BDSIM User's Manual v0.1

bdsim development group

last updated October 12, 2005

Table of Contents

BDSIM beta User's Manual

This file is updated automatically from 'manual.texi', which was last updated on October 12, 2005.

1 About BDSIM

BDSIM is a particle tracking code designed for simulation of particle transport in accelerator beam lines. It is intended for use in such problems as collimation design, background studies, laser beam diagnostics etc, See [Blair], page 12.

2 Obtaining, Installing and Running

To run BDSIM you will need a file with accelerator geometry definition (optics) and various run-time parameters (input cards).

BDSIM is invoked by the command `bdsim` '`options`'

where the options are

`--batch`
: run in batch mode

`--verbose`
: verbose

`--file = file`
: use *file* for GMAD input

BDSIM can be also invoked by the `bdsimrun` shell script which handles batch job support and opther features.

`bdsimrun` '`options`'

`-b`
: run in batch mode

`-v`
: verbose

`-f file`
: use *file* for GMAD input

3 Lattice description

The beamline, beam properties and physics processes are specified in the input file written in the GMAD language.[1] This language is a variation of the MAD language and is described in this section.

---

[1] Note: To increase flexibility, application spectrum and compatibility GMAD will probably be replaced by an XML-based lattice description. This however needs more investigation.

3.1 Program structure

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

```
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);
use,period=fodo,range=#s/#e;
beam, particle=electron,energy=1;
```

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 in this section.

3.2 Physical elements and Entities

GMAD implements almost all standard MAD elements, but also allows to define arbitrary geometric entities and magnetic field configurations. The syntax of a physical element declaration is

`element : element_type, attributes;`

for example

`qd : quadrupole, l = 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

An already defined element can be used as a new element type. Its attributes are then inherited.

3.2.1 Coordinate system

3.2.2 Units

In GMAD the SI units are used.

`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`
[eV/c**2]

`particle momentum`
[eV/c]

`beam current`
[A] (Amperes)

`particle charge`
[e] (elementary charges)

`emittances`
[pi m mrad]

There are some predefined numerical values

`pi` 3.14159265358979

`me` electron rest mass

`mp` proton rest mass

`KeV` 10^3 (in [eV] units)

`MeV` 10^6 (in [eV] units)

`GeV` 10^9 (in [eV] units)

`TeV` 10^12

for example, instead of one can write either 100 or 0.1 * KeV when energy constants are concerned.

3.2.3 marker

`marker` has no effect but allows one to identify a position in the beam line. It has no attributes.

Example:

```
m1 : marker;
```

3.2.4 drift

`drift` defines a straight drift space. Attributes:

`l` - length [m] (default 0)

Example :

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

3.2.5 rbend

`rbend` defines a rectangulat bending magnet. Attributes:

`l` - length [m] (default 0)

`angle` - bending angle [rad] (default 0)

`B` - magnetic field [T]

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, angle = 0.01;
```

3.2.6 sbend

`sbend` defines a sector bending magnet. Attributes:

`l` - length [m] (default 0)

`angle` - bending angle [rad] (default 0)

`B` - magnetic field [T]

Example :

The meaning of `B` and `angle` is the same as for `rbend`.

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

3.2.7 quadrupole

`quadrupole` defines a quadrupole. Attributes:

`l` - length [m] (default 0)

`k1` - normal quadrupole coefficient k1 = (1/B rho) (dBy / dx) [m^-2] Positive `k1` means horisontal focusing of positively charged particles. (default 0)

ks1 - skew quadrupole coefficient ks1 = (1/B rho) (dBy / 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.

Example :

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

3.2.8 sextupole

`sextupole` defines a sextupole. Attributes:

`l` - length [m] (default 0)

`k2` - normal sextupole coefficient k2 = (1/B rho) (d^2 By / dx^2) [m^-3] Positive `k1` means horisontal focusing of positively charged particles. (default 0)

ks1 - skew sextupole coefficient ks2 = (1/B rho) (d^2 By / 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.

Example :

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

3.2.9 octupole

3.2.10 multipole

3.2.11 collimator

`collimator` defines a collimator

Attributes:

`l` - length [m] (default 0)

`aperture` - aperture , defined by the `aperture` element

`material` - material , defined by `material`

Example :

```
coll : collimator,l=1, aperture=<aperture>, material=<material>
```

3.2.12 solenoid

3.2.13 coordinate transformation

3.2.14 element

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

`geometry` = <geometry_description>

`bmap` = <bmap_description>

`emap` = <emap_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. The possible formats are given in Appendix A [Geometry], page 11.

Example :

`qq : element, geometry = plain:qq.geom, bmap = plain:qq.bmap;`

`<bmap>` and `<emap>` are definitions of E and B field maps according to (field maps).

3.2.15 line

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

`line_name` `: line=(`*`element1`*`,`*`element2`*`,...);`

element`n` can be any element or another line.

Example :

A sequence of three 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);
beamline : line{fodo,fodo,fodo};
```

3.2.16 aperture

3.2.17 material

`<material> : material,Z=,A=,density=,temperature=`

Attributes

`Z` - atomic number

`A` - mass number

`density` - [kg/m]

`temperature` [K]

3.2.18 pipe

the beam pipe parameters are used for particle tracking inside elements when the use geometry is not defined. The beam pipe radius is assigned by the `pipe` command

`pipe, range=<range>, range=, r=, thickness=, material=<material>;`

Attributes

`range` - element range to assign the radius for

`r` - radius [m]

`thickness` - thickness [m]

`material` - beam pipe material

Example :

Supposing we want to define a copper beam pipe for ...

```
iron : material, Z=1,A=1,density=100, temperature=;
copper : material,Z=,A=,density=,temperature=;;

fodo : line=(qf,d,qd,d);
pipe, range=qf/qd, r=0.2, thickness = 0.1,material=copper;
pipe, range=d[2], r=0.1, thickness = 0.05,material=iron;
```

3.2.19 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.

