="e-",energy=1*GeV;
```

Generally, the user has to define a sequence of elements (with **drift**, **quadrupole**, **line** etc.), then select the beamline with the **use** command and specify beam parameters and other options with **beam** and **option** commands. The **sample** command controls what sort of information will be recorded during the execution.
The parser is case sensitive. However, for convenience of porting lattice descriptions from MAD the keywords can be both lower and upper case. The GMAD language is discussed in more detail below.

2.2 Arithmetical expressions

Throughout the program a standard set of arithmetical expressions is available. Every expression is ended with a semicolon. For example

```
x=1;
y=2.5-x;
z=sin(x) + log(y) - 8e-5;
```

The variables then could be used along with numerical constants. The **if-else** clause is also available, for example

```
z=1;
if(z<2)
y=2.5-x
  else
y=15;
```

2.3 Physical elements and Entities

GMAD implements almost all the standard MAD elements, but also allows to define arbitrary geometric entities and magnetic field configurations. The geometry description capabilities are extended by using "drivers" to other geometry description formats which makes interfacing and standardization easier. The syntax of a physical element declaration is

```
element_name : element_type, attributes;
```

for example

```
qd : quadrupole, l = 0.1*0.1, k1 = 0.01;
```

element_type can be of basic type or inherited. Allowed basic types are

- marker
- drift
- sbend
- rbend
- quadrupole

- sextupole
- octupole
- multipole
- vkicker
- hkicker
- rcol
- ecol
- laser
- transform3d
- element

All elements except **element** are by default modeled by an iron box (given by the **boxSize** option) with the vacuum-filled beampipe (defined by **beampipeRadius** option). An already defined element can be used as a new element type. The child element will have the attributes of the parent one as default

```
q:quadrupole, l=1*m, k1=0.1;
qq:q,k1=0.2;
```

2.3.1 Coordinate system

A standard coordibate system used in accelerator studies is assumed. The horizontal coordinates are x and x', vertical coordinates are y and y' and the longitudinal coordinates are the distrance along the nominal orbit z and the momentum. z is influenced by every compunent of nonzero length and x and y coordinates - by bending magnets and coordinate transformations **transform3d** .

2.3.2 Units

In GMAD the SI units are used.
There are some predefined numerical values
for example, instead of one can write either 100 or 0.1 * KeV when energy constants are concerned.

2.3.3 marker

marker has no effect but allows one to identify a position in the beam line (say, where a sampler will be placed). It has no attributes.
Example:

```
 m1 : marker;
```

Table 1: Units

Length	[m] (metres)
angle	[rad] (radians)
quadrupole coefficient	$[m^{-2}]$
multipole coefficient	2n poles $[m^{-n}]$
electric voltage	[MV] (Megavolts)
electric field strength	[MV/m]
particle energy	[GeV]
particle mass	$[GeV/c^2]$
particle momentum	[GeV/c]
beam current	[A] (Amperes)
particle charge	[e] (elementary charges)
emittances	[pi m mrad]

2.3.4 drift

drift defines a straight drift space. Attributes:

- **l** - length [m] (default 0)
- **aper** - aperture [m] (default same as beampipe radius)

Example :

```
d13 : drift, l=0.5*m;
```

2.3.5 rbend

rbend defines a rectangular bending magnet. Attributes:

- **l** - length [m] (default 0)
- **angle** - bending angle [rad] (default 0)
- **B** - magnetic field [T]
- **aper** - aperture [m] (default same as beampipe radius)

when **B** is set, this defines a magnet with appropriate field strength and **angle** is not taken into account. Otherwise, **B** that corresponds to bending angle **angle** for a particle in use (defined by the **beam** command, with appropriate energy and rest mass) is calculated and used in the simulations.
Example :

```
rb1 : rbend, l=0.5*m, angle = 0.01;
```

Table 2: predefined numericlal constants

pi	3.14159265358979
me	electron rest mass
mp	proton rest mass
GeV	1
eV	1^-9
KeV	10^-6
MeV	10^-3
TeV	10^3
m	1
mm	1^-3
cm	1^-2
rad	1
mrad	1^-3
clight	2.99792458e+8

2.3.6 sbend

sbend defines a sector bending magnet. Attributes:

- **l** - length [m] (default 0)
- **angle** - bending angle [rad] (default 0)
- **B** - magnetic field [T]
- **aper** - aperture [m] (default same as beampipe radius)

The meaning of **B** and **angle** is the same as for **rbend**.Example :

```
rb1 : rbend, l=0.5*m, angle = 0.01;
```

2.3.7 quadrupole

quadrupole defines a quadrupole. Attributes:

- **l** - length [m] (default 0)
- **k1** - normal quadrupole coefficient $k1 = (1/B\rho)(dB_y/dx)[m^{-2}]$ Positive **k1** means horizontal focusing of positively charged particles. (default 0)
- **ks1** - skew quadrupole coefficient $ks1 = (1/B\rho)(dB_y/dx)[m^{-2}]$ where (x,y) is now a coordinate system rotated by 45 degrees around s with respect to the normal one.(default 0).
- **tilt** [rad] - roll angle about the longitudinal axis, clockwise.

- **aper** - aperture [m] (default same as beampipe radius)

Example :

```
qf : quadrupole, l=0.5*m , k1 = 0.5 , tilt = 0.01;
```

2.3.8 sextupole

sextupole defines a sextupole. Attributes:

- **l** - length [m] (default 0)
- **k2** - normal sextupole coefficient $k2 = (1/B\rho)(d^2B_y/dx^2)[m^{-3}]$
- **ks2** - skew sextupole coefficient $ks2 = (1/B\rho)(d^2B_y/dx^2)[m^{-3}]$ where (x,y) is now a coordinate system rotated by 30 degrees around s with respect to the normal one.(default 0).
- **tilt** [rad] - roll angle about the longitudinal axis, clockwise.
- **aper** - aperture [m] (default same as beampipe radius)

Example :

```
sf : sextupole, l=0.5*m , k2 = 0.5 , tilt = 0.01;
```

2.3.9 octupole

octupole defines an octupole. Attributes:

- **l** - length [m] (default 0)
- **k2** - normal sextupole coefficient $k3 = (1/B\rho)(d^3B_y/dx^3)[m^{-4}]$ Positive **k1** means horisontal focusing of positively charged particles. (default 0)
- ks3 - skew sextupole coefficient $ks3 = (1/B\rho)(d^3B_y/dx^3)[m^{-4}]$ where (x,y) is now a coordinate system rotated by 30 degrees around s with respect to the normal one.(default 0).
- **tilt** [rad] - roll angle about the longitudinal axis, clockwise.

Example :

```
octp : octupole, l=0.5*m , k3 = 0.5 , tilt = 0.01;
```

2.3.10 multipole

will be implemented starting from v0.2

2.3.11 rcol

rcol defines a rectangular collimator
Attributes:

- **l** - length [m] (default 0)
- **xsize** - horizontal aperture [m]
- **xsize** - vertical aperture [m]
- **material** - material

Example :

```
col1 : rcol,l=0.4*m, xsize=2*mm, ysize=1*mm, material="W";
```

The longitudinal collimator structure is not taken into account. To do this the user has to describe the collimator with the generic type **element**.

2.3.12 ecol

ecol defines an elliptical collimator.Attributes:

- **l** - length [m] (default 0)
- **xsize** - horizontal aperture [m]
- **xsize** - vertical aperture [m]
- **material** - material

Example :

```
col2 : ecol,l=0.4*m, xsize=2*mm, ysize=1*mm, material="W";
```

Here the longitudinal collimator structure is also not taken into account.

2.3.13 solenoid

will be implemented starting from v0.2

2.3.14 hkicker and vkicker

hkicker and **vkicker** are equivalent to an **rbend** and an **rbend** rotated by 90 degrees respectively.

2.3.15 transform3d

An arbitrary 3-dimensional transformation of the coordinate system is done by placing a **transform3d** element in the beamline. The next element after it will be placed with respected to the new coordinates. The attributes are:

- **x** = {x offset}
- **y** = {y offset}
- **z** = {z offset}
- **phi** = {phi Euler angle}
- **theta** = {theta Euler angle}
- **psi** = {psi Euler angle}

Example:

```
d:drift,l=1*m;
rot : transform3d, psi=pi/2;
test:line=(d,rot,sb);
```

Here the sector bend will act in the vertical plane.

2.3.16 element

All the elements are in principle examples of a general type **element** which can represent an arbitrary geometric entity with arbitrary field maps. Its attributes are

- geometry = {¡geometry_description}
- bmap = {bmap_description}

Descriptions are of the form

```
"format:filename"
```

where **filename** is the path to the file with the geometry description and **format** defines the geometry description format and
Example :

```
qq : element, geometry = "mokka:qq.geom", bmap ="mokka:qq.bmap";
```

Possible formats are specific to each geometry driver and described in **??**.

2.3.17 line

elements are grouped into sequences by the **line** command.