3.2.20 gas

gas command is used to introduce gas into the beam pipe.[2]

gas, period=, components={c1,c2,...},parts={p1,p2,...} ;

where

c1,c2,... - gas components names

p1,p2,... - parts (100%=1). They need not sum up to 1.

the gas componens are defined by

c1 : gas, name=<name>, A=<A>, Z=<Z>, profile=<profile_name>;

where

<Z> - atomic number

<A> - mass number

<profile_name> - name of gas profile definition

the gas profile is defined as

<profile_name> : gas_profile = (<element>:<pressure>, <element>:<pressure>)
;

where

<element> - name of the beamline component

<pressure> - gas pressure (bar)

The gas pressure is then interpolated between the points where it is defined. Issuing multiple gas commands acts additively.

Example :

To introduce the gas into a fodo cell

```
...element definitions...

fodo : line=(qf,d,qd,d);

co2 : gas, name="c02", Z=22,A=44,profile=co2profile;
h20 : gas, name="h2", Z=1,A=1,profile=co2profile;

c02profile : gas_profile = (qd:0.01, qf:0.02*nbar,d:0.03*nbar);
h20profile : gas_profile = (qd:0.04, qf:0.01*nbar,d:0.03*nbar);

gas, period=fodo,components= { c02,h20},parts={0.7,0.8};
```

[2] in realistic situations the gas profile can vary in transverse dimensions. This is not taken into account for a)technical reasons b)one does not know the profile anyway

3.3 Run control and output

The execution control is performed in the GMAD input file through `option` and `beam` commands. How the results are recorded is controlledby the `sample` command. When the visualization is turned on, it is however controlled through command prompt and has different syntax (Geant4 syntax).

3.3.1 option

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

the baseic options are

`use_low_em_physics`
: use low em physics

`turn_on_synhrotron`
: turn on synchrotron

Other options (more advanced) can be used to control the tracking procedure. For a full list if options and their effects see Chapter 5 [Physics], page 10

Example :

`beam, particle=electron,energy=100, momentum=1,1,1;`

3.3.2 beam

3.3.3 sample

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

`sample, range=<range>,particle=<particle>,values={value1,value2,...}`

the parameters are

- `element`
- `range`
- `particle` - particle to record. One `sample` command activates sampling only for one particle type
- `value`
  - `x` - horizontal
  - `px` - horizontal momentum
  - `xx` -
  - `y`
  - `py`
  - `E` - energy
  - `id` - track id. This enables later trajectory analysis.

Example :

```
sample, element=qd,particle=electron, values={x,px,y,py,e,id };
sample, range=qd/qf:0.1*m,particle=photon, values={x,px,y,py,e,id };
```

3.3.4 `use`

`use` command selects the beam line for study

`use, period=,range=`

3.3.5 visualization control

when bdsim is invoked in interactive mode, the run is controlled by the Geant4 shell. Some examples

`/run/beamOn 100` runs the simulation with 100 particles

To display help menu

`/help`;

For more details see [Geant], page 12.

4 Visualization

Visualization system description

5 Physics

5.1 Transportation

5.1.1 Tracking of charged particles in EM fields of low multipole order

5.1.2 Tracking of charged particles in arbitrary EM fileds

5.1.3 Neutron transport

not implemented yet

5.2 Showers

5.3 Synchrotron Radiation

5.4 Bremsstrahlung

5.5 Compton

5.6 Gas scattering

5.7 Tuning the tracking procedure

6 Implementation Notes

BDSIM uses Geant4 libraries and execution control mechanism.

Appendix A Geometry description formats

The element with user-defined physical geometry is defined by command

`<element_name> : element, geometry=format:filename, attributes`

for example,

`colli : element, geometry=plain:colli.geom`

A.1 `gmad` format

`gmad` is the format which can basically describe elements with axial simmetry and should work for most accelerator components like collimators, RF cavities etc.

The idea is to represent the elements as sequence of simpler elements which have axial simmetry and ... (See [geometry_image], page 11.)

Example :

For example, an absorber of NLC type can be represented by the following code

A.2 `mokka`

mokka desciption...

Appendix B Field description formats

The element with user-defined physical geometry is defined by command

`<element_name> : element, geometry=format:filename, attributes`

for example,

`colli : element, geometry=plain:colli.geom`

Appendix C Bunch description formats

7 Authors

Name	Email	Module Responsibilities
Ilya Agapov	agapov@pp.rhul.ac.uk	Project Management, Parser, Physics
Grahame Blair	blair@pp.rhul.ac.uk	Physics,
John Carter	carter@pp.rhul.ac.uk	Physics, Mokka detector Interface
Olivier Dadoun	dadoun@lal.in2p3.fr	Visualization

8 References

1. G. Blair, Simulation of the CLIC Beam Delivery System Using BDSIM, CLIC Note 509
2. Root User's Guide, http://root.cern.ch/root/doc/RootDoc.html
3. Geant4 User's Guide, http://wwwasd.web.cern.ch/wwwasd/geant4/G4UsersDocuments/Overview/html
4. MAD-X User's Guide, http://mad.home.cern.ch/mad/uguide.html
5. NLC Zero-order design report