```
 line_name : line=(element_1,element_2,...);
```

where element_n can be any element or another line. For example, a sequence of FODO cells can be defines as

```
qf: quadrupole, l=0.5, k1=0.1;
qd: quadrupole, l=0.5, k1=-0.1;
d: drift, l=0.5;
fodo : line=(qf,d,qd,d);
section : line=(fodo,fodo,fodo);
beamline : line=(section,section,section);
```

2.3.18 laser

laser defines a drift section with a laser beam inside.

```
<laser_name>: laser, position = {<x>,<y>,<z>, direction={ <dx>, <dy>, <dz> }
              wavelen=<val>, spotsize=<val>, intensity=<val>;
```

Attributes

- **l** - length of the drift section
- **position** - position of an arbitrary point on the beam axis relative to the center of the drift section
- **direction** - vector pointing in the beam direction
- **wavelen** - laser wave length [m]
- **spotsize** - spot size (sigma)[m]
- **intensity** -[W]

the laser is considered to be the intersection of the laser beaam with the volume of the drift section. For example

```
laser1: laser, l=10*cm, position={0.,0.,0.}, direction={1.,0.,0.},
               wavelen=532e-9*m, spotsize=1e-6*m, intensity=10e6;
```

2.3.19 Element number

when several elements with the same name are present in the beamline they can be accessesd by their number in the sequence. In the next example the sampler is put before the second drift

```
bl:line=(d,d,d);
sample,rang=d[2];
```

2.3.20 Element attributes

Elements attributes such as length, multipole coefficiens etc. can be accessed by putting square brackets after the element name, for example

```
d:drift,l=0.5*m;
x=d[l];
```

2.3.21 Material table

There is a set of predefined materials for use in elements such as collimators, f.e. "Al", "W", "Iron", "Copper", "Graphite" etc. Note that each geometry driver such as Mokka has its own set of materials

2.4 Run control and output

The execution control is performed in the GMAD input file through **option** and **sample** commands. How the results are recorded is controlledby the **sample** command. When the visualization is turned on, it is also controlled through Geant4 command prompt

2.4.1 option

Most of the options in **BDSIM** are set up by the command

```
option, name=value,...;
```

The following options influence the geometry

```
beampipeRadius     - default beampipe radius [m]
beampipeThickness  - default beampipe thickness [m]
tunnelRadius       - tunnel Radius [m]
boxSize            - default accelerator component size [m]
```

The following options influence the tracking and output

```
deltaChord           - chord finder precision
deltaIntersection    - boundary intersection precision
chordStepMinimum     - minumum step size
lengthSafety         - element overlap safety
thresholdCutCharged  - charged particle cutoff energy
thresholdCutPhotons  - photon cutoff energy
randomSeed           - seed for the random number generator
stopTracks           - if set, tracks are terminated after interaction with
                       material and energy deposit recorded
physicsList          - determines the set of physics processes used
ngenerate            - number of primery particles fires when in batch mode
nperfile             - number of events recorded per file
```

For a more detailed description of how the option influence the tracking see **??**

2.4.2 beam

The parameters related to the beam are given by the **beam** command

```
beam, name=value,...;
```

The available parameters are

```
particle           - particle name, "e-","e+","gamma","proton" etc.
energy             - particle energy
distrType          - type of distribution
distrFile          - input bunch file
```

```
beam, particle="e+",energy=100*MeV, distrType=="gauss";
```

2.4.3 sample

To record the tracking results one uses the **sample** command:

```
sample, range=<name>;
```

puts a plane sampler before element ¡name¿.

```
csample, range=<range>, l=<l>, r=<r>;
```

puts a cylindrical sampler of length l and radius r around element ¡name¿,
Example :

```
d:drift,l=1*m;
sample, range=d;
csample, range=d;
```

2.4.4 use

use command selects the beam line for study

```
test:line=(sb,d,d,qf);
use, period=test;
```

Acknowledgement

References

[1] ”MAD-X User’s Guide”, http://mad.home.cern.ch/mad/uguide.html

[2] http://flc.pp.rhul.ac.uk/bdsim.html

[3] ”Geant4 User’s guide”, http://wwwasd.web.cern.ch/wwwasd/geant4